English

FUJITSU

LMS V3.4B

SDF Format

Edition June 2013

Comments… Suggestions… Corrections…

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

Feel free to send us your comments by e-mail to:
manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

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

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

1 Preface

This manual describes the functions and mode of operation of the Library Maintenance System (LMS).

1.1 Brief product description

The Library Maintenance System (LMS) creates and manages program libraries and processes the members they contain.

Program libraries are BS2000/OSD PAM files which are processed using the library access method PLAM (Program Library Access Method); for this reason they are also known as PLAM libraries.

The Library Maintenance System (LMS) can also be called by a user program as a subroutine. This provides the user with convenient facilities for processing LMS libraries and their contents directly from his main program. This subroutine interface is described in a separate manual [1].

LMS supports the software development process using the borrowing mechanism and make functionality.

1.2 Target group

This manual is aimed at BS2000/OSD users who use the Library Maintenance System (LMS) for managing their files and members.

You should be familiar with BS2000/OSD, in particular with its major commands.

You should also be familiar with the BS2000 command language SDF (System Dialog Facility) since the user interface, screen layout and operator guidance of LMS are defined by SDF. Appropriate information can be found in [3].

1.3 Summary of contents

This manual covers the following topics:

- **Preface**
  A brief description of the BS2000/OSD product LMS and information on using the manual
- **Characteristics of LMS**
  An overview of the concepts used in LMS, such as program libraries, members and member types, and of the input and output capabilities of LMS
- **Program libraries**
  Information about the structure and accessing of program libraries
- **Members**
  A description of the member types and of the applicable convention for the member designation, notes on multiple selection and the construction specification, a description of member attributes and the relationships between members, information about version management and member protection.
- **LMS functions**
  An outline of the facilities provided by LMS
- **Support for the software development process**
  the make functionality and the borrowing mechanism
- **Statements**
  All statements in alphabetical order
- **Format of LMS outputs in S variables**
  Statements in which S variables (system variables) can be generated
- **Examples**
  Selected examples for using LMS
- **Compatibility**
  Notes on BS2000 version dependencies
- **Appendix**
  Information on compatibility and version dependencies (BS2000/OSD-BC, products)

You will find an index and a list of related publications at the back of the manual.

1.4 Notational conventions

This symbol indicates that the subsequent, indented paragraph contains important or indispensable information.

Note
The word "*Note*" before an indented paragraph indicates that the subsequent paragraph contains important information.

[1]
Numbers enclosed in square brackets are references to the corresponding manual in the list of related publications at the end of the manual.

`[ ]`
Square brackets in SDF syntax examples: the characters within the brackets are optional.

Boldface
In SDF syntax examples, the actual lines concerned are shown in boldface.
The rules as described in the relevant part of the reference section apply for the remaining syntax examples.

`SYNTAX/example`
SDF syntax and example inputs/outputs are distinguished via different fonts. Syntax examples are also enclosed in frames.

1.5 Readme file

The functional changes to the current product version and revisions to this manual are described in the product-specific Readme file.

Readme files are available to you online in addition to the product manuals under the various products at *http://manuals.ts.fujitsu.com*. You will also find the Readme files on the Softbook DVD.

Information under BS2000/OSD

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

`SYSRME.<product>.<version>.<lang>`

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `/SHOW-FILE` command or an editor. The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

Additional product information

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

1.6 Changes since LMS V3.3A

Messages

The "Messages" chapter is no longer included.
You can find the messages on the manual server (URL: *http://manuals.ts.fujitsu.com*) by means of an HTML application and on the "BS2000/OSD SoftBooks" DVD.

The main new features are:

- Changes with LMS V3.3B
  - The statement OPEN-LIBRARY has been extended by the operand SNAPSET. From BS2000/OSD-BC V7.0 on, files can be accessed from snapsets.
  - As of PLAM V3.4A the CCSN of newly created elements will be derived from the CCSN of the library file unless the program creating the element doesn't specify a value of its own.

- Changes with LMS V3.4A
  - As of BS2000/OSD-BC V8.0, BS2000 supports the SQ-Series based on Intel X86 processors. Elements generated for this SQ-Series with type L can be displayed with //SHOW-ELEMENT and updated with //MODIFY-ELEMENT. They are identified by HSI-CODE: X86.
  - LMS calls EDT V17 in compatibility mode by default so that existing procedures run furthermore without changes. During editing of Unicode elements, EDT switches over to Unicode mode automatically.

    To edit elements with records longer than 255 characters but no Unicode character set specified please use the following commands after you started editing with LMS statement //EDIT-ELEMENT respectively EDT:

    ```
    @MODE OP=UNICODE
    @OPEN L=<lib>(<elem>)
    @CLOSE
    ```
  - The statement MODIFY-ELEMENT-ATTRIBUTES has been extended by the new value *LIBRARY-DEFAULT for the operand CODED-CHARACTER-SET.
  - The statement MODIFY-LMS-DEFAULTS has been extended by the new operand EDT-MODE.
  - The statement SHOW-LMS-DEFAULTS has been extended by the new value EDT-MODE for the DEFAULTS operand.
  - The statement CALL-EDT has been extended by the new operand EDT-MODE.
  - The statement EDIT-ELEMENT has been extended by the new operand EDT-MODE.
- Changes with LMS V3.4B
  - The new EDIT-ELEMENT-ATTRIBUTES statement starts the guided dialog mechanism for the MODIFY-ELEMENT-ATTRIBUTES statement.
  - The new EDIT-ELEMENT-PROTECTION statement starts the guided dialog mechanism for the MODIFY-ELEMENT-PROTECTION statement.

1.7 Software configuration requirements

LMS V3.4B runs on BS2000/OSD-BC V6.0 or later. Additionaly the following is necessary for selected functions:

– BS2000/OSD-BC as of V7.0, if libraries on snapsets are to be opened. Snapsets are copies of a pubset on the base of snap units. These snapsets created by the system-administration can be used as a logical backup of all files on a pubset.

– EDT for editing of text elements and for the output of the LMS log to an EDT work file. For the editing of Unicode elements EDT V17.0 or later is necessary.

– SDF-P for input from and output to S variables and for full usage of the make functionality

– SECOS data protection with GUARDS and SAT logging of security member accesses

2 Characteristics of LMS

This section provides a brief overview of the characteristics of LMS:

– definition of concepts: program library, member and delta method
– input and output capabilities of LMS
– mode of operation of LMS, illustrated by an example
– behavior of LMS in interactive and batch modes

2.1 Program libraries

LMS uses **program libraries** as the library format. A program library is a file with a substructure. It contains members and a directory of the stored members. Program libraries serve to store source programs, macros, object modules, phases (load modules), lists, procedures, text etc. (see chapter “Program libraries” on page 27 for details).

2.2 Members

A member is a logically coherent data set such as a file, a procedure, an object module or a source program. Each member can be individually addressed within the library by way of its member designation. **In the context of LMS the terms “member” and “element” are used synonymously**.

The member designation identifies a member and consists of three parts: name, version and type.

Name: The name component describes the logical contents of the member.

Version: The version component describes the current development state of the member.

Type: The type component serves to classify the members.

For a description of the member types, see page 33ff.

2.3 Delta method

If the delta method is used, only the differences (deltas) to each preceding version are stored when a member has several versions. This saves storage space. When such versions are read, LMS merges these deltas at the appropriate locations so that the complete member is available to the user. For further information on filing members using the delta method, see page 91ff.

2.4 Input and output stream

LMS reads all its inputs via the dialog interface SDF. See [3] for details.

The following figure illustrates the LMS input and output capabilities:

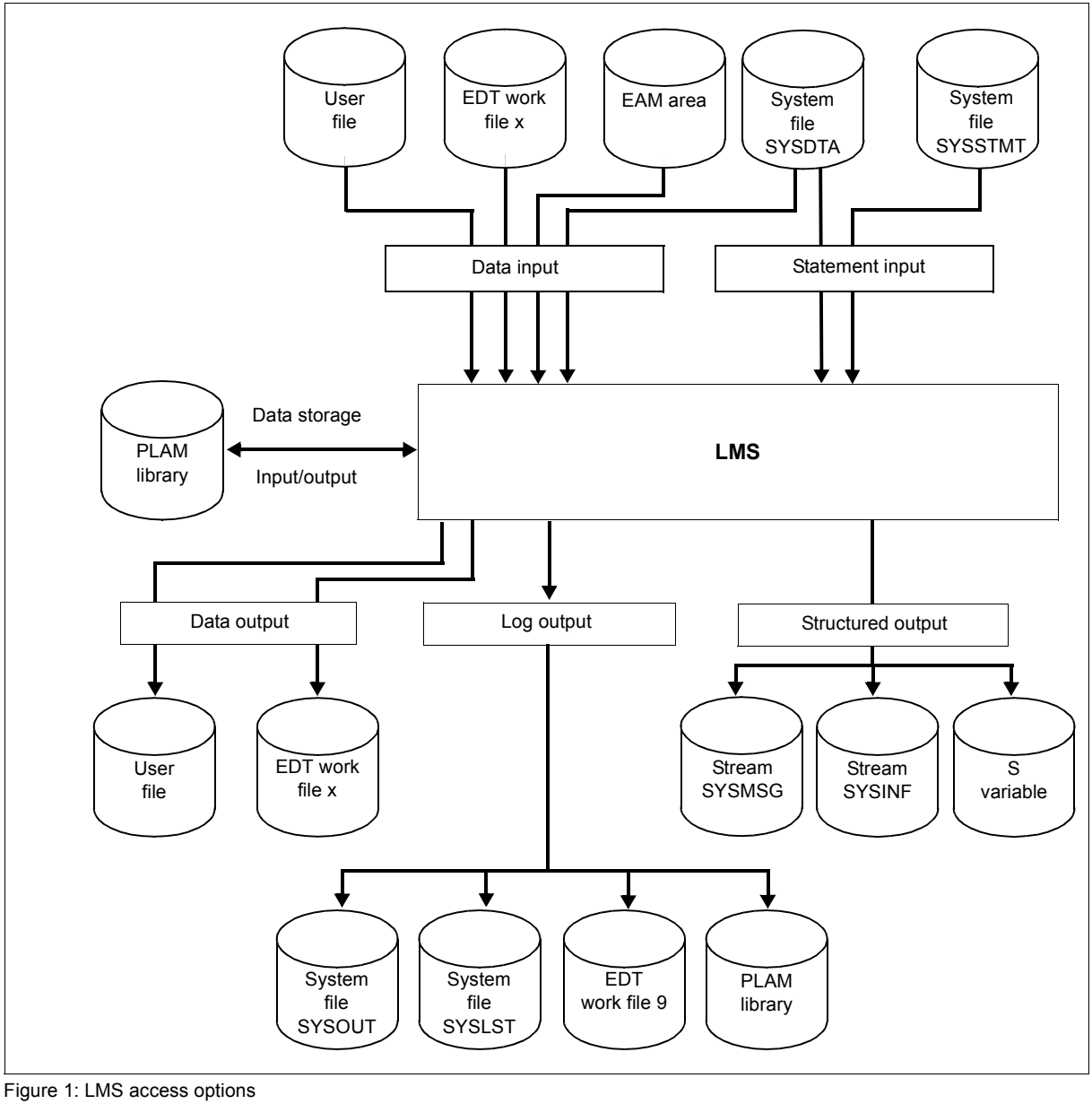


Figure 1: LMS access options

Output redirection

LMS output is to SYSOUT, unless directed elsewhere. With the LMS statement MODIFY-LOGGING-PARAMETERS, it is possible to redirect LMS output to the system file SYSLST, to a user-defined library member or to EDT work file 9, or to switch output off.

Example of an LMS run

The following example shows a short LMS run in order to provide an initial impression of the way in which LMS operates. To make the example executable, the two files A.EXAMPLE and A.SOURCE.A must exist under the user ID. The user inputs are indicated by lowercase letters and bold type, the LMS messages by uppercase letters.

The example contains the following functions:

– start LMS program
– open new library
– add new member A.SOURCE.A to the library
– add new member A.EXAMPLE to the library and simultaneously rename it
– output directory of the library
– terminate LMS program

```
/show-file-attributes file-name=a.
0000003 :N:$USER.A.EXAMPLE
0000003 :N:$USER.A.SOURCE.A
:N:  PUBLIC:      2 FILES. RES=        6, FREE=        4, REL=        0 PAGES
/start-lms ──────────────────────────────────────────────────────────────── (1)
//open-library library=lib1,mode=*update ──────────────────────────────────── (2)
//add-element from-file=a.source.a,to-elem=*lib(type=s) ──────────────────── (3)
//modify-logging-parameters logging=*maximum ──────────────────────────────── (4)
//add-element from-file=a.example,to-elem=*lib(elem=examp,type=d) ──────── (5)

INPUT  FILE
OUTPUT LIBRARY= :N:$USER.LIB1 ──────────────────────────────────────────────── (6)

         ADD :N:$USER.A.EXAMPLE AS (D)EXAMP/@(0001)/2012-05-29
```

```
//show-element-attributes ──────────────────────────────────────────────  (7)
INPUT  LIBRARY= :N:$USER.LIB1 ────────────────────────────────────────────  (8)

TYP NAME  VER (VAR#) DATE

(D) EXAMP  @  (0001) 2011-08-02

   1 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS

TYP NAME       VER (VAR#) DATE

(S) A.SOURCE.A  @  (0001) 2011-08-02

     1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//end ──────────────────────────────────────────────────────────────────────  (9)

/
```

(1) LMS is called.

(2) LMS creates LIB1 as a global library. The library is opened with UPDATE, i.e. it is possible both to read in the library and also to write to the library.
If the library does not yet exist, it will be created.

(3) File A.SOURCE.A is added to the library as an S-type member and is not renamed.

(4) The option for the LMS logging scope is changed to maximum. LMS outputs a complete log, i.e. not only error messages but also positive acknowledgments.

(5) File A.EXAMPLE is added to the library as a D-type member with the member designation EXAMP.

(6) Positive acknowledgment: since the logging output scope was changed to maximum (cf. (4)), LMS confirms the inclusion of file A.EXAMPLE as member EXAMP.

(7) The directory of program library LIB1 is to be listed.

(8) Directory entry of program library LIB1.

(9) LMS is terminated.

2.5 Input and output in S variables (SDF-P)

S variables are always replaced automatically at input by the SDF dialog interface (see the manual “SDF Dialog Interface” [3]), which LMS uses for all inputs.

LMS supports the output of selected data in S variables, which always produces lists of structures (see the manual “SDF-P” [12]). In the supported statements, output in the S variable can be switched locally using the operand STRUCTURE-OUTPUT=, apart from (independent of) the output stream.

Information can be output in S variables with the following statements:

COMPARE-ELEMENT
FIND-ELEMENT
SHOW-ELEMENT-ATTRIBUTES
SHOW-LIBRARY-ATTRIBUTES
SHOW-STATISTICS
SHOW-TYPE-ATTRIBUTES

The variable must be declared as follows:

/DECL-VAR varname(TYPE=*STRUC),MULT-ELEM=*LIST

"varname" is a freely selectable name. The variable can also be output via the SYSINF stream. In this case, the command /ASSIGN-STREAM STREAM=SYSINF, TO=*VAR(varname) must be executed before the statement that is to create the variable.

The command /ASSIGN-STREAM STREAM=SYSINF,TO=*DUMMY stops the creation of variables. LMS checks the SYSINF assignment internally if STRUCTURE-OUTPUT= *SYSINF applies, and, for reasons of performance, it only produces output in the SYSINF stream if the SYSINF assignment (direct or via SYSVAR) is not equal to *DUMMY.

Support for MIP variables

Selected LMS messages can be stored in MIP variables (MIP = Message Improvement Program). MIP variables are S variables in which the MIP product stores messages. The MIP variables can be evaluated in S procedures in order to control further operation. The following messages are provided for MIP variables:

```
LMS0302  ELEMENT (&00) NOT FOUND
         (&00): element designation in format
                *LIB-ELEM(library,element(version),type)

LMS0303  ELEMENT (&00) NOT IN RANGE OF REFERENCE CONDITION
         (&00): element designation (format, see LMS0302)

LMS0310  LMS VERSION '(&00)' STARTED
         (&00): LMS version in format dd.dldd   d=digit, l=letter
```

```
LMS0311  LMS VERSION '(&00)' TERMINATED NORMALLY
         (&00): LMS  version (format, see LMS0310)

LMS0312  LMS VERSION '(&00)' TERMINATED ABNORMALLY
         (&00): LMS  version (format, see LMS0310)
```

These messages appear in the message file with the attribute WARRANTY=*YES. The warranty applies to the message code and the inserts only, and not to the message text.

To be able to use MIP variables in LMS, the user must do the following:

1. Declare the MIP variable as follows:

   ```
   /DECL-VAR varname(TYPE=*STRUC),MULT-ELEM=*LIST
   ```

   "varname" is a freely selectable name. The above declaration defines a list of structures. Each member in the list can hold one message and has the following format (e.g. the i-th list member):

   varname#i.MSG-TEXTcomplete message text
   varname#i.MSG-IDmessage code
   varname#i.I0insert 0
   :
   varname#i.In insert n

   The number of inserts depends on the message.

2. Enter the command

   ```
   /ASSIGN-STREAM STREAM=SYSMSG,TO=*VAR(VAR-NAME=varname)
   ```

   to save the messages to the MIP variable, and the command

   ```
   /ASSIGN-STREAM STREAM=SYSMSG,TO=*DUMMY
   ```

   to stop this save process. All commands and statements executed in the meantime (not only those from LMS) save their guaranteed messages to the MIP variable.

Examples:

– You want to check whether the LIB library is empty. In this case, LMS issues message LMS0302. The variable is to be called LMSMIP.

```
/DECL-VAR LMSMIP(TYPE=*STRUC),MULT-ELEM=*LIST
/START-LMS
//HOLD-PROGRAM
/ASSIGN-STREAM SYSMSG,TO=*VAR(LMSMIP)
/RESUME-PROGRAM
//SHOW-ELEM-ATTR *LIB(LIB),TEXT-OUTPUT=*NONE
//HOLD-PROGRAM
/ASSIGN-STREAM SYSMSG,TO=*DUMMY
/IF  (IS-DECLARED('LMSMIP#1.MSG-ID'))
/    IF  (LMSMIP#1.MSG-ID='LMS0302')
/    WRITE-TEXT 'Library empty'
/    END-IF
/END-IF
/RESUME-PROGRAM
:
:
```

– Non-guaranteed LMS messages can be intercepted using the SDF-P built-in function STMT-SPINOFF [13].

```
/ASSIGN-SYSDTA TO=*SYSCMD
/BEGIN-BLOCK DATA-INSERTION=YES
/START-LMS
//OPEN-LIB XXX
/A=STMT-SPINOFF()
//STEP
/IF (A EQ 'YES')
//  WRITE-COM 'OPEN ERROR'
/ELSE
//  SHOW-ELEM-ATTR
/END-IF
//END
/END-BLOCK
```

2.6 Support for program development

LMS supports the process of program development by providing the “borrowing” and “return” functions for managing members (referred to in this manual as the “borrowing mechanism”) and by offering an efficient means of updating program systems (referred to as the “make functionality”).

The **borrowing mechanism** provides a means of controlling access to members which are being worked on by two or more developers in a project. The mechanism is intended to prevent a member (member version) or a sequence of members (“development line”) from being modified by two or more persons simultaneously. Once the mechanism has been activated for a library or member, only the user who has been entered as the current holder of the original version (base version) is allowed to save modifications. When the member is saved, certain information is added to the member’s new status, including the time stamp of the return, the user ID of the holder and perhaps a user-specific comment. The processing status of the member is output as an attribute in the directory and can be used as a selection criterion.

The **make functionality** describes the dependencies which exist between library members (including the DMS files) of a program system and the rules (“actions”) for updating them. LMS provides users with a number of make substatements, and the standard SDF statements may also be used.

For a given target component (“TARGET-OBJECT”), the make functionality describes the original components (“FROM-OBJECTs”) on which the target is dependent and the actions (“ACTIONS”) which must be performed to generate the target component.
These actions may be specified either directly or in the form of “standard” actions. Standard actions need be declared only once for a given pair of member types (“TARGET-TYPE”, “FROM-TYPE”). The components can also be referenced in the individual steps of an action (“text lines”) by means of S variables. Starting with the selected target components, all the targets are newly generated if the original components of a line have been changed since the target was last generated. Not just the selected target components, but all the related original components represent further subtargets which are handled analogously. The functionality generates a BS2000 procedure which can be started synchronously or asynchronously. make offers a means of efficiently updating program systems since it carries out only those actions which are absolutely necessary.

In order to be available for future use, the sequence of make substatements should be stored in its own member, which is called a “makefile”.

2.7 LMS in interactive/batch mode

LMS runs in interactive or batch mode.

The LMS log is output to system file SYSOUT or to the medium defined by means of MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT= (system file SYSLST, library member or EDT work file 9). If LMS is to output positive acknowledgments as well as error messages, the logging scope must be defined using MODIFY-LOGGING-PARAMETERS LOGGING=*MAXIMUM.

Interactive mode

Since it is also possible to select members by specifying “wildcards”, it is not immediately apparent which member is currently being processed. A step-by-step approach is therefore advisable in the case of statements which delete or overwrite member data. This is controlled by means of a dialog with the user.

In interactive mode, LMS offers an optional inquiry mechanism which allows the user to control the execution of statements that process members.

The user is asked for each member whether he wishes to process or skip the member or to abort the statement.

The inquiry mechanism is activated by means of the operand DIALOG-CONTROL =*YES. The inquiry mechanism is permitted in the ADD-, COPY-, DELETE-, EDIT-, EXTRACT- and MODIFY-ELEMENT statements and also in the MODIFY-LMS-DEFAULTS statements.

In interactive mode, an inquiry mechanism is activated in the following cases:

– A member cannot be accessed, e.g. because it is locked by another user (temporary exclusive use of a member, e.g. due to modification).

– A library cannot be accessed, e.g. because the current access rights do not permit access (temporary exclusive use of a library).

Batch mode

In case a library, member or type should be locked, the user can set the NEXT-ATTEMPT operand of the MODIFY-LMS-DEFAULTS statement to define the number of open attempts to be made and the time interval between them. The default setting is that no further attempts are made.

2.8 Addressing mode

LMS runs in 31-bit addressing mode by default. When called as a main program, LMS is started directly in 31-bit addressing mode.

When called via subroutine interfaces, LMS switches to 31-bit addressing mode when needed.

The addressing mode in which a user exit is called is determined by the AMODE attribute of the activated user routine:

AMODE = 31 / *ANY :	31-bit addressing mode
AMODE = 24 :	24-bit addressing mode

3 Program libraries

LMS processes PLAM libraries. PLAM libraries are PAM files in BS2000 that are processed with the library access method PLAM (Program Library Access Method).

A PLAM library contains members and a table of contents or directory of these stored members. PLAM libraries are used for the storage of source programs, macros, object modules, phases (load modules), lists, procedures, text etc. They are characterized by the following:

– all member types can be processed in a single library with standardized statements,

– members with identical names may exist which are distinguished only by their type or version designation,

– the library can be accessed by several users simultaneously, even in write mode,

– for most data elements (=members) produced during a software development process, uniform data storage exists with standardized access functions and

– the utility routines and compilers can access this stored data and process the individual members directly.

Program libraries are also referred to simply as “libraries” in the following:

3.1 Structure of a library

A **library** is a file with a substructure. It contains members and a directory (table of contents, TOC).

A member (also referred to as “element” in examples and messages) is a logically related set of data, e.g. a procedure, an object module or a source program. Each member of a library can be referenced individually.

Storing a number of files as members in a library decreases the burden on the file name catalog since each library has only one catalog entry. Storage space is saved because the members are always stored in compressed form in the library and, furthermore, may also be stored as delta members.

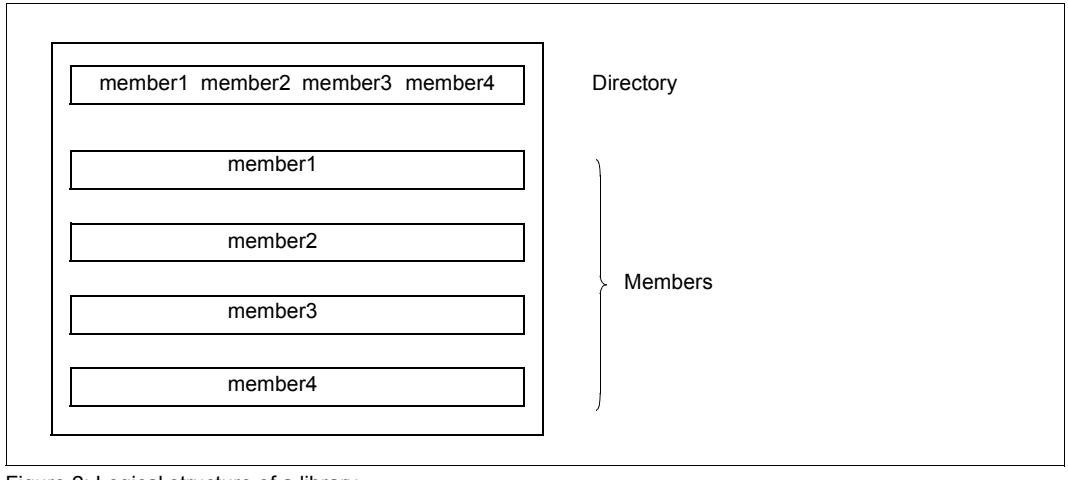


Figure 2: Logical structure of a library

Each library has a single entry in the system catalog. The user can define the name and other file attributes such as the retention period or shareability. Catalog entries and changes to them are made by the user with the aid of system commands.

3.2 Input and output libraries

LMS processes a library in the form of an input and/or output library:

An **input library** is assigned globally by means of the LMS statement OPEN-LIBRARY or locally by means of the operand ELEMENT=*LIBRARY-ELEMENT(LIBRARY =).

An **output library** is assigned globally by means of the LMS statement OPEN-LIBRARY or locally by means of the operand TO-ELEMENT=*LIBRARY-ELEMENT(LIBRARY =).

For further information on global and local libraries, see section “Library assignment” on page 74.

3.3 Multiple access to libraries

Multiple access to libraries occurs when several users access the same library at the same time, either in read or in write mode. The accesses can be made from within different tasks and the tasks can be assigned to different user IDs.

Multiple access to libraries is permitted only for those users who have the appropriate access right.

A library can be opened by one or more users in write mode as well as in read mode.

Below is a summary of the limiting conditions under which multiple access is possible:

- Libraries on public volume set (PVS)
  Unrestricted multiple access is possible, provided
  - the necessary access rights have been granted
  - access is not via Remote File Access (RFA)
- Libraries on shared public volume set (SPVS)

  Unrestricted multiple access is possible, provided
  - the necessary access rights have been granted,
  - access is not via Remote File Access (RFA), and
  - all accessing tasks from the various computers form a HIPLEX cluster.

- Libraries on shared private disk (SPD)
  Unrestricted multiple access is possible, provided
  - the necessary access rights have been granted,
  - access is not via Remote File Access (RFA), and
  - all accessing tasks are running on the same computer. If tasks are accessing from different computers, only read accesses are possible simultaneously. If the library is being write-accessed by a task, it cannot be accessed by tasks from other computers.

3.4 Restricting multiple access

The user can restrict the use of multiple access by means of the following commands:

`/ADD-FILE-LINK ...,SUPPORT=*DISK(SHARED-UPDATE=*NO)`

PLAM libraries are opened by default with SHARED-UPDATE=*YES. An ADD-FILE-LINK command with SHARED-UPDATE=*NO and the opening of this library with the specified link name prevents further update accesses.

`/SECURE-RESOURCE-ALLOCATION FILE=`

The library is reserved exclusively for this task.

`/MODIFY-FILE-ATTRIBUTES ...,PROTECTION=`

Only users who have the necessary access rights can access the library.

For a description of these commands, see [4].

4 Members

A member is a set of logically related data, e.g. a file, procedure, object module or source program. Each member can be addressed individually in the library by way of its member designation. The member designation identifies a member and consists of three parts: name, version and type.

Name: Describes the logical contents of the member.

Version: Describes the current development status of the member.

Type: Classifies the member.

Program libraries may contain any LMS-supported member types.

The program library features permit all data associated with a project, from source program, through object modules and phases (load modules), compilation procedures and test data to documentation, to be stored in the appropriate storage units of a library.

Types of storage

The user has a choice of two forms of storage for textual members: members can be stored as non-delta or delta members. The type of storage is controlled by means of the STORAGE-FORM operand in the LMS statements ADD-ELEMENT, COPY-ELEMENT, EDIT-ELEMENT, MODIFY-LMS-DEFAULTS, MODIFY-LIBRARY-ATTRIBUTES, MODIFY-TYPE-ATTRIBUTES, PROVIDE-ELEMENT and RETURN-ELEMENT.

When the delta method is used, only the differences (deltas) to the previous version are stored whenever several versions of a member are present. This helps save even more storage space. When such member versions are read, LMS merges these deltas at the appropriate locations. The user is thus always offered the complete member. In addition, hierarchical relationships can be established between members (delta sequence, delta tree).

4.1 Multiple access to members

A non-delta member can be read simultaneously by several users; it can, however, be written to by one user only. When a non-delta member has been opened for writing, no other access - including read access - to this member can be performed, but access to other non-delta members of the library is possible.

A delta member can be read simultaneously by more than one user.

If a delta member is open for writing, then no members in the data tree concerned which are of the same type and have the same name can be read or written to by other users.

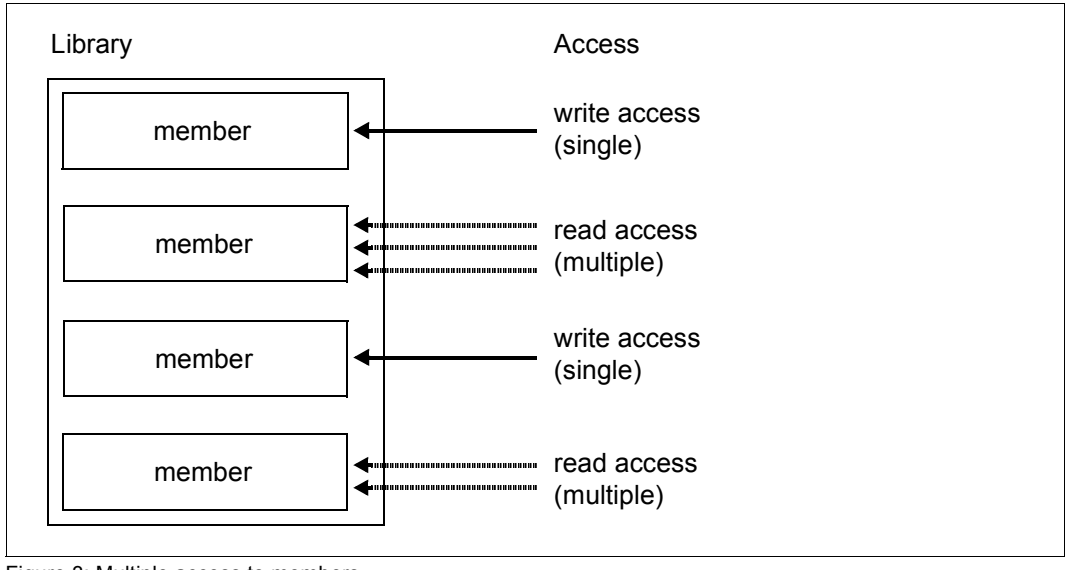


Figure 3: Multiple access to members

As a result of the multiple access options to a library a member may still exist while the directory is being listed, but be no longer in existence when it is subsequently accessed: another user has deleted it in the meantime.
A listing of the library's directory (see the LMS statement SHOW-ELEMENT-ATTRIBUTES) will therefore only show the current state of the input library.

The user is responsible for the logical coordination of accesses to the library members.

4.2 Member type definition

The member type indicates how the contents of the LMS members are to be interpreted.

4.2.1 Standard types

Standard or predefined types are one character long or they begin with $ or SYS:

Type	Meaning
$...	Reserved
C	Phases (load modules)
D	Documents
F	IFG format masks
H	Compiler result information
J	Procedures and ENTER jobs
L	Link and load modules (LLMs)
M	Macros
P	Edited data
R	Object modules
S	Source programs
SYS...	Reserved
SYSJ..	Compiled procedures
U	IFG user profiles
X	Data of any format

Table 1: Standard types

The maximum record length of stored members is 32 Kbytes (including record header).

EDT processes text members with a maximum record length of 256 bytes.

Member type $... - reserved types

Types starting with $ are reserved.

Member type C - phases

A phase generated by the linkage editor TSOSLNK is normally stored in a file. LMS can be directed to write such a file as a C-type member to a library. Alternatively, the phases generated by the linkage editor can be stored directly in a library.

Member type D - documents

Type D members are intended for documents.

Member type F - IFG format masks

Members of this type are generated by IFG and stored in libraries.

Member type H - compiler result information

Members of this type are generated by the compilers and the assembler and stored in libraries.

Member type J - procedures and ENTER jobs

BS2000 procedures and ENTER jobs can be stored in this member type.

Member type L - link and load modules (LLM)

Both the linkage editor BINDER (see [5]) and the compiler store the generated link and load modules (LLMs) in members of this type.

Member type M - macros

The assembler takes the macro members referenced in the program from the assigned library.

Member type P - list members

Edited data is referred to as a list member. The first character of the record must be a valid feed control character; this is checked on output to SYSLST.

Member type R - object modules

Object modules generated by the compilers or the assembler are normally stored in the temporary EAM area.
LMS can be directed to write such object modules as R-type members to the library. Alternatively, the object modules generated by the compilers or the assembler can be stored directly in a library.

These members serve as input to the linkage editors and the dynamic binder loader DBL.

Member type S - source programs

Source programs in libraries can be used as input to compilers and assembler for language processor runs.

Member type SYS... - reserved types

Types starting with SYS are reserved.

Member type SYSJ.. - compiled procedures

Compiled procedures are stored in this member type. Member types SYSJ and J are handled identically by LMS.

Member type U - IFG user profiles

Members of this type are generated by IFG and stored in libraries.

Member type X - data of any format

The type X member can accept any data. They are stored in text or PAM members, depending on the data format.

Textual member types - text members

Textual member types include the types S, M, J, P, D and X, as well as other types derived from them. Text members are members of these types, provided they contain no block-oriented records.

PAM members

In the following sections, members with block-oriented records are also referred to as PAM members (of type X or derivatives of type X) because as a rule they resulted from a file of type FILE-STRUCTURE=PAM being added as a member.

4.2.2 Derived types

In addition to using the standard types, users can derive their own types (called (user-defined types). User-defined types are between two and eight characters long and start with $ or SYS. A redirection mechanism allows user-defined types to be used instead of standard types in all commands and programs (see section “Library assignment” on page 74).

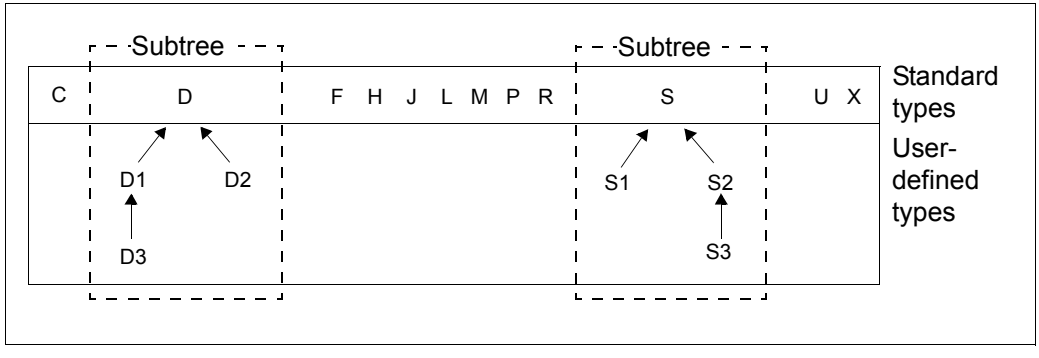
Supertype

Each user-defined type can be assigned to a supertype, i.e. a type above it in the type tree, with MODIFY-TYPE-ATTRIBUTES. LMS then processes the members of a user-defined type in the same way as it processes the members of its supertype. The supertype itself can, on the other hand, be a user-defined type. This may result in several type trees being created, which should each have a standard type at its peak.

Base type

The type at the peak of a type tree is called the base type and should be a standard type. It defines how the user types derived from it are to be handled by LMS.

Example



D has no supertype and is therefore its own base type.
D1 has D as its supertype and its base type.
D3 has D1 as its supertype and D as its base type.

4.3 Convention for member designations

Members are identified in libraries by means of a **member designation**. This is stored in the directory of the library and can be output using the LMS statement SHOW-ELEMENT-ATTRIBUTES.

The member designation comprises the following three components:

– member name for the logical contents of the members

– member version for the current status of the members

– member type for classification of the members

4.3.1 Member designations in statements

The member designation, i.e. member name, version and type, in the LMS statements corresponds to the operands ELEMENT, VERSION and TYPE in the data structure (LIBRARY-)ELEMENT.

The specification of the version is optional. If no value is specified for the version in a statement, the member having the highest value is selected by default. If a different version is to be selected, note that the version specification must be a substructure of the member name:

```
    .
    .
  ELEMENT = <composed-name ...>(...)

    <composed-name ...>(...)
      |
      |  VERSION = ....
      |
      | ,BASE = .....

 ,TYPE = .....
    .
    .
```

When LMS is called, the member type has the value *UNDEFINED by default. It must be defined globally with the MODIFY-LMS-OPTIONS statement, or locally in every statement.

Syntax of member designations

The notational conventions used here are described on page 139ff. They are the same for source and target members.

ELEMENT	<composed-name 1..64 with underscore_with wildcards(132)>(...) The member name may begin with a catalog ID not exceeding four characters in length.
VERSION	<composed-name1..24 with underscore_with wildcards(52)> For further information on possible version entries, with particular regard to the use and meaning of keywords, see page 50. '@' is not permitted as a version entry since it is used to represent the version specification '*UPPER-LIMIT'.
BASE	<composed-name 1..24 with underscore_with wildcards> For further information on possible entries, see page 52ff.
TYPE	<alphanum-name 1..8 with wildcards(20)> Member types which consist of only a single character or which start with $ or SYS are interpreted as standard types. These are reserved types and should not be used as user-defined types. User-defined types are at least two and not more than eight characters in length.

4.3.2 Logging the member designations

The member designations are logged as follows with each output from LMS:

```
(type)membername/version[(variantnumber)]/date
```

The variant number is set to (0001) by default and is incremented by 1 by each write access.

4.3.3 Multiple selection of member designations

If certain members are to be selected in the LMS statements for processing, this can be done in two ways:

– through the use of “wildcard” specifications in the ELEMENT, VERSION and TYPE operands.

  The “wildcard” syntax is described in the syntax description section on page 138ff.

  It is also possible to use the keyword *ALL for an individual asterisk (*) in the operands ELEMENT, VERSION and TYPE, but not in the EXCEPT-ELEMENT structure.

  Negative selection, i.e. exclusion of members, can also be effected through the EXCEPT-ELEMENT operand as well as with the minus sign. This operand offers more options than the minus sign.

– through the qualification of attributes, e.g. date and time.

Examples of multiple selection

– Multiple selection

```
ELEMENT = AB/C*
```

  All members whose names begin with AB, have any character in the 3rd, and a C in the 4th position are selected. The contents from the 5th position are freely selectable.

```
ELEMENT = <:999>(VERSION=B*)
```

  All members having a name length of up to 3 characters and with a B in the first position of the version, are selected.

```
ELEMENT = *(VERSION=*),CREATION-DATE = INTERVAL(FROM=2013-01-01)
```

  All members entered since 1.1.2013 are selected.

```
ELEMENT = AB*(VERSION=*HIGHEST-EXISTING)
```

  The highest version of all members whose name begins with AB is selected each time.

– Multiple selection with limiting values

```
ELEMENT = A*(VERSION = *HIGH), USER-DATE = INTERVAL(TO=12-12-31)
```

  All members of the highest version whose names begin with A and which have a date earlier than 1.1.2013 are selected.

```
ELEMENT = AB<:9>(V=107)
```

  All members whose names begin with AB, are up to 3 characters long and have the version 107 are selected.

– Multiple selection with members for exclusion

```
ELEMENT = -ABC
```

All members except member ABC are selected.

```
ELEMENT = A*, EXCEPT-ELEMENT = (ELEMENT = *ANY(VERSION = B*))
```

All members whose names begin with A and whose versions do not begin with B are selected.

```
ELEMENT = *(VERSION = <402:>*), EXCEPT-ELEMENT=(EL=*(V=402))
```

All members whose version is greater than 402 are selected.

```
ELEMENT = L<:999>, EXCEPT-ELEMENT = (ELEMENT=*ANY(VERSION=1))
```

All members whose names begin with L and are up to 4 characters long, except those having version 1, are selected.

4.3.4 Construction specification for member designations

For those LMS statements which permit a second member designation in addition to the multiple selection, the designation of the second member can be constructed from the designation of the multiple selection.

Such a construction specification is possible with the following LMS statements:

– ADD-ELEMENT
– COMPARE-ELEMENT
– COPY-ELEMENT
– EDIT-ELEMENT
– EXTRACT-ELEMENT
– MODIFY-ELEMENT
– MODIFY-ELEMENT-ATTRIBUTES
– PROVIDE-ELEMENT
– RETURN-ELEMENT

The construction specification is restricted to the member designation, i.e. to member name, version and type. This corresponds in the statements to the ELEMENT, VERSION and TYPE operands in the data structure LIBRARY-ELEMENT. In each case here like-named operands, which are identified by means of certain wildcard characters (see “Syntax for construction specification” below), are mapped onto one another:

Multiple selection:

```
ELEMENT    =*LIBRARY-ELEMENT(ELEMENT=...(VERSION=..),TYPE=...)
```

Construction specification:

```
TO-ELEMENT =*LIBRARY-ELEMENT(ELEMENT=...(VERSION=..),TYPE=...)
```

The syntax is described in section “Syntax description” on page 138ff.

Notes

– The EXCEPT-ELEMENT operand is ignored for construction specifications.

– If you use a wildcard in the selection specification, you will need to enter at least one wildcard in the construction specification. Entering *ALL in the selection specification has the same effect as a single asterisk (*) and thus requires a wildcard in the construction specification.

  *ALL cannot be entered in the construction specification (except in COMPARE-ELEMENT).
  (Example: *ALL ⟶ *B* is equivalent to * ⟶ *B*)

4.4 Member attributes

All members have certain attributes, regardless of their type. Here a distinction is made between attributes assigned by the access method and attributes which can be assigned by the user. The values of these attributes are output when the library directory is output.

4.4.1 Attributes assigned by the access method

The access method records the following member attributes:

CREATION-DATE and CREATION-TIME	(1)
MODIFICATION-DATE and MODIFICATION-TIME	(2)
ACCESS-DATE and ACCESS-TIME	(3)
STORAGE-FORM	
SECONDARY-NAME and SECONDARY-ATTRIBUTE	
Member size	(4)

(1): In the case of delta members, this date is the same for all deltas and refers to the time at which the initial data was generated.

(2): The MODIFICATION-DATE operand of the MODIFY-ELEMENT-ATTRIBUTES statement is used to specify whether this time stamp is to be updated.

(3): This time stamp is recorded only if MODIFY-LIBRARY-ATTRIBUTES ...,ACCESS-DATE=*KEEP has been set for the library.
Only after this setting has taken effect can members have an access date and time. Except for accesses via Remote File Access (RFA), no write privilege for the library file is required to have access dates and times recorded. If you wish to have access dates and times recorded when accessing via RFA but have no write privilege for the library file, your access request will be rejected.
For libraries on NK disks, PLAM determines the scope of any partial backup. The backup granulate is a single member. Members for which only the access date has changed are not included in partial backups. In the event that a backup is restored, these members are given the time of the last full backup of the library file as an access date.

(4): In the case of delta members, the size refers to the memory allocation of the whole delta tree and not to that of the individual versions.

4.4.2 Attributes that can be assigned by users

Users can specify the following attributes in the LMS statements MODIFY-ELEMENT-ATTRIBUTES or MODIFY-ELEMENT-PROTECTION:

– Organizational attributes:
  access rights (PROTECTION)
  member status (STATE)

– Content-descriptive attributes:
  user date and time (USER-DATE and USER-TIME)
  name of a character set (CODED-CHARACTER-SET)

4.4.3 Input format for dates

The input format for dates is:

[YY]YY-MM-DD	[YY]YY	:	Year; choice of two-digit or four-digit entry
	MM	:	Month
	DD	:	Day

When the year is entered as two digits, LMS uses a reference year to add the other two digits for the century :

If YY < 60, LMS precedes YY with 20, i.e. 20YY, and

if YY ≥ 60, LMS precedes YY with 19, i.e. 19YY.

4.5 Relationships between members

The following means exist for describing temporal and logical interdependencies, i.e. the relationship between the members:

Delta tree

A delta tree is a set of members which is formed by the relationship “member-is-successor-of”.

Naming conventions

Within a branch of a delta tree the members have the same name and type. They are differentiated only by their version.

Reference entries

Reference entries are entries in the secondary directory of the library. They occur when the user generates reference records (record type 163) in the form <secondary-name> <secondary-attribute> during writing of a member. The layout of record type 163 is described in the manual “LMS Subroutine Interface” [1].

The reference entries serve to document the relationship “A particular <secondary-name> and <secondary-attribute> occurs in the member”. Type-based sorting of the reference entries permits the query:

“In which member of type TYPE does a particular reference entry occur?”.

This is implemented by means of the following LMS statement:

```
//SHOW-ELEMENT-ATTRIBUTES *LIB(*STD,*,TYPE=...,SECONDARY-NAME=..,
                           SECONDARY-ATTRIBUTE=...,)
```

This relationship is used with modules (type R or L) as the basis for the autolink function (see [5]); reference entries are for example <name><CSECT> and <name><ENTRY>.

PLAM guarantees the consistency of the relationship; the user is responsible for the generation and integrity of the relationships.

Dependencies on member type

LMS has statements

– that are independent of the member type (no relation to the member contents) and

– others that are dependent on the member type.
In the latter case, only a few standard types are permitted and must be used to derive any user-defined types desired. Members of other types cannot be processed.

The following table shows which member types can be used in the various LMS statements and what type checks LMS performs during the LMS run:

Only if the conditions specified in the Type check column are true, will the relevant statement be executed.

	Member type		
Statement	**Source**	**Target**	**Type check**
SHOW-ELEMENT	alphanum-name1..8	-	
SHOW-ELEMENT-ATTRIBUTES	alphanum-name1..8	-	
DELETE-ELEMENT	alphanum-name1..8	-	
MODIFY-ELEMENT-PROTECTION	alphanum-name1..8	-	
FIND-ELEMENT	alphanum-name1..8	-	
MODIFY-ELEMENT-ATTRIBUTES	alphanum-name1..8	alphanum-name1..8	BT(q)=:=BT(z)
COPY-ELEMENT	alphanum-name1..8	alphanum-name1..8	BT(q)=:=BT(z)
PROVIDE-ELEMENT	alphanum-name1..8	alphanum-name1..8	BT(q)=:=BT(z)
RETURN-ELEMENT	alphanum-name1..8	alphanum-name1..8	BT(q)=:=BT(z)
COMPARE-ELEMENT	alphanum-name1..8	alphanum-name1..8	BT(p)=:=BT(s)
EDIT-ELEMENT	S, M, P, D, J, X, derived type	S, M, P, D, J, X, derived type	
MODIFY-ELEMENT	R, C, L S, M, P, D, J, X, derived type	R, C, L S, M, P, D, J, X, derived type	BT(q)=:=BT(z) BT(q)=:=BT(z)
MODIFY-LOGGING-PARAMETERS	-	S, M, P, D, J, X, derived type	

Statement	Member type		
	Source	**Target**	**Type check**
ADD-ELEMENT	“text“ (I)SAM-f “blocks“ PAM file “module“ file,*OMF “phase“ PAM file	S, M, P, D, J, X, derived type X derived type R C	
EXTRACT-ELEMENT	S, M, P, D, J, X, derived type X derived type R C	(I)SAM file PAM file (I)SAM file PAM file	

BT	Base type (name of type of the highest node)
BT(q)/BT(z)	Base type of source/ target member
BT(p)/BT(s)	Base type of the primary or secondary member
Standardtyp	One character in length or with $ or SYS at the beginning
Derived type	Derived from the counted types; alphanum 2..8 without $ or SYS at the start
=:=	The types to the left and right of this sign are either identical or both are text types. For type X, text type means that only text members may be involved.

4.6 Version management

The version of a member is defined in the member designation and identifies the current state of the member.

The following section describes the possible version designations, provides information on version maintenance and storage, and deals with version conventions, which is relevant to automatic version incrementation.

4.6.1 Version maintenance and storage

It is a characteristic of software members that they can be easily modified, and experience shows that they frequently are modified. Stable and relevant states of a development object are therefore maintained in the form of members.

Several versions per member type and member name

In libraries, a member is uniquely defined by its type, name and version designation. Furthermore it is possible to store several versions under one member type and member name.

If the user does not specify the version to be processed, LMS takes the following actions as a standard procedure:

– In read mode
that member is sought whose specified name is accompanied by the highest version designation. The date is ignored.

– In write mode
the actions depend on the statement:

  – ADD-ELEMENT and MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=
The member is generated or overwritten with the highest version X'FF'. LMS identifies this version by @.

  – Other statements
The output member is given the version designation of the input member.

  If an identically named member is overwritten, the internal variant number is incremented by 1. This serves as a write counter.

Different states of a development object are stored in different members. Initially, only the user is aware of the relationships between individual members; they are not established in the library. Each member is an independent unit in the context of a library.

Delta storage

Members that have been created through modification from a predecessor member generally differ only minimally from the predecessor. It is therefore sensible to store this set of members in a more compact form, namely by physically storing only the differences from the relevant predecessor. This compact storage of the differences produces the delta tree.

All members of a delta tree which have no successor are referred to as “leaves”. Only leaves can be overwritten.

Delta trees

Only through the use of delta trees can a relationship between the members also be recorded in the library. The following applies to such members

– on the member designation level

  Since the new member was created through modification from another member, it still belongs to the same class and reflects the same logical content. Both aspects are recorded in the type and name of the member, i.e. all members of a delta tree have the same type and name.
  The version component of a member designation should reflect the state.

– on the physical level

  The relationship between the two members is also to be recorded on the physical level. Which members are to be interlinked is not determined on the basis of the version string, however, but is to be derived from user specifications. For this purpose the user must declare one member as the “predecessor” and one member as the “successor”.

4.6.2 Version designations

The member versions which are to be processed by LMS are identified in the LMS statements by means of the VERSION and BASE operands. Distinctions are made between the source version, the target version and the base version of the target version.

Source version

If a member is used as the *input* for a function, e.g. if it is to be copied or changed, the member version is then called the *source* version.

The source version can be specified as follows:

– composed-name
The version specified by composed-name is the source version.

– *UPPER-LIMIT
This entry selects the highest possible version (represented internally by X´FF´).

– *HIGHEST-EXISTING,BASE=*STD
This entry selects the highest existing version of the specified member name.

– *HIGHEST-EXISTING,BASE=composed-name*
This entry selects the highest existing version with the prefix “composed-name”.

Target version

If the member is used as the *result* of a function, e.g. if it is be written back, the member version is then called the *target* version.

The target version can be specified as follows:

– *BY-SOURCE
The source version is also the target version. If *BY-SOURCE is entered and the source is not a library member, *UPPER-LIMIT will be assumed for the target version.

– composed-name
The version specified by composed-name is the target version. If ’@’ is entered, it will be rejected.

– *UPPER-LIMIT
This entry selects the highest version (represented internally by X´FF´) as the target version.

– *INCREMENT
Specifies a base version according to the convention applicable to the type and the entry made for BASE. An incrementing procedure stipulated by the convention is then applied to the specified base version to determine the target version.

For the first version of a member, no base version can yet exist. In this case, the default version dependent on the convention applicable to the type is the target version.

Convention	Default version
NONE	001
STD-SEQUENCE	EXAMPLE
STD-TREE	001.001
MULTI-SEQUENCE	EXAMPLE

– *HIGHEST-EXISTING
Specifies a base version according to the convention applicable to the type and the entry made for BASE. This base version is also the target version.

For the first version of a member, no base version can yet exist. In this case, the default version dependent on the convention applicable to the type is the target version.

Base version for the target version

Except for the first version of a name, a target version is always generated for each base version. The base version fulfils several functions:

1. If you wish to produce a version with STORAGE-FORM=*DELTA, the base version determines which version the delta forms.
2. If automatic version incrementation (VERSION =*HIGHEST-EXISTING or *INCREMENT) is used to determine the target version, it takes the base version as the point of departure for calculating the target version.
3. In order to generate a version in a scope with active WRITE-CONTROL, a user must also hold the base version.

The base version for the target version can be specified as follows:

*STD

Depending on the VERSION entry, the existing member versions and the convention applicable to the type, a base version is specified as the default.

composed-name

The base version is specified by composed-name.

composed-name*

The base version is the highest existing version with the prefix composed-name.

Although the BASE= operand in statements has the data type <composed-name with-wildcards>, no wildcard characters are permitted in the BASE= entry except for an asterisk (*) at the end to designate the entry as a prefix.

Summary

The following table provides an overview of how the target version and the base version for the target version are determined.

BASE	VERSION	Target version	Base version
*STD	composed-name	composed-name	default base
	*INCREMENT	*INC (default base) or default version	default base
	*HIGHEST-EXISTING	default base or default version	default base
composed-name1*	composed-name2	composed-name2	highest existing with prefix composed-name1
	*INCREMENT	*INC (base version)	highest existing with prefix composed-name1
	*HIGHEST-EXISTING	highest existing with prefix composed-name1	highest existing with prefix composed-name1
composed-name1	composed-name2	composed-name2	composed-name1
	*INCREMENT	*INC (composed-name1)	composed-name1
	*HIGHEST-EXISTING	composed-name1	composed-name1

4.6.3 Version conventions

Each member version is generated in accordance with a specific convention. Each of the conventions is valid for a certain member type within a library.

All member types comply with one of the following conventions:

NONE	No restrictions
STD-SEQUENCE	Only one main line
STD-TREE	Main line with ramifications
MULTI-SEQUENCE	Each prefix forms a line

The convention is set by means of the CONVENTION operand of the MODIFY-TYPE-ATTRIBUTES statement.

It is always possible to set the CONVENTION operand to NONE, but it is not permissible to switch between either of the other values as long as related members types exist.

The LMS statement SHOW-TYPE-ATTRIBUTES shows which convention is set.

This section deals with the following aspects of each convention:

Default version

is the first version of a name generated through automatic version incrementation.

Format

specifies the version-designation format permitted by the convention.

Incrementation procedure

describes the convention’s procedure for determining the target version when VERSION=*INCREMENT.

Standard base selection

describes the convention’s procedure for determining the base version of the target version when BASE=*STD.

Checks

describes the convention’s checks as a prerequisite for statement execution.

4.6.3.1 Convention: NONE

Default version: 001

Format: any

Incrementation procedure

The concluding digit group of a version designation is incremented by 1 (if this is not possible, an error message is output).

Standard base selection

1. When the target version is specified explicitly (composed-name / *UPPER-LIMIT):
   a) if the target version exists, it will also be used as the base version;
   b) if the target version does not exist, the highest existing version will be used as the base version.
2. When the target version is not specified explicitly (*HIGHEST-EXISTING / *INCREMENT), the highest existing version will be used as the base version.

Checks: none

4.6.3.2 Convention: STD-SEQUENCE

Default version: determined by EXAMPLE of MODIFY-TYPE-ATTRIBUTES

Format: [<prefix>]<digit group>

The prefix is a string of any permissible characters ending with a digit. The prefix is the same for all the versions of a type.

Incrementation procedure

The concluding digit group of a version designation is incremented by 1 (if this is not possible, an error message is output).

Standard base selection

The highest existing version is used as the base version.

Checks

The target version must be at the same level or higher than the highest existing version.

4.6.3.3 Convention: STD-TREE

Default version: 001.001

Format: R.L[.B.S] R,L,B,S = nnn n = digit

R.L (Release.Level) are versions of the main branch.
R.L.B.S (Release.Level.Branch.Sequence) are versions of side branches.
Leading zeros in the individual digit groups may be omitted if

- no wildcard characters are specified in the version designation and
- the member type is specified explicitly.

Example '1.1' may be entered for '001.001'

Incrementation procedure

The concluding digit group of a version designation is incremented by 1, provided the base version is a leaf of a branch. Otherwise, a new, higher branch with sequence 1 is produced.

Standard base selection

1. When the target version is specified explicitly (composed-name / *UPPER-LIMIT):
   a) if the target version exists, it will also be used as the base version;
   b) If lower versions than the target version exist on the same branch, the highest of those versions is used as the base version.
   c) If a main branch version for the target version exists, that main branch version is used as the base version.
   d) If side branch versions for the target version exist, the highest of those versions is used as the base version.
   e) Otherwise, there is no base version.
2. When the target version is not specified explicitly (*HIGHEST-EXISTING / *INCREMENT), the highest existing version will be used as the base version.

Checks

Only the data of the highest version in a branch can be changed. Only higher versions of a given name/type can be added to a branch, or new, or higher side branches added.

4.6.3.4 Convention: MULTI-SEQUENCE

Default version: specified using EXAMPLE in //MODIFY-TYPE-ATTRIBUTES.

Format: [<Prefix>]<digit group>

The prefix is a string of any permissible characters ending with a digit. The prefix is the same for all the versions of a type.

Incrementation procedure

The concluding digit group of a version designation is incremented by 1 (if this is not possible, an error message is output).

Standard base selection

1. Ehen the target version is specified explicitly (composed-name / *UPPER-LIMIT),
   a) the highest version with the same prefix is the base version, insofar as such a version exists.
   b) the highest version of all the existing versions is the base version.
2. When the target version is specified implicitly (*HIGHEST-EXISTING / *INCREMENT),

   The highest version of all the existing versions is the base version.

Checks

Each prefix forms a subname space. The target version must be higher that the highest existing version in the same subname space or must form a new subname space.

Notes

– LMS ensures that the structures of version designations match the delta structure.

– If a main branch version is deleted on which a side branch was dependent, it is no longer possible to copy the entire tree to a type complying with a convention because there is no longer a suitable base for the side branch concerned.

– Unless it is a blank string, composed-name must end with a period (.) when entered as BASE=composed-name*. This means that the base is the highest version of a main or side branch.

4.7 Member protection/data protection

This section describes the functions used by LMS to support member and data protection. LMS can apply protection attributes both to members and to libraries.

An AUDIT, i.e. a log of selected events, can also be generated for security documentation (see [6]).

4.7.1 Access protection for members

Since they are contained in libraries, members enjoy the same degree of protection as the libraries themselves. It is also possible to give members added protection by making access rights explicit, instead of implicit.

Member access rights

r : read

w : write

x : execute

h : hold

a : administer

The rights r, w, x and h are set for individual members and modified by means of the LMS statement MODIFY-ELEMENT-PROTECTION (see page 314). To display existing authorization settings, use the SHOW-ELEMENT-ATTRIBUTES statement (see page 406).

Administer authorization is set for all the members in a given library with the LMS statement MODIFY-LIBRARY-ATTRIBUTES (see page 324) and for all the members of a given type with the statement MODIFY-TYPE-ATTRIBUTES (see page 356). To display existing administer authorization settings, use the statement SHOW-LIBRARY-ATTRIBUTES (see page 425) or SHOW-TYPE-ATTRIBUTES (see page 438).
Administer authorization is used to specify the person(s) authorized to create, delete and rename members within a library or type.

The default setting for administer authorization is NONE, which allows all administration functions that are allowed by the protection of the library file.

Protection attributes can be modified only by the owner of the library.

For each right, one of the following protective mechanisms can be set:

NONE: No special protection

STD: Standard protection (BACL) [+ password]

BY-GUARD: Protection through GUARDS

Password protection is linked to BACL protection and so plays no role in GUARD protection.

Only one mechanism can be active for a given right at any point in time, but different mechanisms can be set for different rights. If the mechanism set for a right is changed, the values of the previously active mechanism are lost.

If it is unclear on the basis of the active mechanism and the current system environment whether an access right exists, no access is permitted.

This may occur for the following reasons:

protection by GUARDS, but the GUARDS subsystem is not installed

Possible remedies are:

change the system environment
change the active protection mechanism (option available to library owners only)

The various mechanisms implement protection as follows:

1. NONE - no special protection

   - no access check

2. STD - standard protection by BACL

   Each of the access rights listed above is assigned by means of protection bits and possibly by means of passwords to the following groups of users:

   OWNER: Owners of the library file

   GROUP: Group of the owner of the library file

   OTHERS: All others

   The group of selected users can be restricted further by means of passwords. Passwords are stored in encrypted form in the library. Access to a member protected by a password is permitted only if the password entered matches the password stored earlier in the password table (using the BS2000 command ADD-PASSWORD).

   The groups of authorized users and the passwords are defined by means of the USER and PASSWORD operands, respectively, in the LMS statement MODIFY-ELEMENT-PROTECTION.

3. BY-GUARD - protection by GUARDS

   Access is determined on the basis of a protection description stored in a guard.

   A guard is an independent object of BS2000, which can be created, modified and deleted using BS2000 commands. The owner of the guard defines both the protection description and the group of users who may use the guard:

   ```
   SCOPE = USERID / USER-GROUP / HOST-SYSTEM
   ```

   (see [6])

   The protection description contains conditions which must be satisfied before access is granted.

   These conditions may include:

   Date
   Time
   Weekday
   Task privilege
   Program loaded

   Access is granted only if the library owner is (still) permitted to use the guard and the conditions specified in GUARDS have been satisfied.

   The guard which is to be used to check access requests is specified by the library owner.

   Only GUARD names which were accepted by the GUARDS subsystem may be used. No check is made to determine whether or not the specified guard even exists. Permissible GUARD names are stored in the library exactly as they were specified.

   All GUARD names refer to the catalog ID (CATID) under which the library containing the GUARD name is cataloged.

   GUARD names without USERID refer to the user ID (USERID) under which the library containing the GUARD name is cataloged.

   GUARD names are displayed to everyone who is permitted to read the contents of the library.

   When using GUARDS protection for libraries and members, it is necessary to bear in mind certain special aspects, for example that it makes sense to protect individual members of a library only if the library itself is also protected, i.e. is in “PROTECTED MODE”.

Example

The library LIBR, which contains the member MEMB1, is available under the user ID USER1. USER2 would like to access MEMB1. USER1 has the guard GUARD1, which allows access to USER1 and USER2. There are three cases to consider:

1. Only the library LIBR is read-protected by GUARD1

   In this case, LIBR is not handled as if it were a file. It is recognized that LIBR is a library, and the access protection is different than it would be if LIBR were a normal file. The library is in PROTECTED MODE, so that access is possible only via LMS or via the COPY-FILE command. A further attempt by USER1 or USER2 to gain read access (e.g. SHOW-FILE LIBR) will be rejected. It is possible, however, to access individual members, for example with SHOW-FILE *L(LIBR,MEMB1,S).
   If one or more members of LIBR are protected with GUARDS, copying is no longer possible (see 3 below).

2. Only the members are protected by GUARD1.

   This method of protection makes no sense because the library must then have USER-ACC=*ALL-USERS in order for USER2 to be able to see the library. In this case, USER2 can copy the entire library and so gain access to all its members, regardless of what protection is specified for the individual members.

3. The library and its individual members are protected by GUARDS.

   This method provides protection for the individual members of a library. All the users who are to be allowed access to members must be granted corresponding access rights to the library. In this case, however, this means only that those users can see the catalog entry and access it with LMS.
   Even then it makes sense to protect individual members with GUARDS. The actual access rights are determined by the lesser of the two access rights, i.e. if a user has no write access to the entire library, he cannot write a member even if a guard at member level would allow him to do so.

If libraries are protected with GUARDS but no further protection is defined for the individual members, users can work with the members in almost the same way as with files.

Initial member protection

The owner of the library file can define initial member protection for the library and/or a specific member type.

If initial member protection was defined, the protection is entered for new members. If no initial member protection was defined, members are created without additional protection and are protected only by the library file protection.

If initial member protection is defined both for the library and for the relevant member type, only the protection defined for the type is taken into account. If the initial member protection is changed, the change affects only the protection of members created after that time, i.e. the protection of existing members remains unchanged.

Overview of protection attributes

Library level:			
a:	NONE	/ BACL [+ password]	/ BY-GUARD
Initial member protection			
r:	NONE	/ BACL [+ password]	/ BY-GUARD
w:	NONE	/ BACL [+ password]	/ BY-GUARD
x:	NONE	/ BACL [+ password]	/ BY-GUARD
h:	NONE	/ BACL [+ password]	/ BY-GUARD
Type level:			
a:	NONE	/ BACL [+ password]	/ BY-GUARD
Initial member protection			
r:	NONE	/ BACL [+ password]	/ BY-GUARD
w:	NONE	/ BACL [+ password]	/ BY-GUARD
x:	NONE	/ BACL [+ password]	/ BY-GUARD
h:	NONE	/ BACL [+ password]	/ BY-GUARD
Member level:			
Member protection			
r:	NONE	/ BACL [+ password]	/ BY-GUARD
w:	NONE	/ BACL [+ password]	/ BY-GUARD
x:	NONE	/ BACL [+ password]	/ BY-GUARD
h:	NONE	/ BACL [+ password]	/ BY-GUARD

Overview of rights required for LMS actions

The following overview shows which rights are required in order to perform specific LMS actions, and what conditions must have been satisfied. It is assumed that the library file can be opened in a suitable way.

The abbreviations used in the overview have the following meanings:

-	Action is either not permitted or not possible
*	Action may be executed by anyone
,	AND operator
/	OR operator
EATTR =	(CCSN, USER-DATE/TIME)
STATE	Member state set with MODIFY-ELEMENT-ATTRIBUTES.
WRITE-CONTROL	Value of WRITE-CONTROL in effect in the member scope (see MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES)
a:	The caller must have administer authorization for the member scope. The caller can gain administer authorization in one of the following ways: 1. Administer authorization for the relevant member type has been granted, and the caller belongs to the group of authorized users. 2. Administer authorization has been granted not for the member type, but for the library, and the caller belongs to the group of authorized users. 3. Neither for the member type, nor for the library has administer authorization been granted:
r:	The caller must have read authorization for the member.
w:	The caller must have write authorization for the member.
x:	The caller must have execute authorization for the member.
h:	The caller must have hold authorization for the member.
E:	The caller must be the owner of the library.
H:	The caller must be entered as the HOLDER of the member.
B:	Explicitly or implicitly specified base version.

LMS action	**Required conditions**			
WRITE-CONTROL=	DEACTIVATED		ACTIVATED	
Modify LIBRARY-ATTR	E		E	
Modify TYPE-ATTR	E		E	
Create 1st version	a		a	
Create nth version	a		H(B) [1]	
STATE=	FREE	IN-HOLD	FREE	IN-HOLD
Show ELEMENT-ATTR	*	*	*	*
Delete version	a,w	-	a,w	-
Rename version[2]	a,w	-	-	-
Overwrite version	w	w,H	-	w,H
Modify EATTR	a/w	-	a/w	-
Set STATE=IN-HOLD	h	-	h	-
Set STATE=FREE	h	H/E	h	H/E
Read version	r	r	r	r
Execute version	x	x	x	x
Modify ELEMENT-PROT	E	E	E	E

[1] The new version is created with the attributes of the base version, i.e. with STATE=*IN-HOLD.

[2] If a member is to be renamed with the name of another existing member, that member must be authorized to overwrite.

Notes

– In systems without a group structure, no access is possible through the user circle “group”. A setting analogous to USER-ACCESS is, however, possible through the user circles “owner” and “all others”.

– Member protection is meaningful only when the library file is protected against UPAM accesses. Library files are protected against UPAM accesses if they are protected by means of a Basic Access Control List (BACL) or GUARDS and have been properly set up.

  Libraries input by a file transfer operation (openFT) are not considered to have been properly installed until they have been accessed for writing by PLAM.

- For libraries that are protected against UPAM accesses, the following restrictions apply:
  - they can no longer be included in a library by ADD-ELEMENT
  - they can no longer be processed via remote file access (RFA).

  For such applications, the BACL protection must be deactivated for the library file. The library file is then no longer protected against UPAM accesses. It can now be processed like any other PAM file.
- Libraries which contain protection attributes and are protected against UPAM access can only be copied by their owner using COPY-FILE.
- GUARD protection is possible, provided, of course, that the GUARDS subsystem is available.

4.7.2 Data protection by overwriting

Data protection by overwriting means that the user deletes files that are no longer required by specificallyoverwriting them. The data is physically deleted by this operation, i.e. overwritten with X'00'. Overwriting of data is controlled locally by the LMS statement DELETE-ELEMENT or globally in MODIFY-LMS-DEFAULTS, by the DESTROY-DATA operand in each case.

The DESTROY-DATA operand is on the one hand a member attribute, i.e. overwriting automatically acts on this member, and on the other hand a processing parameter of the LMS statement DELETE-ELEMENT. As a processing parameter, DESTROY-DATA causes all members covered by the statement to be overwritten on deletion.

The data is overwritten with X'00' if one of the following specifications calls for overwriting:

- class 2 option DESTLEV
- specification for member: value of DESTROY-DATA on last creation or on last write access to the member
- specification via the DESTROY-DATA operand

4.7.3 Auditing

The access method PLAM used by LMS has an interface with the subsystem SAT (security audit trail) in the security package SECOS (see [6]). When SAT is active, the following events can be selected by the security administrator for logging:

– CREATE ELEMENT
– MODIFY ELEMENT
– READ ELEMENT
– EXECUTE ELEMENT
– CLOSE ELEMENT
– DELETE ELEMENT
– RENAME ELEMENT
– CREATE-SECURITY-ATTRIBUTES
– MODIFY-SECURITY-ATTRIBUTES
– DELETE-SECURITY-ATTRIBUTES

5 LMS functions

This chapter gives an overview of the LMS functions.

5.1 Starting/terminating LMS

5.1.1 Starting LMS

LMS is called as an autonomous program by the /START-LMS or /LMS command.

Area of application: **UTILITIES**

START-LMS
VERSION = ***STD** **/** <product-version> **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-REST** / <integer 1..32767>

VERSION =
Specifies the desired product version.

VERSION = *STD
The version defined by the SELECT-PRODUCT-VERSION command is selected. If no version has been defined, the system selects the highest possible version.

VERSION = <product-version>
Entry of the selected version.

MONJV = *NONE / <filename 1..54 without-gen-vers>
Name of the job variable that is to monitor the LMS run. (Facility available only to users of the software product JV (see [8].)

During LMS execution, the system sets the job variables to the following values:

Value	Meaning/Reason for value assignment
$R	LMS running.
$T	LMS terminated normally.
$A	LMS terminated abnormally.

MONJV = *NONE
No job variable is defined.

CPU-LIMIT = *JOB-REST / <integer 1..32767>
The maximum amount of CPU time (in seconds) which LMS may use for execution. If this time is exceeded in interactive mode, the user is informed by the system; in batch mode, the LMS run is terminated.

CPU-LIMIT = *JOB-REST
If the interactive job was started with a time limitation, the value defined at system generation is used as the time limit for the LMS run. Otherwise, there is no time limit for the program.

Command return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	2	LMS0238	Error while loading LMS
	32	LMS1002	Internal errror
	130	LMS0041	System address space exhausted

Note

If a start file is specified, the command return code can be overwritten by the return codes entered in the statements in the start file.

5.1.2 Monitoring LMS execution with job variables

If LMS is started with `/START-LMS MONJV=<name>` containing a monitoring job variable, LMS places a value in the variable when it terminates. MONJV is supplied with a value independent of the value of the SDF-P /BEGIN-BLOCK command PROPAGATE-STMT-RC operand.

The MONJV value is composed of a three-byte status indicator and a four-byte return code indicator. The return code indicator is composed of a one-byte termination code (TC) and a three-byte program information indicator (PI). LMS sets the status indicator and the termination code as follows:

Status indicator

Status indicator			**Return code**				**Remarks**
			Termination code (TC)	**Program information (PI)**			
1	2	3	4	5	6	7	Byte
$T			0	See below			normal termination
			1				
$A			2				abnormal termination
			3				

Termination code

BC	Meaning
0	Normal termination LMS executed without error
1	Normal termination Warnings were issued or errors occurred which were less serious than the value set as MAX-ERROR-WEIGHT (see MODIFY-LMS-DEFAULTS).
2	Abnormal termination A criterion for abortion as defined with MAX-ERROR-WEIGHT was met (see MODIFY-LMS-DEFAULTS).
3	Abnormal termination The error which occurred was so serious that continuing the LMS run was either not possible or was not worth while. The LMS run was terminated internally.

Program information

PI	Meaning
000	TC: 0 LMS executed without error
001	TC: 1 Nothing worse than warnings occurred.
002	TC: 1 or 2 Nothing worse than RECOVERABLE-class errors occurred, e.g. it was not possible to find or to overwrite a member.
003	TC: 1 or 2 Nothing worse than SIGNIFICANT-class errors occurred, i.e. no major errors.
004	TC: 2 Nothing worse than SERIOUS-class errors occurred.
005	TC: 3 The error which occurred was so serious that continuing the LMS run was either not possible or was not worth while. The LMS run was terminated internally.

Note

If the END statement is used in conjunction with the LMS functionality from EDT, the considerations on page 121 should be borne in mind.

5.1.3 Start file

Users have the option of creating a SAM or ISAM file which will be processed automatically when LMS is started. The file may contain any LMS statements desired. The LMS statements which are to be executed automatically when LMS is started may also be stored in a library member.

SYSDTA must have the value PRIMARY or SYSCMD when LMS is started or the sequence of start statements will not be executed. After the start file has been processed, SYSDTA is reset to its original value.

LMS searches for the start file as follows:

1. via the S variable SYSLMSPAR
2. via the link name $LMSPAR
3. under the file name SYSPAR.LMS
4. via the global start file

The system administrator can select any installation location for the global start file. The logical identification SYSPAR is defined for this global start file in the SYSSII file.

The system administrator informs the installation monitor of the installation file name using the SET-INSTALLATION-PATH command. If no installation file name is defined for the global start file, a global start file is sought via the file name $.SYSPAR.LMS.

The names $LMSPAR, SYSLMSPAR, SYSPAR.LMS and the global start file must not be used except in conjunction with the start file. Using any of these names for other purposes, for example SYSPAR.LMS as the name of a PLAM library, may lead to errors when LMS starts. Following output of an error message, the LMS run is continued.

The S variable SYSLMSPAR must contain a valid value of the TO-FILE= operand of the ASSIGN-SYSDTA command (see the ASSIGN-SYSDTA command in [4]). In other words, the variable may contain the name of a file or a library member in the form *LIB-ELEM(LIB=...,ELEM=...,TYPE=...).

The S variable must be declared with SCOPE=TASK if it is to be set in a procedure (e.g. the LOGON procedure) yet still be accessible outside the procedure for other LMS runs. Otherwise, the variable need not be declared.

The start file can be switched off with /ADD-FILE-LINK FILE-NAME=*DUMMY,LINK-NAME=$LMSPAR, provided the sequence of start statements is not referenced using the variable SYSLMSPAR. If the variable exists but should not cause the sequence of start statements to be executed, the variable must contain the string *DUMMY.

Examples

– The values LOGGING=*MAXIMUM and TYPE=S are to be preset in the SYSPAR.LMS file.

  The start file contains the statements:

```
//MODIFY-LOGGING-PARAMETERS LOGGING=*MAXIMUM
//MODIFY-LMS-DEFAULTS TYPE=S
```

  Following the `/START-LMS` call, the above values are set automatically.

– This time, the start statements are to be stored in the START member under type S in the library X.

  The S variable SYSLMSPAR contains:

```
/SYSLMSPAR='*LIB-ELEM(LIB=X,ELEM=START,TYPE=S)'
```

  The START member contains the statements:

```
//MODIFY-LOGGING-PARAMETERS LOGGING=*MAXIMUM
//MODIFY-LMS-DEFAULTS TYPE=S
```

  Following the `/START-LMS` call, the above values are set automatically.

5.1.4 Preset options following LMS startup

Default values come into effect following LMS startup. If values other than these are required for certain statements, the values can be modified by means of the LMS statement MODIFY-LMS-DEFAULTS or can be changed locally.

The following table indicates which statements are affected by the default values, or whether the default values influence the entire LMS run.

If positive acknowledgments are to be logged in addition to error messages, the operand LOGGING=*MAXIMUM must be set in the MODIFY-LOGGING-PARAMETERS statement.

5.1.5 Terminating the LMS run

The statement END must be specified to terminate LMS .

END

The END statement has no operands (for further details on the END statement, see page 256).

5.2 Library assignment

Libraries must first be assigned before they can be processed. Only when assignment has been successfully performed can members be added and/or processed.

In LMS statements, libraries are specified by means of the LIBRARY operand. There are several ways of assigning a library, all of which ultimately lead directly or indirectly to PLAM libraries:

1. Via a globally defined library (*STD)
2. Using the direct name of a PLAM library
3. Indirectly via a link name
4. Using a library list ("SYSPLAMALT-" variable)
5. By means of type redirection ("SYSPLAMLIB." variable)

The globally defined library is assigned by means of OPEN-LIBRARY and reset to undefined by means of CLOSE-LIBRARY. Numbers 2 - 5 of the above options can be used to define the global library.

Library lists enable alternate libraries to be specified. A library list is a list showing which libraries are to be searched for a member and in which order. Library lists can only be accessed in read mode, and are defined by means of a "SYSPLAMALT-" variable (ALT stands for ALTernate library).

The type redirection mechanism allows a standard member type to be redirected to a user-defined member type. This enables existing programs which do not allow user-defined types to nonetheless function with these types. A type redirection is defined by means of a "SYSPLAMLIB." variable. Type redirection is a central mechanism which can be used in all commands, independently of LMS.

The LMS statement SHOW-LIBRARY-STATUS provides information on the status of the libraries to be processed.

Direct name of a PLAM library

The specified file name designates a PLAM library. This specification is often used during work with LMS. Normally the specified file name does in fact indicate a file, but there are the following exceptions:

– if the file name begins with "SYSPLAMALT-" and an S variable of the same name exists: this indicates a library list (see page 77)
– if the file name begins with "SYSPLAMLIB." and an S structure variable of the same name exists: this indicates a type redirection (see page 80)

Indirectly via a link name

The name <link> specified under *LINK designates a link name. As a rule, the name is a file link name defined via /ADD-FILE-LINK, to which a PLAM library is assigned. However, the following exception applies:

if an S structure variable SYSPLAMLIB.<link> exists: this indicates a type redirection (see below).

Examples

1. Assigning a global library with OPEN-LIBRARY

   A library is opened as a **global** library if it is assigned through the LMS statement OPEN-LIBRARY.

   Only one global library is possible per LMS run. If a new global library is opened by a second OPEN-LIBRARY statement, the first global library is closed by an internal LMS CLOSE and the new library is considered to be the global library.

   A global library may already exist or can be newly created. If a library is newly created, it must be generated with MODE=*UPDATE.

   A global library is normally opened only for reading. If it is to be opened for reading and writing, the operand MODE=*UPDATE must be set in the OPEN-LIBRARY statement.

   In the LMS statements a global library is addressed by LIBRARY=*STD. This value is the default in these statements so no other explicit library specification is required.

```
/START-LMS
//WRITE-COMMENT 'Open a global library for reading and writing'
//OPEN-LIBRARY LIBRARY = global-lib, MODE = *UPDATE
 .
 .
//WRITE-COMMENT 'Display directory of the global library'
//SHOW-ELEMENT-ATTRIBUTES
 .
 .
```

2. Assigning a global library via its link name

   A global library can also be assigned by means of its link name. However, when using the link name, a /ADD-FILE-LINK command must be issued before calling LMS in order to establish the link with the file name of the library.

```
/ADD-FILE-LINK FILE-NAME = global-lib, LINK-NAME = testlib
/START-LMS
//WRITE-COMMENT 'Open a global library via link name'
//OPEN-LIBRARY LIBRARY = *LINK(LINK-NAME = testlib)
 .
 .
```

3. Assigning local libraries in a statement

   By specifying one or more libraries in the data structure *LIBRARY-ELEMENT of the LIBRARY operand, the libraries are defined locally within the statement. The definition of the libraries is valid only for that statement.

```
/START-LMS
 //WRITE-COMMENT 'Example of assignment of a local library '
 //WRITE-COMMENT ' in the Statement EDIT-ELEMENT '
  .
 //MODIFY-LMS-DEFAULTS TYPE=S
 //EDIT-ELEMENT LIBRARY-ELEMENT(LIBRARY=local-lib,ELEMENT=test)
  .
  .
```

4. Assigning a local library via its link name

   A local library can also be assigned by means of its link name. However, when using the link name, a /ADD-FILE-LINK command must be issued before calling LMS in order to establish the link with the file name of the library.

```
/ADD-FILE-LINK FILE-NAME = local-lib, LINK-NAME = testlib
/START-LMS                                        .
//WRITE-COMMENT 'Open a local library via link name'
//WRITE-COMMENT ' in the statement EDIT-ELEMENT '
 .
//EDIT-ELEMENT (LIBRARY = *LINK(LINK-NAME = testlib), ELEMENT=test)
 .
 .
```

Assigning a library from a list

Library lists enable alternate libraries to be specified. The order of the alternate libraries in the list determines the order in which they are searched for members.

When a member is sought in a library list, PLAM processes the list to first of all find a member with the desired type and name. The libraries in the list are searched in the order in which they appear until the specified member name with the associated type is found. This means that if more than one library in the list contains members with the same type and name, the member is always found in the first of these libraries; this library is called the hit library.

The version specification for the member determines which member version is selected from the hit library. So one member version in a library covers all member versions in libraries named later, and the selection process is easy to follow (see example on page 78)

Library lists can only be accessed in read mode. A library list is defined by means of an S variable. The variable has the type *string* and the name begins with 'SYSPLAMALT-' (ALT stands for ALTernate library). The contents of the variable are set as follows:

SYSPLAMALT-<name> = '(<lib>,<lib>,...)'

<name>:	Rest of the variable name
<lib>:	Library specification

If the name of the library list is specified as a file name ":catid:$userid.SYSPLAMALT-", catalog ID and user ID are ignored. Names beginning with 'SYS' are reserved for future development. Lowercase letters in the contents of the "SYSPLAMALT-" variable are interpreted as uppercase letters.

Library lists have one level, which means that specifications in the library list are always interpreted as PLAM libraries or type redirections. All libraries in the list must be present.

Note

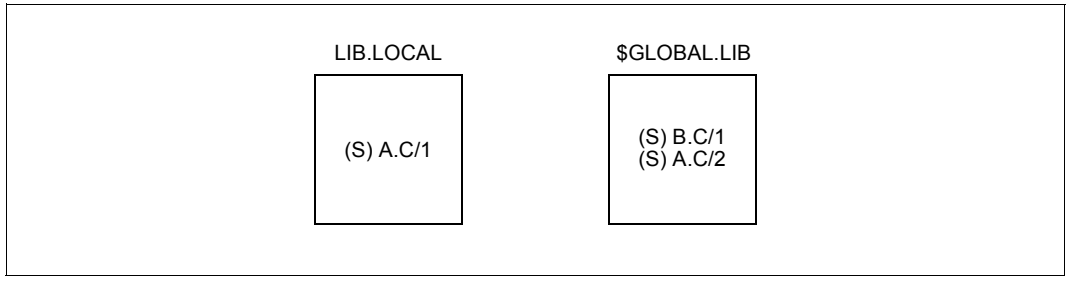
Library lists can only be accessed in read mode, i.e. if the LIBRARY operand specifies an output library, no library list is permitted. Wildcards are not allowed in library lists.

Library lists are only supported by other products if this is explicitly described in the corresponding manual. Use of library lists with any other products can result in unpredictable behavior.

Example

Library lists allow convenient use of the make functionality for projects in which sources are administered with the borrowing mechanism. The sources are located in one of the libraries LIB.LOCAL or $GLOBAL.LIB. The borrowing mechanism is used to copy to and fro between these two libraries. In the make file, SYSPLAMALT-HUGO is specified directly as the only source library. The contents of SYSPLAMALT-HUGO determine which libraries are to be searched.

The contents of the libraries are as follows:



To combine the libraries LIB.LOCAL and $GLOBAL.LIB for the local make run:

```
/set-variable sysplamalt-hugo = '(lib.local,$global.lib)'
```

A.C from LIB.LOCAL, and B.C from $GLOBAL.LIB are searched.

With a central make run, SYSPLAMALT-HUGO only receives one library:

```
/set-variable sysplamalt-hugo = '($global.lib)'
```

Only sources A.C and B.C from $GLOBAL.LIB which have already been returned are searched. The statements for make are the same in both cases:

```
//modify-make-defaults library=sysplamalt-hugo
```

Validity of global and local libraries

A global library remains valid for the entire LMS run until either a new global library is opened or the library is closed by an explicit CLOSE-LIBRARY statement.

If a local library is also opened, then this is only valid within this one statement. On termination of this statement, the global library is valid once again.

The LMS statement SHOW-LIBRARY-STATUS (see page 428) provides information on the status of the libraries.

Working with more than one library

Only one global library can be assigned at a time. It if necessary to work simultaneously with two libraries, the second library must be locally defined.

If more than one library is assigned globally in the LMS run (by several OPEN-LIBRARY statements), LMS uses the most recently specified library.

Example

Member xyz, which is contained in library lib1, is copied to the library lib2 which already exists. lib1 is to be used as the global library and lib2 as the local library. Since the member type is not yet defined after calling LMS, it is defined as global member type S. The member is not to be renamed:

```
/START-LMS
 .
 .
//MODIFY-LMS-DEFAULTS TYPE=S
//OPEN-LIBRARY LIBRARY = lib1
//COPY-ELEMENT ELEM=*LIB-ELEM(,xyz), TO-ELEM=*LIB-ELEM=(LIBRARY=lib2)
 .
 .
```

Redirection mechanism

In addition to the standard member types, LMS allows users to work with member types which they have defined themselves. These user-defined types should be derived from a standard type and have one or more apt names (e.g. COBSRC, ASSSRC, etc.) defined as a synonym for the single-character designation of the standard type.
Programs which have not been prepared for user-defined types can nonetheless function with these members. PLAM offers a redirection mechanism that makes it possible to access libraries containing user-defined member types. All commands and programs in BS2000/OSD can use type redirection.

The redirection mechanism is implemented by means of the S variable SYSPLAMLIB, which is a *structure*-type variable. Its structure members specify what things are to be redirected and the destinations to which they are to be redirected.

The name of the structure member specifies what is to be redirected,
while the content of the structure member specifies the destination of the redirection.

The name of the structure member is composed of the link name and the user-defined member type.

The contents of the structure member consist of:
– the library name and the member type or
– the link name and the member type.

If no member type is specified, the member type indicated by the name of the structure member is used.

The S variable SYSPLAMLIB, which is written in the form of a variable assignment, has the following structure:

```
SYSPLAMLIB.<link>.<utype>=´<libname>(TYPE=<typename> / *SAME)´
```

or

```
SYSPLAMLIB.<link>.<utype>=´*LINK(LINK=<linkname>,TYPE=<typename> / *SAME)´
```

`link`	Link name in the form <structured-name1..8>
`utype`	User-defined member type in the form <alphanum-name1..8> Generally, a user-defined member type is a synonym for an existing standard type.
`typename`	Member type in the form <alphanum-name1..8> If the member type is not specified, `typename=utype` applies.
`libname`	Library name
`linkname`	Link name in the form <structured-name1..8>
`*SAME`	The member type of the structure member contents is the same as the member type in the name of the structure member.

The content of the S variable is a string containing a library component and a type component which mirrors the real library environment. TYPE=*SAME in the variable content means <utype>. The library component must not be a library list.

Access via the S variable SYSPLAMLIB can be effected in two ways:

– specification of a library via the file name SYSPLAMLIB.<link>

– specification via the link name <link>

If SYSPLAMLIB is declared as a variable with SCOPE=VISIBLE or =TASK, the contents of the structure member are evaluated and redirected into the relevant library.

Example

The object of this example is to compile COBOL sources of the member type COBSRC. They are located in the library MY.COBLIB, which is specified by the link name SRCLIB. The copy members should be of the COBCOPY type and also located in the MY.COBLIB library.

```
/DECLARE-VARIABLE SYSPLAMLIB(TYPE=*STRUCTURE),SCOPE=*TASK ————————————————————  (1)
/SET-VARIABLE SYSPLAMLIB.SRCLIB.S = ´MY.COBLIB(COBSRC)´ ————————————————————  (2)
/SET-VARIABLE SYSPLAMLIB.COBLIB.S = ´MY.COBLIB(COBCOPY)´ ————————————————————  (3)
/ADD-FILE-LINK FILE-NAME=SYSPLAMLIB.SRCLIB,LINK=SRCLIB ————————————————————  (4)
/ADD-FILE-LINK FILE-NAME=SYSPLAMLIB.COBLIB,LINK=COBLIB
/MODIFY-JOB-SWITCHES ON=1
/ASSIGN-SYSDTA *SYSCMD
/ASSIGN-SYSLST #L
/START-EXECUTABLE-PROGRAM COBOL85
COMOPT SOURCE-ELEMENT=LMSCOBS
COMOPT SOURCE-VERSION=399
COMOPT MODULE=MY.COBLIB
COMOPT SYSLST=(DIAG,MAP,SOURCE)
END
/STEP
/MODIFY-JOB-SWITCHES OFF=1
/EXIT-PROC
```

(1) Declares the central S variables (with SCOPE=TASK).

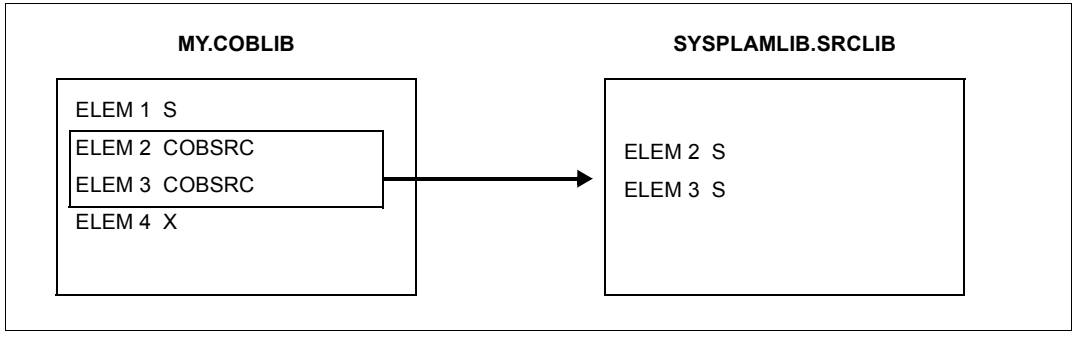
(2) Sets the redirection;
redirection is effected via the link name SRCLIB or the file name SYSPLAMLIB.SRCLIB.

For example, when the Cobol compiler accesses the type S, the access is mapped PLAM-internally to the type COBSRC.

(3) The copy members are mapped on the COBCOPY type as described at (2).

(4) Couples the link name SRCLIB with the real library SYSPLAMLIB.SRCLIB. This is necessary in the event that programs run a distributed check for the existence of the library.

After SYSPLAMLIB.SRCLIB.S = ´MY.COBLIB(COBSRC)´ the following situation exists:



Only COBSRC-type members can be accessed, and not all of those present in the library can be accessed. The members are accessed via member type S, and type S is shown instead of the “real” type COBSRC in program outputs.

The library MYCOPY contains members with the member type COBCOPY, which are to be designated with the member type S in an LMS run. To circumvent LMS checks, the library SYSPLAMLIB.COBLIB must exist.

```
/DECLARE-VARIABLE SYSPLAMLIB(TYPE=STRUCTURE)
/SET-VARIABLE SYSPLAMLIB.COBLIB.S = ´MYCOPY(COBCOPY)´
/ADD-FILE-LINK SYSPLAMLIB.COBLIB,LINK=COBLIB
/START-LMS
//SHOW-ELEM-ATTR *LIB(SYSPLAMLIB.COBLIB)
   Displays all members of the type COBCOPY from MYCOPY. Type S is logged.
//SHOW-ELEM-ATTR *LIB(SYSPLAMLIB.COBLIB,*ALL(*ALL),COBCOPY)
   No hits (the type COBCOPY does not exist in the user view).
//SHOW-ELEM-ATTR *LIB(SYSPLAMLIB.COBLIB,*ALL(*ALL),S)
   Displays all members of the type COBCOPY from MYCOPY.
//SHOW-ELEM-ATTR (*LINK(COBLIB))
   Displays all members of the type COBCOPY from MYCOPY. Type S is logged.
//END
```

5.3 Processing of members

The following sections provide an overview of the possible ways in which members can be processed with LMS.

LMS permits members to be

- entered in libraries as non-delta and delta members
- output to files
- output to other libraries (copied)
- listed
- deleted
- compared
- renamed
- edited
- corrected
- and can output the library's directory.

All of the LMS statements mentioned in this section are described on page 156ff.

5.3.1 Adding members to a library

The following statements add members to the assigned library:

ADD-ELEMENT, COPY-ELEMENT and MODIFY-LOGGING-PARAMETERS.

The WRITE-MODE operand determines whether or not an identically named member in the output library is overwritten.

ADD-ELEMENT

The ADD-ELEMENT statement (see page 165) adds files, modules from the EAM area and records from the LMS statement stream to the assigned library as members. If no library is specified, the library opened by OPEN-LIBRARY is used.

This statement enables the user to additionally define whether the member is stored as a non-delta member or as a delta member.

The FIXED and UNDEFINED record formats are converted into the VARIABLE record format; i.e. given a 4-byte record header. Libraries permit files with a RECORD-SIZE of up to 32 Kbytes (including the record header) to be stored.

If an ISAM file is added, the SOURCE-ATTRIBUTES operand determines whether the file attributes, the ISAM key and information on ISAM secondary keys are included.

ISAM keys having a length of up to 255 bytes may then be stored.

Members having ISAM keys are suitable only for archiving (see Note).

If the operand SOURCE-ATTRIBUTES=*KEEP is set, it is also possible to include files with RECORD-FORMAT=*FIXED; if not, only RECORD-FORMAT=*VARIABLE is allowed.

Notes

– The ISAM keys of a source program file should not be included in the member, since the compiler cannot translate the source program from this member without errors if ISAM keys are present.

– If system file SYSDTA is assigned to a member which has stored the ISAM key, the ISAM keys are also read. The ISAM keys must then be removed from the program which carries out the processing.

Files can be stored under the following member types:

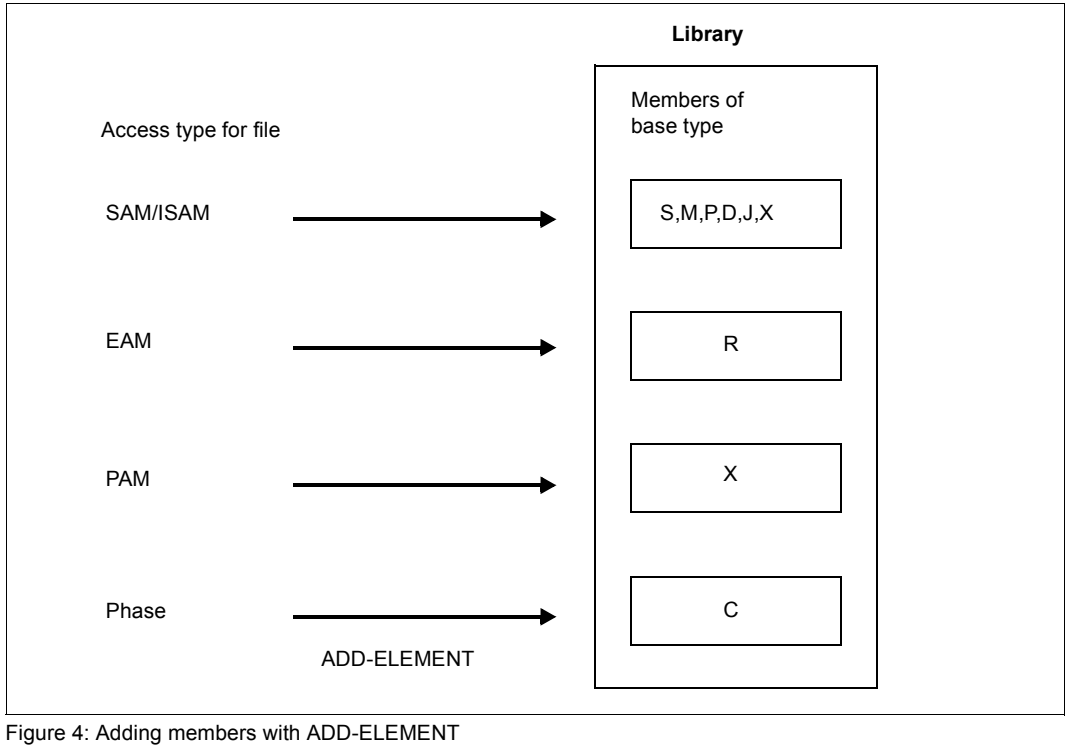


Figure 4: Adding members with ADD-ELEMENT

COPY-ELEMENT

The COPY-ELEMENT statement (see page 213) copies members from the input library to the output library, storing them there with different member designations, if desired. This statement enables the user to additionally define whether the member is stored as a non-delta member or as a delta member.

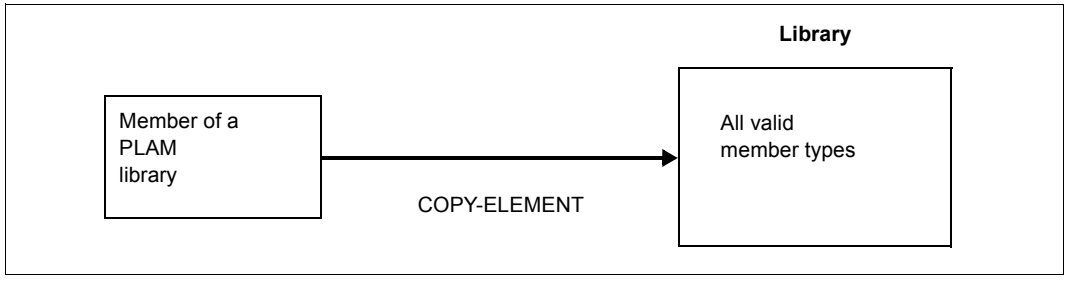


Figure 5: Adding members with COPY-ELEMENT

MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=*LIBRARY-ELEMENT

The statement MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=*LIBRARY-ELEMENT (see page 351) includes the LMS log (see page 97) in the member specified by *LIBRARY-ELEMENT.

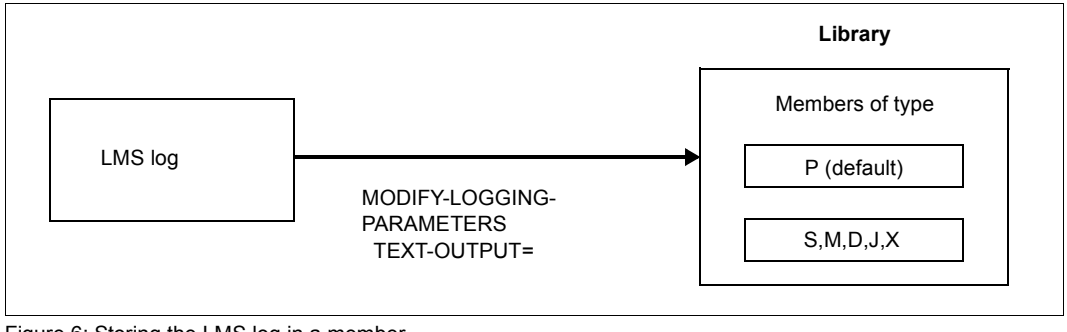


Figure 6: Storing the LMS log in a member

5.3.2 Outputting members to a file

The members of a library are output to a file by means of the EXTRACT-ELEMENT statement (see page 257).

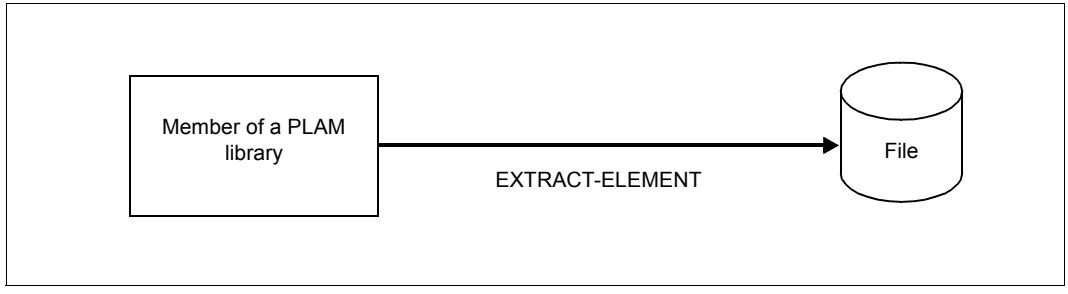


Figure 7: Outputting members

5.3.3 Listing members

The listing of members is controlled by the SHOW-ELEMENT statement (see page 392); it is possible to define the format in which the members are to be output and how much information is to be displayed.

5.3.4 Deleting members

The DELETE-ELEMENT statement (see page 230) deletes members in the assigned library.

A distinction is made between logical and physical deletion:

– Logical deletion
  The entries in the directory are deleted and storage space for the member concerned is released.

– Physical deletion
  In addition to logical deletion, the storage space of the corresponding member is overwritten with binary zeros.

A member of a library is physically deleted if the operand DESTROY-DATA=*YES has been set, if the member contains a code indicating physical deletion or if the class 2 option DESTLEV calls for physical deletion.

Delta members are not deleted physically until the last delta member of a delta tree, i.e. the complete delta tree, has been deleted.

5.3.5 Comparing members

The COMPARE-ELEMENT statement (see page 201) is used to compare members. The members are compared one record at a time, where the range of the comparison operation can be defined with the RECORD-PART operand. Comparison statistics are always produced.

The COMPARE-ELEMENT statement is also executed even if only one of the comparison members is found in the specified libraries. This allows the counting of records in members.

If two members are compared with one another, LMS uses the terms primary member and secondary member. The user is free to select the new or the old member as the base member. LMS always considers the secondary member to be the base for the comparison. This means that LMS identifies missing records in the secondary member as inserted records and missing records in the primary member as deleted records.

The differences established can be logged if requested (INFORMATION operand). This log is referred to as a comparison log. Following the comparison log, the SHOW-STATISTICS statement can be used to output the comparison statistics in the form of a table showing the results of the comparison in terms of numbers.

When deciding whether two records match, a distinction is made between formal and logical comparison.

In formal comparisons all record characters are compared, while in logical comparisons blanks are *ignored*.

The results of the two comparison modes (formal and logical) are logged in the same way.

Comparison log

Normally, the compared parts of the records are logged and not the complete records.

In the comparison log the results are based on the comparison of two ranges, the secondary member being taken as the point of reference.

Comparison result	Meaning
SAM (same)	The compared parts of the two records in the primary and secondary members match.
DEL (deleted)	The record with this comparison range only occurs in the secondary member.
INS (inserted)	The record with this comparison range only occurs in the primary member.

Comparison statistics

The comparison statistics are displayed by means of the SHOW-STATISTICS statement (see page 434).

The comparison statistics supply the following information about a comparison:

– total number of records compared in primary and secondary members
– number of records inserted
– number of records deleted
– number of identical records.

In addition the result of the entire comparison is indicated:

S (same)	No differences were found during the comparison.
C (changed)	Differences were found during the comparison.
I (inserted)	The secondary member was not found.
D (deleted)	The primary member was not found.
ERR (error)	An error occurred during the comparison.

5.3.6 Correcting members

LMS provides the following two statements for correcting members; the type of the member(s) requiring correction determines which statement should be used:

– EDIT-ELEMENT corrects text members with EDT
(member types S, M, J, P, D, X or types derived from them)

– MODIFY-ELEMENT corrects object modules, link and load modules, phases and text members via substatements
(member types R, L, C)
(member types S, M, J, P, D, X or types derived from them)

Correcting with EDIT-ELEMENT

The EDIT-ELEMENT statement (see page 237) calls the editor EDT as a subroutine. The specified member is then processed using EDT statements. On termination of EDT, the corrected member is written to the output library. It may contain a new member designation.

Correcting with MODIFY-ELEMENT

The MODIFY-ELEMENT statement (see page 277) corrects object modules, link and load modules, phases and text members. Correction of these members is controlled by various substatements. These are read from the statement stream immediately following MODIFY-ELEMENT up to the END-MODIFY substatement.

The corrected member is then written back to the assigned library. It may receive a new member designation.

For types R, L and C, you may use the following functions:

– correct text records
– cancel corrections
– delete record types from the input member

For type R only, you may also use the following functions:

– generate REP records
– modify control section attributes
– rename symbols

For text members, you may use the following functions:

– insert records
– delete records

5.3.7 Renaming members

The MODIFY-ELEMENT-ATTRIBUTES statement (see page 304) renames the specified members of the assigned library. This statement also permits the renaming of members whose designations do not conform to LMS conventions.

Renaming of delta members is not permitted for audit reasons.

5.3.8 Outputting library directories

The SHOW-ELEMENT-ATTRIBUTES statement (see page 406) logs the directory entries of the specified members or of the entire library.

The directory is always output sorted on the member type. The remainder of the sort sequence is determined by the SORT operand. Unless otherwise specified, member designations are output sorted by type, name, version and date.

To obtain the complete directory of a library, all you have to do is enter the SHOW-ELEMENT-ATTRIBUTES statement without further operands, provided no individual member type was specified using the MODIFY-LMS-DEFAULTS statement.

5.3.9 Storing procedures

LMS allows the user to store BS2000 procedures as members in libraries (member type J).

Existing procedure files can be incorporated as members into libraries by means of ADD-ELEMENT.

Storing procedures in this way, especially where small command files are concerned, saves storage space. The number of catalog entries is decreased.

Note, however, that any ISAM keys that have been stored (see page 83) must be removed from the members before the procedure is called.

A library member can also be assigned as the system input file (SYSDTA) by means of the BS2000 command ASSIGN-SYSDTA (see [4]).

5.4 Archiving members using the delta method

Two methods are available for storing multiple versions of one member name in libraries:

– the non-delta storage method

– the delta storage method

The **non-delta storage method** is used to store exactly one member, i.e. all records of a member, in its own container (a unit of storage in the library). If another version is added under this name, all records of this member will also be kept in an individual container. Any relationship that may exist between these members is unknown to LMS.

Such members are henceforth referred to as non-delta members.

During processing, e.g. reading a non-delta member, LMS can directly access all records of the member specified and perform the action. This method applies to all member types and, because of its fast access features, it is particularly suitable for members that are still being developed or subject to change.

For the text-oriented member types S, M, J, P, D and X or types derived from them, the **delta storage method** can be used to store multiple versions of one member. With this storage-saving method only the temporarily first member is stored in its entirety in its own container. When other versions of the same member are added, then only the records that are different from those of the previous member are identified and inserted (comparison of members). In addition, the link between new version and previous version reveals the logical relationship between the individual members.

Such members are henceforth referred to as delta members.

During processing, e.g. reading a delta member, the relevant records of the referenced member are filtered out. This may slow down the entire action if a great number of related delta members are present. The delta storage method is therefore specially suited for the efficient and transparent filing of member versions.

5.4.1 Delta as a storage form and organizational aid

Delta members can be distinguished from non-delta members not only on account of their efficient storage form but because of the unique relationship that exists among delta members.

Storage space is saved due to the delta structure:

– Redundant records of a member with respect to the predecessor member can be identified and will not be stored again.

– For records to be identified as redundant a formal comparison is made between the records of the new version and the specified base version.

Unique relationships are established via:

– Unique member names
  Delta members that are interrelated have the same member name and thus form a range of names. For this reason, non-delta members and delta members must have different member names, which means that

  – a non-delta member is created only if no delta member exists that has the same name,

  – a delta member is created only if no non-delta member exists that has the same name.

  All delta members having the same name form exactly one logically structured “delta tree” which in its simplest form is a “delta sequence”.

  One name is associated with exactly one delta tree.

– Unique version designations
  When a member is included as a delta member, It is possible to specify which member is the predecessor member, i.e. which member is to be used as the basis for comparison. The default is BASE=*STD.

The conventions for member names and version designations that are established when members are added cannot be subsequently altered, for reasons of auditing and consistency. A new delta tree, if required, may be created by copying and simultaneously renaming the existing delta tree.

5.4.2 Adding delta members

Delta members are included in the assigned library by means of the operand STORAGE-FORM=*DELTA in the ADD-ELEMENT, COPY-ELEMENT and EDIT-ELEMENT statements.

If LMS is to define the storage form of the new member, STORAGE-FORM=*STD must be specified. Depending on whether the new name exists, LMS determines whether the new member is to be included as a non-delta member or a delta member. If the name for the new member to be added already exists as a delta, then a new delta member will be created; otherwise a non-delta member is created.

Adding a member to a delta sequence

A delta sequence is a linear delta tree, i.e. a delta tree without branches.

Thus, for example, a member is included in the delta sequence by specifying VERSION = *INCREMENT and BASE = *STD:
The predecessor is a delta member having the same name and the highest version designation. The new delta member is appended to the currently highest version. The delta quantity is established and stored in the container. The version designation of the new delta member should be higher than the currently highest designation.

If no predecessor exists, the member is created as the first member of a delta sequence.

Example of a delta sequence (the arrows represent the relationships)

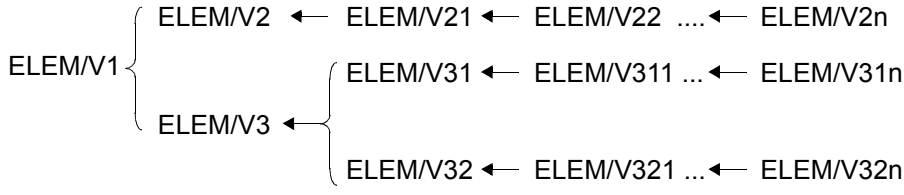
ELEM/V1 <-- ELEM/V2 <-- ELEM/V3 ..<-- ELEM/Vn

Adding a member to a delta tree

A delta tree can be built by specifying VERSION = <text 1..24> and BASE = <predecessor>:

The predecessor is the delta member having the same name and the specified version designation. The new delta member can be appended laterally to the specified version. The relation to the predecessor member is to be obtained from the version designation of the new delta member.

Example of a delta tree (the arrows represent the relationships)



5.4.3 Overview of delta members

An overview of the existing delta members in a library can be output by means of the operand INFORMATION = DELTA-STRUCTURE in the SHOW-ELEMENT-ATTRIBUTES statement. A complete delta tree will then always be listed, regardless of which member has been specified.

As well as the member designations the internal delta numbers of the members are output to field DELTA# and the internal numbers of the predecessor members are output to field BASE#.

The delta number reflects the chronological order in which the members have been included. Both numbers uniquely describe the chaining of members in a tree; they cannot be controlled by the user. The user can visibly define the chaining of members by proper version specification.

5.4.4 Deleting delta members

A distinction should be made between logical and physical deletion:

– Logical deletion

  This merely deletes the entry from the directory. The records and the relationship with the predecessor member remain intact.

– Physical deletion

  Controlled by the DESTROY-DATA operand (it must be specified when the member is included) the storage space allocated to the member is overwritten with binary zeros. When used for delta members this operand will only become active once the last member of the delta tree is deleted. Specification of DESTROY-DATA=*YES for a member suffices to delete the entire container.

5.4.5 Locking delta members

When processing a **non-delta member**, the specified member will be locked (member = container).

When processing a **delta member**, all members stored in the referenced container will be locked (delta tree = container).

5.4.6 Restrictions when using the delta method

Here are some details that should be considered when archiving members by means of the delta method:

– **Renaming** single delta members or an entire delta tree is not possible as this would disrupt the auditability of an archive or put the consistency of data at risk if a current action is somehow aborted.

– **Overwriting** delta members is permitted only if the delta members are leaves of a delta tree, i.e. have no successor.

5.5 Controlling the LMS run

The following section describes the facilities provided by LMS for controlling the LMS run. These include logging parameters, user interfaces, job switches, etc.

5.5.1 LMS logging parameters

LMS logging parameters are global parameters. LMS has the following logging parameters:

Logging parameter	Meaning of parameter
LOGGING	Scope of message logging
OUTPUT	Medium for log output
OUTPUT-LAYOUT	Scope and layout of log output

All logging parameters have as their default value the keyword *UNCHANGED, i.e. the current setting is not changed. At the beginning of the LMS run, the logging parameters have the values immediately following *UNCHANGED but these values can be modified with the MODIFY-LOGGING-PARAMETERS statement.

5.5.2 Controlling log output

The LMS log contains everything output by LMS such as for example the result of the statements, their execution or abnormal termination, the assigned I/O libraries as well as lists generated, for example, when members are listed or compared.

The log may be written to the system file SYSOUT, SYSLST, to a library member or to work file 9 of EDT. The output medium is specified by means of the TEXT-OUTPUT operand of the MODIFY-LOGGING-PARAMETERS statement.

If the log is written to a member, LMS normally creates a P-type member.

If job switch 4 was set when LMS was invoked, the start and end messages of LMS are suppressed.
Error messages are always output.

The following tables shows which operands in which statements control output of the log:

Statement	Operand	Function
MODIFY-LOGGING-PARAMETERS	LOGGING	Specifies whether or not positive acknowledgments are to be logged
	TEXT-OUTPUT	Specifies the output medium for the log
	OUTPUT-LAYOUT	Specifies the output format of the log
	LINES-PER-PAGE	Specifies the number of lines on a log page
	LINE-SIZE	Specifies the length of the lines
	EXTRA-FORM-FEED	Generates a form feed signal when a change of member occurs
	HEADER-LINES	Specifies whether or not headers are to be output
SHOW-ELEMENT	OUTPUT-FORM	Specifies the format for a member display
	TEXT-/ MODULE-/ PHASE-/ LLM-INFORMATION	Specifies the scope of information output in a member display
SHOW-ELEMENT-ATTRIBUTES	SORT	Specifies how the directory is to be sorted
	LAYOUT	Specifies the format for the directory log
	INFORMATION	Specifies the scope of directory logging
	TEXT-OUTPUT	Controls the log output
COMPARE-ELEMENT	LAYOUT	Specifies the format for the comparison log
	INFORMATION	Specifies the scope of comparison logging
	TEXT-OUTPUT	Controls the log output

Positive and negative acknowledgments

If the operand LOGGING=*MAXIMUM is set in the LMS statement MODIFY-LOGGING-PARAMETERS, the execution of each LMS statement affecting a member will be logged. If the statement is executed successfully, LMS will issue a positive acknowledgment.

If the statement cannot be executed, LMS will log a negative acknowledgment and, if applicable, a corresponding error message.

You can find the messages on the manual server (URL: *http://manuals.ts.fujitsu.com*) by means of an HTML application and on the “BS2000/OSD SoftBooks” DVD.

All positive and negative acknowledgments have the following format:

```
[NO] statement member[word member][cause]
```

`NO`	The statement has not been executed.
`statement`	Statement name.
`member`	Member designation or file name (in ADD-ELEMENT and EXTRACT-ELEMENT).
`word`	Keyword: AS, INTO, WITH.
`cause`	Result: EXISTING, REPLACED, etc.

5.5.3 Controlling screen overflow

LMS does not itself perform any screen overflow control. The system handles this function. LMS outputs can therefore only be aborted if the program interrupt key (K2) key is pressed and a SEND-MSG command (see page 337) is subsequently entered and sent.

5.5.4 Error handling in interactive and procedure modes

LMS differentiates between the interactive mode and the procedure mode of execution.

– Interactive mode

  In interactive mode, following output of the error message, the prompt // appears, requesting entry of the next statement.

– Procedure mode

  LMS supports two error handling mechanisms in procedure mode. A spin-off mechanism and a statement return code mechanism. The setting of the PROPAGATE-STMT-RC operand in the /BEGIN-BLOCK command determines which of the two is effective.

5.5.4.1 Spin-off mechanism

The spin-off mechanism is effective for LMS statements by default, i.e. a jump is made to the next //STEP- or //END statement when an error occurs. If the jump is to //END, LMS terminates with TERM UNIT=STEP, MODE=ABNORMAL. The spin-off is then propagated to the next command. The user can decide which errors cause LMS to trigger the spin-off mechanism. It controls this via the MAX-ERROR-WEIGHT operand (see //MODIFY-LMS-DEFAULTS).

Note

If an error occurs during substatement processing which causes the main application to terminate, a main statement is always expected after a specified //STEP statement.

5.5.4.2 Statement return code mechanism

LMS supports statement return codes analogous to the command return codes (see [4]). The statement return code allows the user to react as necessary to specific conditions after each LMS statement.

Structure of the statement return codes

The statement return code has three parts:

Maincode: Message code; the meaning can be retrieved with /HELP-MSG-INFORMATION.

Subcode1: Error class (decimal); indicates the gravity of the error.

Subcode2: Supplementary information (decimal); e.g. subcode2 = 2 in conjunction with subcode1 = 0 indicates a warning.

Controlling the output of statement return codes

The output of statement return codes is activated with the command:

```
/BEGIN-BLOCK PROGRAM-INPUT=*MIXED-WITH-CMD( -
/                          PROPAGATE-STMT-RC=*TO-CMD-RC )
```

The statement return codes can then be evaluated in S procedures with SDF-P means (see the SDF-P manual "Programming in the Command Language" [12]). The SDF-P built-in functions MAINCODE() or MC(), SUBCODE1() or SC1() and SUBCODE2() or SC2() can be used for evaluation.

Error handling at the command level is triggered when statement return codes are output, i.e. a jump to the next /IF-CMD-ERROR, /IF-BLOCK-ERROR or /STEP command.

The error handling is triggered if subcode1 of the statement return code is not equal to 0. This is the case with all error messages. The setting of MAX-ERROR-WEIGHT has no effect.

The statement return code can be saved with the /SAVE-RETURNCODE command. /IF-CMD-ERROR implicitly executes the /SAVE-RETURNCODE command.

Note

With maincodes LMS1002, LMS1003 and LMS1004, the true error must be taken from the LMS protocol, if this is present. Maincode LMS1003 after a statement with a wildcard entry may mean that several errors have occurred.

The following possible statement return codes are defined for LMS.

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
1	0	LMS0036	Library not assigned
2	0	LMS0053	Member and file attributes different
2	0	LMS0064	GCCSN macro error; no CCS name specified
2	0	LMS0071	XHCS not loaded
2	0	LMS0084	VTSUCB macro error
2	0	LMS0095	Input records missing
2	0	LMS0102	Incomplete module in EAM file
2	0	LMS0129	Statement aborted by user
2	0	LMS0151	Input or output medium set to standard
2	0	LMS0163	At least one record truncated
2	0	LMS0199	Record length invalid with fixed record format
2	0	LMS0201	Only the comparison area is logged
2	0	LMS0274	Block control value changed
2	0	LMS0286	File attributes not modified
2	0	LMS0712	Touch not possible
2	0	LMS0714	Touch not possible on empty file
2	0	LMS0721	The specified target is already current
	2	CMD0230	Syntax error
	32	LMS0238	Error while loading LMS
	64	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0035	Member protection not transferrable to file
	64	LMS0093	Protocol member already exists
	64	LMS0211	Library already exists
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0301	File not found
	64	LMS0302	Member not found
	64	LMS0303	Member not in the range of the reference condition
	64	LMS0304	Type not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
		LMS1003	Error during batch (wildcard) processing with at least one member or file

(SC2)	SC1	Maincode	Meaning
	64	LMS1004	Other error
	64	PLA0223	Only the leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrowing status prevents member access
	64	PLA0475	Function violates the version automation
	64	PLA0476	Version does not comply with the valid convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0081	No further storage space for SYSLST file
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

(SC2) : Subcode2 = 0 is represented by blanks

The possible return codes are shown again in the descriptions of the separate LMS statements.

Example

Search for member A in LIB1 or LIB2 :

```
/BEGIN-BLOCK PROGRAM-INPUT=*MIXED-WITH-CMD( -
/            PROPAGATE-STMT-RC=*TO-CMD-RC )
/START-LMS
//SHOW-ELEM-ATTR  *LIB(LIB = LIB1, ELEM = A, TYP = S),TEXT-OUT=*NONE
/IF-CMD-ERROR
//  SHOW-ELEM-ATTR  *LIB(LIB = LIB2, ELEM = A, TYP = S),TEXT-OUT=*NONE
/   IF-CMD-ERROR
/      SET-VAR LIB = 'none'
/   ELSE
/      SET-VAR LIB = 'LIB2'
/   END-IF
/ELSE
/   SET-VAR LIB = 'LIB1'
/END-IF
/WR-TEXT 'Member A is in the &LIB library'
//END
/END-BLOCK
```

5.5.5 User interfaces

LMS enables the user to branch to a user program during the listing or comparison of members.

This subroutine can perform the following actions prior to the processing of a member record:

– manipulate the current member record

– insert own records before the current member record, or at the end of the member

– exclude the current member record from processing

For details, see the LMS statement ACTIVATE-USER-EXIT, page 158ff.

5.5.6 Interrupting the LMS run

The LMS run can be interrupted by the user or by the program.

User interrupt

The user can interrupt the LMS run by pressing a program interrupt key (e.g. K2).

Continuation of the LMS run can be controlled by the /SEND-MSG command, which may optionally be supplied with an input text. This input text is subsequently interpreted by LMS during interrupt handling. The current function is informed of the type of termination and terminates in the desired way. The possible inputs are described under the DIALOG-CONTROL operand of the MODIFY-LMS-DEFAULTS statement (see page 330).

Further program systems, such as EDT, can be invoked under LMS. These systems may have their own STXIT routines. LMS always sets up its own STXIT management block.

If different systems use different STXIT routines, then the associated management blocks in those systems are linked with those of LMS, or vice versa.

When an event occurs, all the STXIT routines associated with this event are activated, i.e. in addition to the LMS routines the routines of other systems are also executed. This means that the LMS routines are activated even if the interrupt event does not occur in LMS.

A good example of this is the SEND-MSG command. The SEND-MSG command can be used to send messages to LMS and to all other STXIT routines that handle this event.

Note

The SEND-MSG command is handled by all associated STXIT routines of the entire program system, i.e. it can lead to problems if individual program elements react differently to the same message. The program is generally continued in the subsystem in which the event occurred.

LMS interrupt caused by errors

Error handling is also controlled by means of the STXIT routine.

In the case of program termination, line loss, or specification of /START-EXECUTABLE-PROGRAM, /LOAD-EXECUTABLE-PROGRAM, /CANCEL-JOB, /LOGOFF, /CANCEL-PROGRAM, /ABEND, /EXIT-JOB, a check is performed to ensure that the libraries remain consistent.

The following applies to all program termination conditions:

– All STXIT routines in LMS are deactivated in order to prevent any incorrect continuation of processing by SEND-MSG.
– LMS simulates an END. This causes all open libraries to be closed.

If any libraries are still open at program termination time, these will be closed.

5.5.7 Using job switches

The user can influence the LMS run by means of BS2000 job switches. They must be set by the system command `/MODIFY-JOB-SWITCHES ON=(no,...)` before LMS is loaded.

The following job switch affects the LMS run:

Job switch 4:

When job switch 4 is set, the start and end messages for LMS are suppressed.

Job switches are only interrogated on initialization; any subsequent setting and resetting has no effect for LMS.

5.6 PAM key elimination

New disk formats, particularly the non-key (NK) format, have been introduced to increase the net storage capacity and the effective data transfer rate. The NK disk is formatted without PAM key (non-key disk). In order to be able to use the NK disk, the K files must be converted to NK files (PAM key elimination).

Several different disk file formats exist for SAM, ISAM and UPAM files: the previous format tied to the PAM key (K format) and the non-PAM key formats (NK2 and NK4 formats). The file format is defined by the BLKCTRL value. BLKCTRL can assume the value PAMKEY, DATA, DATA2K, DATA4K or NO. For details on the file formats, please refer to [9].

5.6.1 Library files

The distinction between K and NK format is primarily related to DMS. This distinction is reflected in the following ways in the internal file organization of the PLAM library:

The PAM key is not required. With regard to files, however, there is a difference which is represented by the BLKCTRL file attribute.

PLAM libraries need not be converted with PAMCONV when migrating between the K and the NK environments.

5.6.2 Member processing

The following diagram provides an overview of the situations which may arise when transferring data between the file and library members.
For members, logical information units are listed; for files, the BLKCTR value is given.

The arrows indicate the transfer direction.

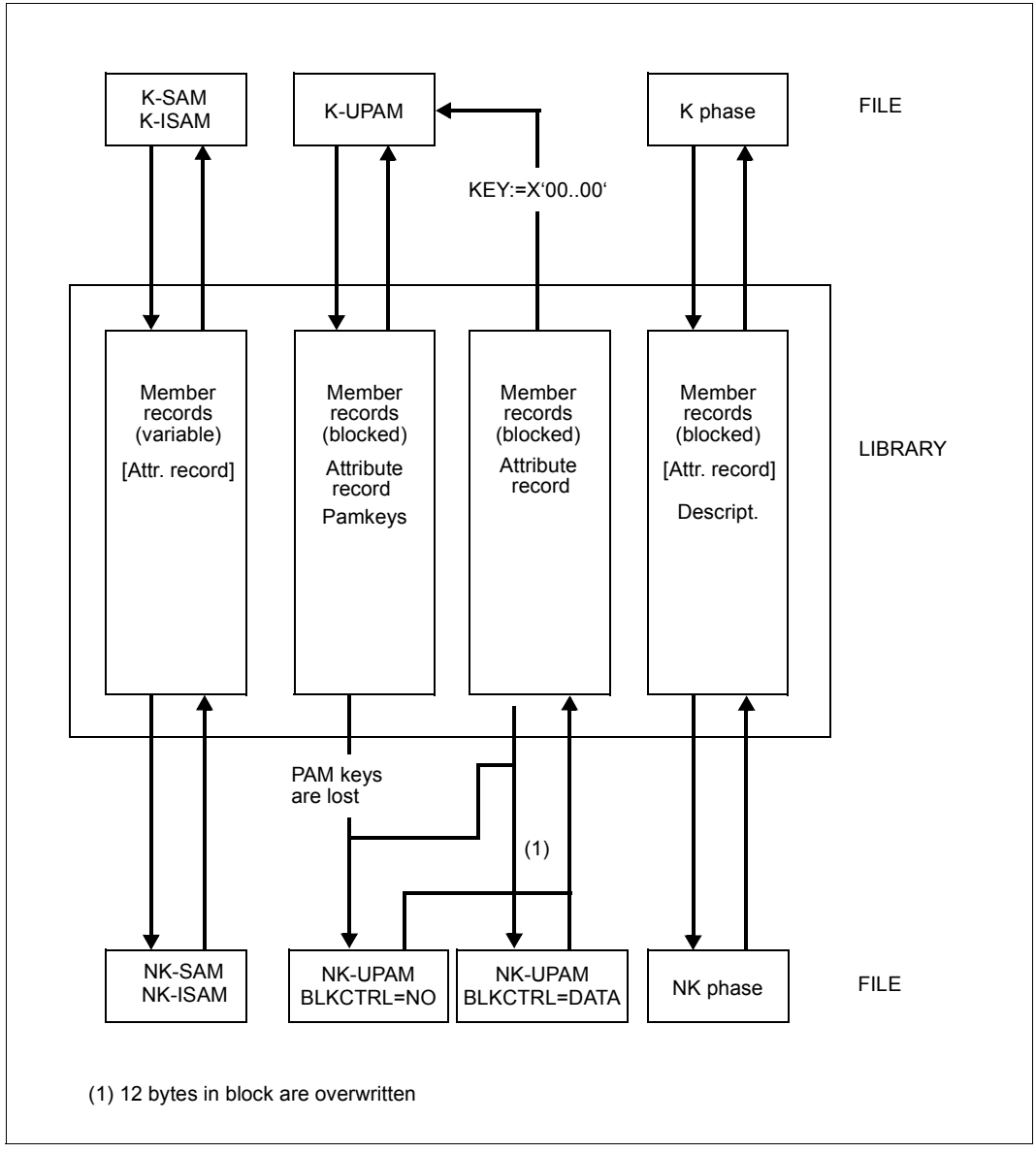


Figure 8: Transfer of information between the file and library members

Use of the ADD-ELEMENT statement

The ADD-ELEMENT statement is used to store file contents in members as follows:

Note the following details:

– SAM/ISAM files:

  When SAM and ISAM files are added, the BLKCTRL value is also stored if SOURCE-ATTRIBUTES=*KEEP has been set, i.e. the original file block structure determined by the BLKCTRL value is documented in the attribute record.

  The individual records are read using the SAM/ISAM logical access method and written unchanged to the member as variable-format records.
  The member structure generated is independent of the original BLKCTRL attribute.

– PAM files

  When PAM files are added, the BLKCTRL value, too, is always stored. The blocks of the file are read using the UPAM access method and stored unchanged as blocks in the member. If PAM keys are specified, i.e. BLKCTRL=PAMKEY, these PAM keys are stored in the member.
  The generated member thus retains the block structure determined by the BLKCTRL value.

– Phases

  When phases are added, the BLKCTRL value is not stored. The corresponding format specification is stored on file in the phase information. In the PLAM library, K phases and NK phases have the same format. The PAM key information is stored in descriptors.

ADD file>member	File type	BLKCTRL entry in attribute record	PAM key storage
File on	SAM/ISAM	— [1])	
NK disk	SAM/ISAM	from the catalog	no
	UPAM	from the catalog	no
File on	SAM/ISAM	— [1])	
PK disk	SAM/ISAM	from the catalog	no
	UPAM	from the catalog	for BLKCTRL=PAMKEY

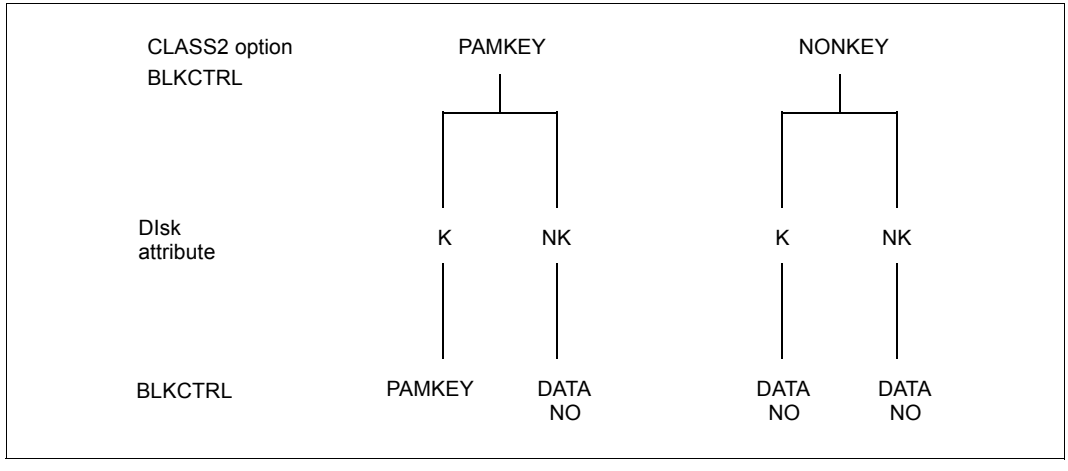
[1]) Storage can be controlled via the SOURCE-ATTRIBUTES operand

Use of the EXTRACT-ELEMENT statement

The EXTRACT-ELEMENT statement is used to output the contents of members to files. The BLKCTRL value is determined via the following hierarchy:

1. The specification in the catalog entry or ADD-FILE-LINK command.
2. BLKCTRL value stored for the member. This is relevant only for files which were originally PAM files.
3. Setting of the CLASS2 option BLKCTRL: PAMKEY or NONKEY. This can be displayed by means of the SHOW-SYSTEM-PARAMETERS command.
4. Disk attribute K or NK

If no catalog entry exists and the BLKCTRL value has not been stored, the class 2 option and the disk attribute determine the BLKCTRL value:



If the class 2 option BLKCTRL has been set to PAMKEY, LMS lets the system define the BLKCTRL value, i.e. BLKCTRL = <not specified>.

If the class 2 option BLKCTRL has been set to NONKEY, LMS sets BLKCTRL = DATA for SAM and ISAM files and BLKCTRL = NO for PAM files.

Note the following details:

– ISAM files

  Variable-length member records are written using the ISAM logical access method. The BLKCTRL value of the file is determined according to the algorithm described above; in this case, however, point 2) above does not apply, as the BLKCTRL value stored for the member is used for documentation purposes only and is ignored.

– SAM files

  If BLKCTRL=DATA is specified, a DMS error occurs if records in the member are longer than 32 Kbytes - 16 bytes. In the K environment, these records may have a length of up to 32 Kbytes - 4 bytes. When selecting records, LMS passes those which are too long to DMS without checking them.

  The BLKCTRL value is determined in the same way as for ISAM files.

– PAM files (type X members)

  In the NK environment, the PAM keys are lost. Phases tied to the PAM key can then no longer be loaded. In addition, when BLKCTRL=DATA is specified, the first 12 bytes of each logical block are overwritten by the system. In both cases LMS issues a warning. In the case of phases and PLAM files, no warning is issued.

– Phases (type C members)

  Phases are handled in a special way.
  In addition to the old phase format (K phase), there is a new PAM-key-free phase format (NK phase) for files.

5.6.3 Summary

– SAM/ISAM files

It is always possible to add and select files. Any BLKCTRL values stored are used for documentation purposes only.

The internal file format is always determined by the SAM/ISAM access method. This method also converts records to the internal block format of the file.

– UPAM files

Neither the UPAM access method nor LMS can be used for the automatic conversion of data, since this would result in a loss of data.

The user has ultimate control.

File type U P A M	**BLKCTRL entry generated / stored in the attribute record**			
	PAMKEY	DATA	NO	—
File located on NK disk	[1] [2]	ADD EXTRACT	ADD EXTRACT	— EXTRACT
File located on K disk	ADD EXTRACT	ADD EXTRACT	ADD EXTRACT	— EXTRACT

[1]) The value BLKCTRL=PAMKEY is not possible.
[2]) The selection process must be controlled by the user, e.g. by specifying a link name in the statement.

5.7 Support for NK4 disks

In BS2000 there are two formats of PLAM libraries supported by LMS:

– the 2K-oriented format (NK2 PLAM file)

– the 4K-oriented format (NK4 PLAM file)

The COPY-LIBRARY statement converts from one format to the other (cf. example under COPY-LIBRARY, page 226). The user determines the appropriate format in the BS2000 command ADD-FILE-LINK and the BUFFER-LENGTH= operand:

```
NK2 PLAM file:  BUFFER-LENGTH=STD(SIZE=1)
NK4 PLAM file:  BUFFER-LENGTH=STD(SIZE=2)
```

LMS supports both library formats. It also supports NK4 disks with the ADD-ELEMENT and EXTRACT-ELEMENT statements.

5.7.1 Adding files with ADD-ELEMENT

Using ADD-ELEMENT, files of any BUFFER-LENGTH can be added to a PLAM library.

5.7.2 Outputting members with EXTRACT-ELEMENT

Here it is necessary to distinguish between members with or without an attribute record:

Member with attribute record

The member has an attribute record with the original BUFFER-LENGTH specification. This occurs if the operand SOURCE-ATTRIBUTES=*KEEP in the ADD-ELEMENT statement was specified or if the member was an original UPAM file, e.g. a PLAM library.

For the output of these members, the following points should be borne in mind:

– A BUFFER-LENGTH value is explicitly preset for the target file, either through an entry in the task file table (TFT) via the ADD-FILE-LINK command, or directly in the catalog entry. In this case, the preset value is always used. When the value is transferred, the following problems may occur:

  – SAM/ISAM file:
    The member records are too long for the preset BUFFER-LENGTH. A DMS error is then output.

  – UPAM file:
    When creating UPAM files, LMS fills up a logical block (except for the last one) with 2K units and only then outputs it with UPAM .

When BLOCK-CONTROL-INFORMATION=WITHIN-DATA-BLOCK, every logical block (BUFFER-LENGTH) begins with a 12-btye block control field (CF). If the preset BUFFER-LENGTH does not correspond to the stored value, data can be overwritten with the CF by the BS2000 Data Management System (DMS). The file is then unusable.

When BLOCK-CONTROL-INFORMATION=NO, however, unusable information may likewise be generated if the BUFFER-LENGTH is changed (e.g. PLAM files).

In order to avoid these situations, LMS issues a warning if the specified BUFFER-LENGTH differs from the stored entry. However, LMS always attempts to generate the file.

– A BUFFER-LENGTH value is not explicitly preset or known for the target file. In this case, the value from the attribute record is used.

  If the value n in STD(SIZE=n) is odd, LMS increments it to (n+1).

Member without an attribute record

When working with members which have not attribute record, bear in mind the following:

– If a BUFFER-LENGTH value is explicitly preset for the target file, LMS treats these members as members with an attribute record (see above).

  When phases are being created, BUFFER-LENGTH specifications other than STD(SIZE=1) or STD(SIZE=2) lead to errors.

– If a BUFFER-LENGTH value for the target file is not explicitly preset or known, LMS determines it as follows:

  – The BUFFER-LENGTH for phases is obtained from the current environment, i.e. BUFFER-LENGTH=STD(SIZE=1) for NK2 disks and STD(SIZE=2) for NK4 disks. The phases do not differ in terms of content.

  – Otherwise, the BUFFER-LENGTH is calculated from the maximum record length.

Recommendations for converting files from K/NK2 to NK4 disks

When converting files from an K/NK2 disk to an NK4 disk via a library, bear in mind the following recommendations:

1. “Actions” on the NK2 disk:

   Extract all “critical” members of the library as files. Critical members are “PAM” members under type X, which, as a file, have one of the following characteristics:

   – BUFFER-LENGTH = STD(SIZE=n), where n is odd
   – PAM key phases
   – 2K-oriented PLAM files

2. Using PAMCONV, convert all files with an odd BUFFER-LENGTH (except PLAM files) into NK4 files.
3. Using LMSCONV via type C, convert all PAM key phases to NK4 phases. The conversion only works if it is carried out on a K disk.
4. Using the LMS statement COPY-LIBRARY, convert NK2 PLAM files into NK4 PLAM files.
5. Using the LMS statement ADD-ELEMENT, add the NK4 files to an NK4 PLAM file and transfer that file to the NK4 disk.

Notes

– An NK4 library on a key disk can, for example, be converted to an NK4 disk via the BS2000 command COPY-FILE (see [4]) or by means of the BS2000 products FILE-TRANSFER (see [14]) and ARCHIVE (see [2]). The relevant operands for handling the BLOCK-CONTROL values can be found in the respective manuals.

– An NK2 PLAM file under type X cannot be converted into the NK4 PLAM format using EXTRACT-ELEMENT.

5.8 Handling alias names (ACS)

This section describes the effects of ACS on LMS. ACS (**A**lias **C**atalog **S**ervice) is a BS2000/OSD subsystem which is used to manage alias names for files. An alias name is an arbitrary file name which the user can employ instead of the files's real name.

Prerequisites

The ACS subsystem must have been started by the system administrator.

An alias catalog in which alias names are unambiguously assigned to real file names must have been generated for the current task.

If a task-specific alias catalog exists, a name specified by the caller of a function accessing a file's catalog entry will be assumed to be an alias name at first. Only when a name cannot be found in the alias catalog of the task is it then considered to be a real name.
If the specified name is defined as an alias name, it is replaced by the real file name with which it is associated.

Alias names with catalog and user identifiers may cause problems. For more information on working with alias names and on managing the alias catalog, see the section on "ACS" in the manual "Introductory Guide to DMS" [9].

Handling alias names in LMS

LMS replaces an alias name with the complete member name or file name in accordance with the catalog entry in the following cases:

Adding files

If a file is added as a member using ADD-ELEMENT, LMS always uses the complete file name for the construction of the member name.

Example

Alias name	File name
X	FILE.X

The following LMS statement generates the member S/FILE.X/001:

```
//ADD-ELEMENT FROM-FILE=X,TO-ELEM=*LIB(,ELEM=*BY-SOURCE(VERSION=001),TYPE=S)
```

Outputting members

If a member is output to a file using EXTRACT-ELEMENT, LMS always uses the complete member name for the construction of the file name.

Example

Alias name	File name
X	FILE.X

The following LMS statement generates the file FILE.X:

```
//EXTRACT-ELEMENT ELEM=*LIB(,ELEM=X,TYPE=S),TO-FILE=*BY-SOURCE
```

Logging file names

LMS always logs the complete, converted file name.

5.9 Using extended character sets in LMS (XHCS)

Computer systems (hosts) and data display terminals each operate with one **character set**, i.e. a set of letters, digits and characters used to form words and other basic components of a language.

By extending the character set, country-specific characters such as umlauts (German) and accents (French) can also be offered within a particular character set.

A **coded character set (CCS)** is the unique representation of the characters in a character set in binary form. The content of a coded character set and its rules, such as sorting order and conversion guidelines, are defined by international standards.

Example In the coded character set EBCDIC.DF.03-DRV (German reference version), the character “ä” is represented by the byte X’FB’, and in EBCDIC.DF.04-1 by X’43’.

Every coded character set (also called simply “code”) is identified by its unique name (**coded character set name, CCSN**).

Example The code EBCDIC.DF.03-IRV (international reference version) is referred to as “EDF03IRV”.

The appendix at the end of the “XHCS” manual [15] provides a list of all existing codes.

In BS2000, character sets are provided by the software product XHCS. By default, these include:

– 7-bit character sets such as, for example. `ISO646` (international 7-bit character set, ASCII), `EDF03IRV` (international reference version, EBCDIC), `EDF03DRV` (German reference version, EBCDIC).
– 8-bit character sets such as, for example. `ISO88591` (Latin Alphabet No.1, ASCII), `EDF041`
(Latin Alphabet No.1, EBCDIC), `EDF04DRV` (extension of `EDF03DRV`) etc.
– The 3 Unicode character sets `UTF16`, `UTF8` and `UTFE`.

5.9.1 Hardware and software requirements

In order to use extended codes for LMS, the following prerequisites must be met:

– software product XHCS (optional subsystem XHCS-SYS)

– software component VTSU

The software product **XHCS** (e**X**tended **H**ost **C**ode **S**upport) is required for generating extended codes in the host and for transmitting data between the host and the data display terminal. A detailed description of the principles and functions of XHCS as well as a list of code tables and names of standard codes are provided in the “XHCS” manual ([15]).

As hardware, it is necessary to have 8-bit data display terminals in order to input and output extended character sets.
You can use the software component VTSU to test data display terminals for 8-bit capability.

5.9.2 LMS-specific application of extended character sets

LMS supports the use of special (national) character sets, so each member can be assigned a character set through allocation of a coded character set name (CCSN). This is passed on to interfaces and taken into account in outputs.

If XHCS is not offered at the relevant interface, the default “no code” is always used.

To also enable selection via wildcards in the MODIFY-ELEMENT-ATTRIBUTES and SHOW-ELEMENT-ATTRIBUTES statements, the overlaying of “name” with <filename 1..20 without-cat-id-user-id-generation-version with-wildcards> is offered.

LMS itself does not require a specific character set and does not evaluate the default setting of the user ID. Internal LMS sort processes, e.g. of the member designations, take place independently of the selected CCS.

A character set can be assigned to every member in a PLAM library, and LMS always transfers the source member’s CCSN to the target member. The CCSN is a descriptive attribute and is output together with the directory.

As of PLAM V3.4A the CCSN of newly created elements will be derived from the CCSN of the library file unless the program creating the element doesn't specify a value of its own.

Setting CCSNs implicitly

CCSNs are set implicitly during the processing of members in the following cases:

– Adding members with ADD-ELEMENT
If a member is added using the LMS statement ADD-ELEMENT, the catalog attribute CCS for the member is also transferred. However, the CCSN is not stored in the attribute record (record type 164) to avoid inconsistencies. The layout of record type 164 is described in the “LMS Subroutine Interface” manual [1].

If a module from the EAM file is added using ADD-ELEMENT, it receives the “no code” attribute.

– Adding members from SYSDTA
If a member from the system file SYSDTA is added, the selected character set is determined, and the name of that character set is assigned to the member as an attribute.

– Processing members with EDT
Extended character sets are supported by EDT. When calling EDT, LMS passes the CCSN of the relevant member to EDT and then writes the member back with the value selected in EDT. If the input member is not specified, “no code” will be assumed as the input CCSN.

With EDT V17.0 the CCSN of an element newly created with EDIT-ELEMENT is the value determined by /MODIFY-TERMINAL-OPTIONS.

With EDT V16.6B this behaviour can be achieved by using the optional Rep A0538001. Without this Rep the value of the CCSN is *NONE.

Elements with ISO character set can't be extracted as ISAM files with EBCDIC ISAM keys.

– Copying members
When members are copied with the LMS statement COPY-ELEMENT, LMS always assigns the CCSN of the source member to the target member.

– Storing members with WRITE-MODE=*EXTEND
If a member is written back with WRITE-MODE=*EXTEND, LMS checks the CCSNs of the source and the target. If they do not match, processing is aborted and error message LMS0277 is issued.

Setting CCSNs explicitly

To set a CCSN explicitly, use the LMS statement MODIFY-ELEMENT-ATTRIBUTES. LMS does not check whether the specified CCSN is authorized in the system.

Logging CCSNs

The LMS statement SHOW-ELEMENT-ATTRIBUTES with the operand INFORMATION=*MAXIMUM outputs the assignment of character sets to the members. In the case of members having the “no code” CCSN, this attribute is not displayed in the directory.

Example

The character set EDF03IRV is assigned to the TEST member by the LMS statement MODIFY-ELEMENT-ATTRIBUTES and is stored under the new name TEST1. The member attributes are then output with SHOW-ELEMENT-ATTRIBUTES.

```
//MOD-ELEM-ATTR (LIB=TESTLIB,ELEM=TEST,TYPE=S), -
//              NEW-ATT=PAR(ELEM=TEST1;CODED-CHAR-SET=EDF03IRV)
OUTPUT LIBRARY= :X:$USERID.TESTLIB
        MODIFY (S)TEST/001(001)2012-06-22 AS (S)TEST1/001(0001)/2012-06-22
//SHOW-ELEM-ATTR (ELEM=TEST1),INFORMATION=*MAXIMUM
INPUT  LIBRARY= :X:$USERID.TESTLIB
TYPE       = S
NAME       = TEST1
VERSION    = 001                                    VARIANT    = 0001
USER-DATE  = 2012-06-22   CRE-DATE   = 2012-06-24 MOD-DATE   = 2012-06-24
USER-TIME  =   13:07:37   CRE-TIME   =   14:38:12 MOD-TIME   =   14:41:40
STORAGE    = FULL         COD-CH-SET = EDF03IRV
STATE      = FREE
ELEM-SIZE  = 1
       1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
```

Evaluating and transferring CCSNs

The CCSN is evaluated when members are output.

– Outputting members to file
  The file is given the CCSN of the member.

– Outputting members to SYSOUT
  When member records are output to SYSOUT (also in edited form) by means of the SHOW-ELEMENT statement, the CCS of the relevant member is used.

  If SYSOUT is assigned to a file, the user must explicitly assign the desired character set to this file using the BS2000 command MODIFY-FILE-ATTRIBUTES.

– Outputting members to SYSLST
  When member records are output to SYSLST, the CCSN is not evaluated.

  If SYSLST is assigned to a file, the user must explicitly assign the desired character set to this file using the BS2000 command MODIFY-FILE-ATTRIBUTES.

– Outputting members to a library member
  If member output is redirected to a library member by means of the LMS statement MODIFY-LOGGING-PARAMETERS, the member receives the “no code” CCSN. Nonetheless, this member can be assigned a character set explicitly with the LMS statement MODIFY-ELEMENT-ATTRIBUTES.

– Outputting the directory
  When the directory or other member information generated by LMS is output, the “no code” CCSN is always assumed.

5.10 Utilizing LMS functionality from within EDT

By means of the @USE statement, EDT offers users the option of defining an external statement routine which can be called up as a subprogram via a user escape character also defined using @USE. This makes user-specified statements available in EDT.

In order to utilize LMS statements and thus the functionality of LMS from within EDT, the @USE statement must be specified in the EDT command line in the following form:

```
@USE COM[MAND]= 'usersymb' (LMSEDT,$.SYSLNK.LMS.034)
```

with the external statement form: *usersymb* `LMS statement`

A detailed description of the @USE statement is provided in the “EDT” manual [10].

Please note that the maximum length of an LMS statement entered in the EDT command line is only 256 bytes.

When an LMS statement is entered, it is analyzed and executed by the SDF command processor, which then returns the user to the EDT command line. Lowercase letters are converted to uppercase prior to analysis.

Notes

- The LMS statements CALL-EDT, EDIT-ELEMENT and BEGIN-MAKE are not supported in the EDT command line.
- External statement routines do not support any LMS start files.
- LMS functionality can be utilized from within EDT
  - if EDT is called up as a main program or
  - if EDT is called up as a subprogram in the main program LMS using one of the LMS statements //CALL-EDT or //EDIT-ELEMENT.
- If LMS is the main program, using LMS functionality within EDT causes no new LMS run to be started, i.e. the LMS settings from the current run remain in effect.
- If EDT is running in Unicode mode e.g. after /START-EDTU (as of EDT V17) then entry LMSEDU is to be used instead of LMSEDT in EDT-statement @USE. If entry LMSEDT is used in Unicode mode then an element created by ADD-ELEMENT gets CCSN *NONE even though the character set in EDT set by CODENAME is not *NONE (e.g. UTF16).

The effects of the LMS statements EXTRACT-ELEMENT, ADD-ELEMENT and END are somewhat altered when they are used from within EDT.

The LMS statement is entered in the EDT command line:

– EXTRACT-ELEMENT

```
usersymbEXTRACT-ELEMENT ELEMENT = ... ,
                          TO-FILE = *STD
```

The member records of the specified member are appended to the end of the current EDT work file.

– ADD-ELEMENT

```
usersymbADD-ELEMENT FROM-FILE=*STD ,
                      TO-ELEMENT = ...
```

The data records are written from the current EDT work file into the specified member.

– END

*usersymb*END

– EDT was called as a main program:

The subprogram LMS called via @USE is terminated, i.e. all libraries still open are closed. The user is returned to the EDT command line.

– LMS was called as a main program:

In interactive mode, the user is asked:

```
LMS0409 TERMINATE LMS? REPLY (Y=YES, N=NO)
```

Y: The main program LMS is terminated.
N: The main program LMS is not terminated.

Extension of LMS statements

```
MODIFY-LOGGING-PARAMETERS ... ,
                       TEXT-OUTPUT = *EDT (WRITE-MODE=*UNCHANGED)
```

LMS log output is written to EDT work file 9; the WRITE-MODE operand refers to the contents of work file 9.

Example

```
/START-EDT ──────────────────────────────────────────────────────────────  (1)

.
 .
1.00
2.00



22.00
23.00
@USE COM='.'(LMSEDT,$.SYSLNK.LMS.034)                          0000.00:001(0)  -- (2)
LTG                                                                 TAST
```

```
1.00  Member contents of xmpl.mem1
2.00



22.00
23.00
.extract-elem (lib=xmpl.lib,elem=xmpl.mem1,type=s)             0000.00:001(0)  --(3)
LTG                                                                 TAST
```

```
1.00  Modified member contents of xmpl.mem1
2.00



22.00
23.00
.add-elem *std,(lib=xmpl.lib,elem=xmpl.mem2,type=s)            0000.00:001(0)  --(4)
LTG                                                                 TAST
```

(1) EDT is invoked.

(2) LMS is defined as an external statement routine with the user escape character "period".

(3) The member xmpl.mem1 of type=s is read into EDT work file 0 from the xmpl.lib library.

(4) The member is renamed and written back into the same library from EDT work file 0.

5.11 LMS and EDT V17

LMS by default calls EDT V17 in compatibility mode so that existing procedures run furthermore without changes. During editing of Unicode elements, EDT switches over to Unicode mode automatically.

To edit elements with records longer than 255 characters but no Unicode character set specified please use one of the following LMS statements to call EDT in Unicode mode:

```
//CALL-EDT EDT-MODE=*UNICODE
```

or

```
//EDIT-ELEMENT ...,EDT-MODE=*UNICODE
```

Alternatively you can set the Unicode mode as default for EDT calls:

```
//MODIFY-LMS-DEFAULTS EDT-MODE=*UNICODE
```

6 Support for the software development process

Some terms which are used in the following sections are defined below to facilitate understanding:

program system

A program system is a logical grouping of software components, for example the components of a software project.

component

A component contains data which belongs together. One example of a component is a library member that contains text.

derived component

A derived component is one which is generated from other components through automatic processing, for example a compilation listing.

source (component)

A source component is one which was not generated from any other component. As a rule, sources are components which are developed manually, for example the source texts of programs, macros or documents.

current

A program system may be referred to as being “current” if it includes all the recent updates of the sources from which it is generated.

dependency

A component is dependent on the components from which it is derived.

6.1 Borrowing mechanism

With its borrowing mechanism, LMS affords a means of monitoring and controlling the development of components by two or more developers at the same time. First, it is necessary to activate WRITE-CONTROL for a given library or a type with the MODIFY-LIBRARY-ATTRIBUTES or MODIFY-TYPE-ATTRIBUTES statement. This ensures that a member can be written only if the user explicitly or implicitly holds the base version. Under the STD-SEQ convention, the holder of the highest version also has the entire version space reserved. Under the STD-TREE convention, the holder of the highest version of a branch has the entire branch reserved. Under the MULTI-SEQ convention, the holder of the highest version of a subname space reserves the entire subname space.

Borrowing member versions

First, the user obtains a member by means of the PROVIDE-ELEMENT statement. This changes the status of the member from FREE to IN-HOLD and makes a copy of the member available. It is then no longer possible to issue a PROVIDE-ELEMENT statement for that member.

It is also possible to mark a member as held by explicitly setting its status with the MODIFY-ELEMENT-ATTRIBUTES statement. In this case, no copy is made available, but can be obtained later using the COPY-ELEMENT statement.

Returning members

To return a member, use the RETURN-ELEMENT statement. If BASE=*STD is specified, the member version held by the user is determined. If the user holds multiple versions, the base must be specified explicitly. The member is then again available for holding by other users; the hold flag is deleted. When you return a member, you can return a comment along with it. If WRITE-CONTROL has been activated, the HOLDER, DATE and TIME of return are automatically recorded. You can view these members using EDIT-ELEMENT.

Cancelling reservations

With the MODIFY-ELEMENT-ATTRIBUTES statement, you can cancel a reservation for a held member, for example if it has become unnecessary to return the member.

Granting hold rights

The owner of the library can specify who has hold authorization for each member using MODIFY-ELEMENT-PROTECTION; only the person(s) thus authorized can hold the member.

6.2 make functionality

make functionality is designed to facilitate the integration of program systems and intended for use in small to medium-scale software projects where integration is to be handled without extensive administrative.

A program system consists of components such as macros, “includes”, source programs, object modules, main modules, phases, etc. A component is referred to as being *dependent* if it is generated from other components.

If changes are made in the program system, only the dependent components need to be regenerated to ensure that the program system is up to date. Frequently, changes affect two or more levels.

Example

An object module is generated from a source program and the macros used, and a main module from a series of object modules.

1. If a source program is changed, it is necessary to generate (compile) the corresponding object module again.
2. The main module containing that object module must then also be regenerated.
3. Then the phase containing that object module must be regenerated.

The make functionality can be used to integrate a program system in such as way that, when changes are made in the source programs, only the program-system components affected by the modifications have to be regenerated .

For the make run, it is necessary to consider the following questions:

– Which component is dependent on which other component?

– What are the rules according to which the dependent components are generated?

This information is stored in the make substatements (see the next page), where the rules (referred to as “actions” in the following sections) are also defined (see page 131).

The components which can be processed with the make functionality are library members and files. A dependent component must be generated again if it is older than any of the components on which it is dependent. Depending on the component involved, the time of the most recent modification is considered to be:

– the MODIFICATION-DATE/TIME in the case of library members

– the LAST-CHANGE-DATE in the case of files.

The time of the modification is indicated in local time.

Starting make

To start the make functionality, you use the BEGIN-MAKE statement, in which you specify the target component (TARGET) and generate a procedure that regulates continuation processing. The system expects the BEGIN-MAKE statement to be followed by make substatements.

make substatements: describing dependencies and defining actions

The substatements specify the dependencies existing among the components and define actions. The actions describe how a target component is to be generated from the source components and are stored in the procedure specified in the BEGIN-MAKE statement.

The following substatements are available for make:

MODIFY-MAKE-DEFAULTS	Modifies make-internal presettings.
SET-STD-ACTION	Specifies dependencies between individual objects by defining standard actions which use components of one member type to generate components of a different member type.
SET-DEPENDENCY	Specifies the dependencies which exist between the individual components. Can also be used to define non-standard actions which are to be executed at a later point in time if the target component is not current. Large numbers of objects can be read in in SET-DEPENDENCY via S variables.
SET-PREPROCESSING	Defines the actions that are to be included at the beginning of the generated procedure.
SET-POSTPROCESSING	Defines the actions that are to be included at the end of the generated procedure.

Terminating make

The make substatement END-MAKE concludes the sequence of make substatements and initiates continuation processing. Generally, the continuation processing generates and executes a procedure.

Schematic diagram of make processing

There are two ways to “update” a program system:
– by generating the updated versions of the outmoded components
– by “touch”, i.e. by updating only the modification time of the outmoded components (see page 134).

figure 9 shows a schematic diagram of make processing.

The diagram depicts a program system and its components, in which the source components S1 and S2 are used to generate the target component T1.

If the source component S1 is modified, the component dependent on it (target component T1) must be generated again. This means that the make run must be called with the target component T1. The make run describes two things: first, which components are dependent on source component S1, and second, the actions which are required to generate the dependent component.

After the make run has concluded, the procedure which was generated during the run and which contains those actions will be executed. Since the dependent component T1 has a modification time which is earlier than that of the component S1, the T1 component is updated according to the actions contained in the procedure.

In figure 9, the numbers in parentheses have the following meanings:

(1) Component T1 is dependent on components S1 and S2. Component S1 is then modified. The make run is called with the target component T1 in LMS.

(2) In the make substatements composing the make run (BEGIN-MAKE through END-MAKE), it is specified that component T1 is dependent on component S1. The statements also define the actions required to generate T1 from S1.
The actions required to update the program system are collected in the procedure specified in the BEGIN-MAKE statement.

(3) After the make run has concluded (after input of the END-MAKE statement), the procedure is executed.

(4) “Touch” updates the components’ modification times to the latest status without modifying their contents.

(5) The updated program system with the newly generated T1 component.

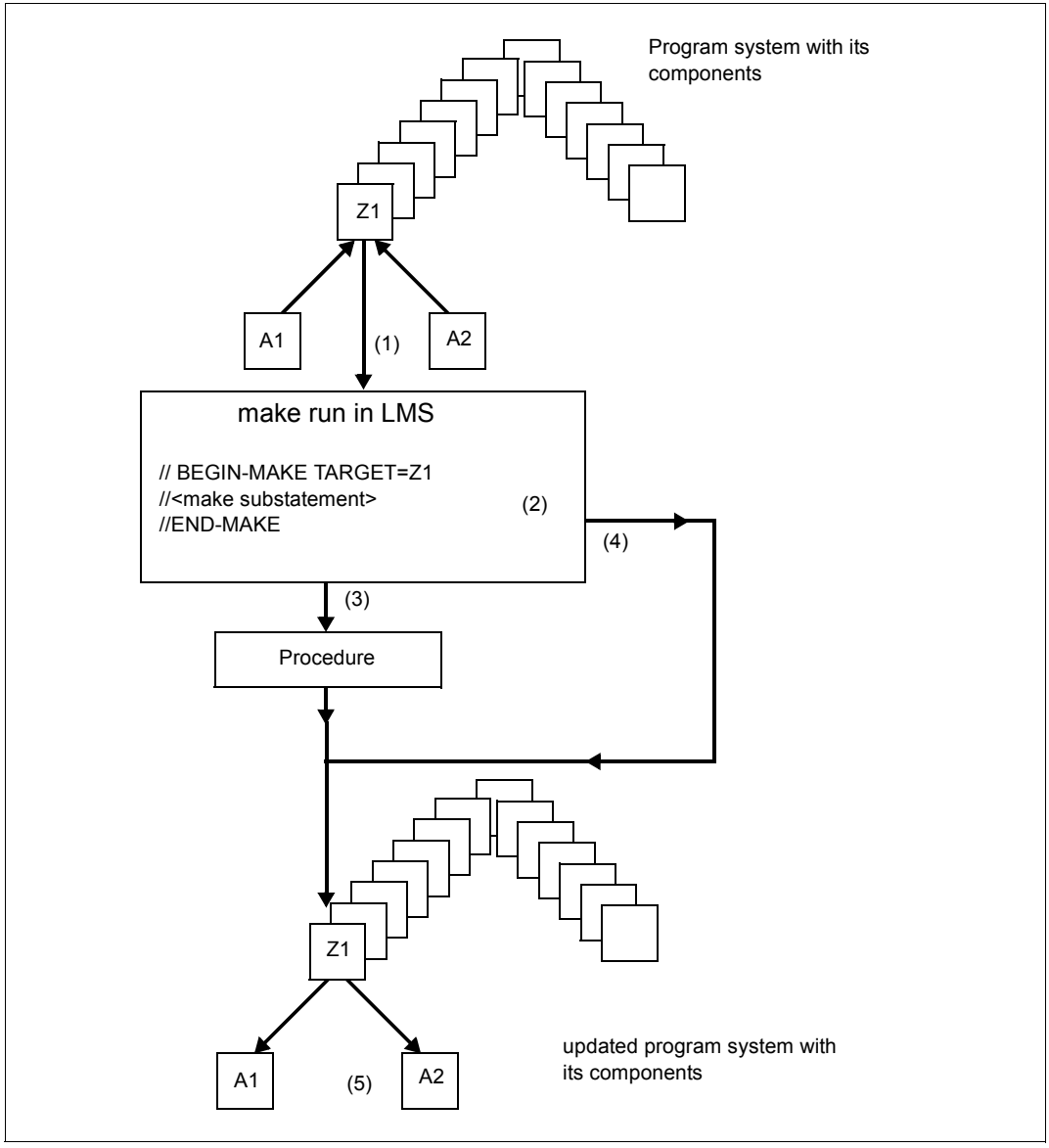


Figure 9: Overview of make processing

6.2.1 Actions

Actions describe how a target component is generated from the source components.

Actions are text lines which are specified in the make substatements SET-DEPENDENCY or SET-STD-ACTION. These text lines are transferred unchanged into the procedure specified in the BEGIN-MAKE statement.
Each action has its own line in the procedure. This makes it possible to specify two or more lines (actions) for a single component. Users formulating make statements must themselves take care of continuation lines within actions.

In the SET-STD-ACTION substatement, you can specify standard actions which use source components of a specific type to generate target components of a different type. These standard actions can be referenced in the SET-DEPENDENCY statement.

6.2.2 Using variables

In order to exploit the full scope of make functionality, it is necessary to have S variables, which in turn means that SDF-P must be available (see [12]).

S variables generated by make (make S variables) can be used in actions. These variables contain information both on the dependent components which are to be generated and the components from which they are to be generated. make inserts the required make S variable assignments into the procedure before the actions.

All of the S variables specified in the make substatements are replaced during the make run, which is made possible by the SDF input language.

You can prevent the S variables from being replaced during the make run by entering the ampersand (&) twice, instead of once, when you specify the actions.

Large quantities of data intended for components can be read in via S variables in the SET-DEPENDENCY statement.

There are two categories of variables that are not to be generated until the generated procedure is executed:

– normal S variables which are supplied and used (in actions) by the user,

– make S variables which are supplied and used in actions by make.

The names of make S variables are defined in the MODIFY-MAKE-DEFAULTS substatement. If make S variables are accessed in the course of actions, the actions must be processed by SDF.

Users may define the following make S variables:

– Name of the current target (CURRENT-TARGET-VAR)
(see make substatement MODIFY-MAKE-DEFAULTS)

– List of all predecessor components (FROM-OBJECTS-VAR)
(see make substatement MODIFY-MAKE-DEFAULTS)

– List of all predecessor components which have been modified more recently than the target (MODIFIED-OBJECTS-VAR)
(see make substatement MODIFY-MAKE-DEFAULTS).

The value for CURRENT-TARGET-VAR is a variable in the output format of LMS (see chapter “Format of LMS output in S variables” on page 445). The values for FROM-OBJECTS-VAR and MODIFIED-OBJECTS-VAR are lists of such variables. The sequence of the variables in these lists corresponds to the sequence of the variables specified in the SET-DEPENDENCY statement.

For library members, the following structure elements are supplied (variable name VAR):

&(VAR.LIB)	STRING
&(VAR.ELEM)	STRING
&(VAR.VERSION)	STRING
&(VAR.TYPE)	STRING

If the S variable is supplied with a value for a file, the variable VAR.LIB receives the complete file name, and the variables VAR.ELEM, VAR.VERSION and VAR.TYPE receive empty strings. The representation of *NONE contains only empty strings.

You supply the S variables with values by inserting commands in the generated procedure, making use of the auxiliary variable SYSLMSMAKE.

Declarations are also made by inserting commands in the generated procedure. Declarations made in the substatement for preprocessing are not overwritten.

6.2.3 Selection and construction specifications in make

Selection and construction specifications result in the processing of multiple components.

Selection specifications in make

With the selection specification, you can reference target components. Patterns in the selection specification can be used to form source components (see the description of construction specifications below). Selection specifications refer to target components which are known to make and which are generated gradually through the specified dependencies during the make run.

Selection specifications apply to these components only, **not** to the contents of libraries. Selection specifications may be made for ELEMENT, TYPE and library name.

Construction specifications in make

Construction specifications can be used to form designations of source components from target components, where ELEMENT corresponds to ELEMENT and TYPE to TYPE. The file names specified correspond to one another and to ELEMENT.

If multiple target and source components are specified, the selection and construction processes are executed for all combinations of those components. The file names specified correspond to one another and to ELEMENT.

The LMS restriction that at least one placeholder of the selection specification must occur in the construction specification does not apply here.

6.2.4 Runtime control during continuation processing

Global runtime control for make is effected by means of the statements MODIFY-MAKE-DEFAULTS and BEGIN-MAKE.

The output of the log for continuation processing can be controlled with BS2000 commands. Either of the SDF-P commands MODIFY-PROCEDURE-OPTIONS or MODIFY-PROCEDURE-TEST-OPTIONS, which control logging, can be specified as the first command under ACTION (in SET-PREPOCESSING for the entire procedure).

The response to errors is controlled by means of the SUPPRESS-ERRORS operand of the make substatement SET-DEPENDENCY. If SUPPRESS-ERRORS=*NONE and a command return code is set following execution of the actions specified by the dependency definition, execution of the procedure generated by make is aborted (see [12]). This corresponds to the conventional spin-off mechanism. In this way, the response to errors can be controlled separately for each dependency definition.

Error situations are reset if STEP commands are inserted in the generated procedure. In this case, no program may be loaded at the end of the actions.

Case-specific error handling may be implemented by inserting the appropriate commands in the actions.

6.2.5 TOUCH

As an alternative to generating and executing a procedure, you can “touch” the components (SUCCESS-PROCESSING=*TOUCH in the BEGIN-MAKE statement), which causes them to be given an updated modification time. In this case, no procedure is generated. Following END-MAKE, the time stamps of the components are updated, but their contents remain unchanged.

6.2.6 make operation

In order to determine where dependencies exist, a graph describing the entire program system is constructed, taking as its starting point the component specified as the target component in the BEGIN-MAKE statement. The graph is then elaborated by evaluating the existing dependencies (make substatement SET-DEPENDENCY).

Dependencies have the following structure:

```
(target-object1, target-object 2),(from-object1, from-object2,...),
(action1,...)
```

where this notation signifies:
– that the target objects are dependent on the source components (from-objects) and
– that the actions describe how a target component is to be generated.

A target object is not current if any of the associated source components (from-objects) have a more recent modification time, or the target component is not available.

The dependency represented above is equivalent to the following:

```
target-object1, (from-object1, from-object2,...), (action1,...)

target-object2, (from-object1, from-object2,...), (action1,...)
```

At first, the graph consists solely of the target of the make run, but is then gradually elaborated with the aid of the dependencies. If a component in the graph is a target component and has as yet no successor, the associated source components are appended to that node.

The graph is finished when make has applied all of the existing dependencies. The components of the graph which have no source components (from-objects) are the sources of the program system. They must exist in full form in order to generate the target of the make run. All the sources for a target component must exist (not just the modified components), or the make run will be aborted and an error message issued.

For each target component, only one uniquely specified action may exist. Otherwise, the make run will be aborted and an error message issued.

Taking the sources as starting points, make evaluates the relations between the target components and the source components in regard to their modification times. If it determines that a target component is not current, make initiates the action required to update it.

Example

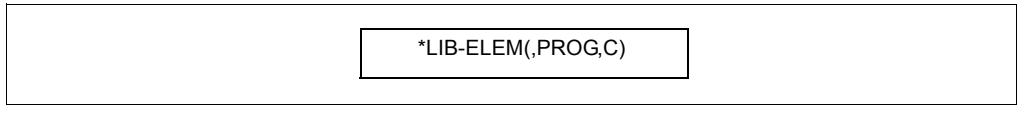
This example shows how the make functionality works when wildcards are used. The library where the components are located is the preset library.

The phase member PROG (member type C) is generated from the object modules PART1 and PART2 (both of member type R). These objects are compiled from sources of type S with the suffix ".COB". The actions are merely mentioned.

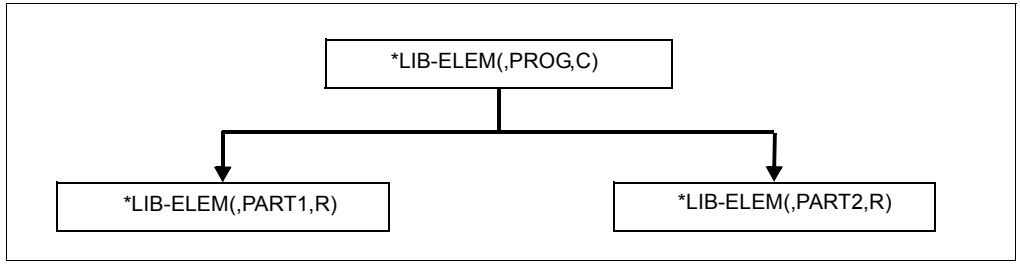
The dependencies (library members in the format (,NAME,TYPE)):

```
(1) TARGET-OBJECT1 = *LIB-ELEM(,PROG,C), -
    (FROM-OBJECT1=*LIB-ELEM(,PART1,R), FROM-OBJECT2=*LIB-ELEM(,PART2,R)), -
    ACTION1 = 'LINK PROG'
(2) TARGET-OBJECT2 = *LIB-ELEM(,*,R) , -
    FROM-OBJECT = *LIB-ELEM(,**.COB,S), -
    ACTION2 = 'COMPILE'
```

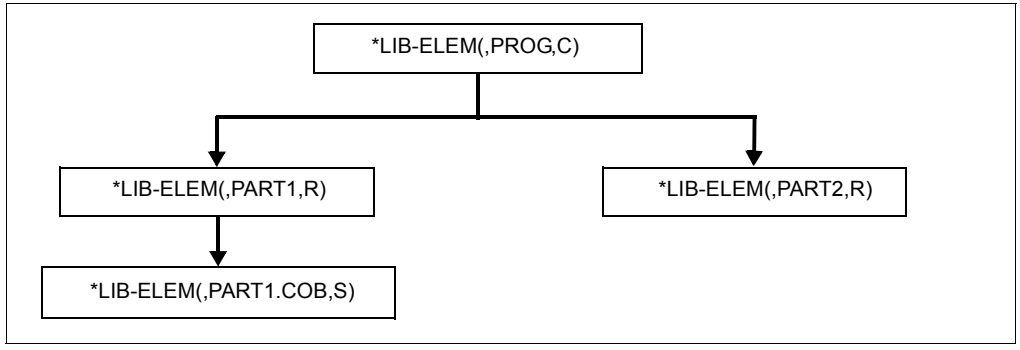
If it is to be generated in the make run, the component *LIB-ELEM (,PROG,C) is the start node of the graph.



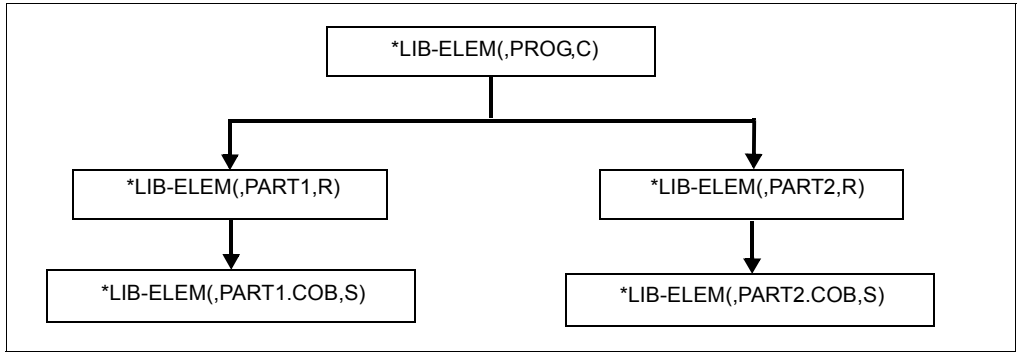
The graph can be elaborated only with dependency (1). The result:

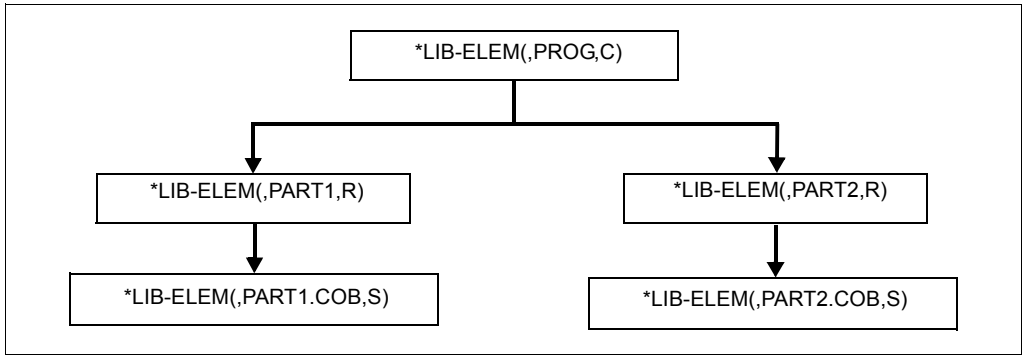


As a make selection specification, the target component of dependency (2) corresponds to *LIB-ELEM(,PART1,R), which supplies the associated source components as the result of the construction *LIB-ELEM(,PART1.COB,S). In the graph, the latter is appended to *LIB-ELEM(,PART,R).



In the same way, (,PART2.COB,S) is appended to (,PART2,R).





*LIB-ELEM(,PART1.COB,S) and *LIB-ELEM(,PART2.COB,S) are the sources required for the generation of *LIB-ELEM(,PROG,C) and must be available in full form.

7 Statements

This chapter describes the statements that can be entered during the LMS run. The description of the LMS statements follows the overview of SDF standard statements and the syntax description of the SDF user interface. In order to avoid ambiguity, it is advisable to write "/ /" in front of all statements in procedures.

7.1 SDF standard statements for LMS

The following SDF standard statements may be specified during the LMS run.

Statement	Function
END	Terminate the LMS run
EXECUTE-SYSTEM-CMD	Execute a system command
HELP-MSG-INFORMATION	Output system message text to SYSOUT
HOLD-PROGRAM	Interrupt a program
MODIFY-SDF-OPTIONS	Modify the SDF options
REMARK	Provide comments for programs
RESET-INPUT-DEFAULTS	Reset task-specific default values
RESTORE-SDF-INPUT	Redisplay the last input
SHOW-INPUT-DEFAULTS	Display task-specific default values
SHOW-INPUT-HISTORY	Output contents of the input buffer
SHOW-SDF-OPTIONS	Display the SDF options
SHOW-STMT	Display the syntax of a statement
STEP	Define a restart point
WRITE-TEXT	Output text to SYSOUT

The SDF standard statements (except for END) are not described in the present manual. A description may be found in [3].

7.2 Syntax description

This syntax description is valid for SDF V4.7. The syntax of the SDF command/statement language is explained in the following three tables.

Metasyntax

The meanings of the special characters and the notation used to describe command and statement formats are explained in 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 3.

The description of the data types is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described in table 3 are explained in the relevant operand descriptions.

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:

case-sensitive	case-sens
cat-id	cat
completion	compl
correction-state	corr
digits	dig
generation	gen
lower-case	low
manual-release	man
odd-possible	odd
path-completion	path-compl
separators	sep
special-characters	spec
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 4 contains a number of items in italics which are not part of the syntax. They are only used to make the table easier to read. For special data types that are checked by the implementation, table 4 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 table 4.

Metasyntax

Convention	Meaning	Examples
UPPERCASE	Uppercase letters denote keywords (command, statement or operand names, keyword values) and constant operand values. Keyword values begin with *.	**HELP-SDF** **SCREEN-STEPS** = ***NO**
UPPERCASE in boldface	Uppercase letters printed in boldface denote guaranteed or suggested abbreviations of keywords.	**GUID**ANCE-**MODE** = ***Y**ES
=	The equals sign connects an operand name with the associated operand values.	**GUID**ANCE-**MODE** = ***NO**
< >	Angle brackets denote variables whose range of values is described by data types and suffixes (see 3 and 4).	**SYNTAX-F**ILE = <filename 1..54>
Underscoring	Underscoring denotes the default value of an operand.	**GUID**ANCE-**MODE** = ***NO**
/	A slash serves to separate alternative operand values.	**NEXT-FIELD** = ***NO** / ***Y**ES
(...)	Parentheses denote operand values that initiate a structure.	,**UNGUID**ED-**DIA**LOG = ***Y**ES (...) / ***NO**
[ ]	Square brackets denote operand values which introduce a structure and are optional. The subsequent structure can be specified without the initiating operand value.	**SELECT** = [***BY-ATTR**IBUTES](...)

Table 2: Metasyntax (part 1 of 2)

Convention	Meaning	Examples
Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	,**GUID**ED-**DIA**LOG = ***Y**ES (...) ***Y**ES(...) **SCREEN-STEPS** = ***NO** / ***Y**ES
|	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 2: 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	Alphanumeric string that can be split into multiple substrings by means of a period or hyphen. If a file name can also be specified, the string may begin with a catalog ID in the form :cat: (see data type filename).
c-string	EBCDIC character	Must be enclosed within single quotes; the letter C may be prefixed; any single quotes occurring within the string must be entered twice.
date	0…9 Structure identifier: hyphen	Input format: yyyy-mm-dd yyyy: year; optionally 2 or 4 digits mm: month dd: day
device	A…Z 0…9 hyphen	Character string, max. 8 characters in length, corresponding to a device available in the system. In guided dialog, SDF displays the valid operand values. For notes on possible devices, see the relevant operand description.
fixed	+, - 0…9 period	Input format: [sign][digits].[digits] [sign]: + or - [digits]: 0...9 must contain at least one digit, but may contain up to 10 characters (0...9, period) apart from the sign.

Table 3: Data types (part 1 of 6)

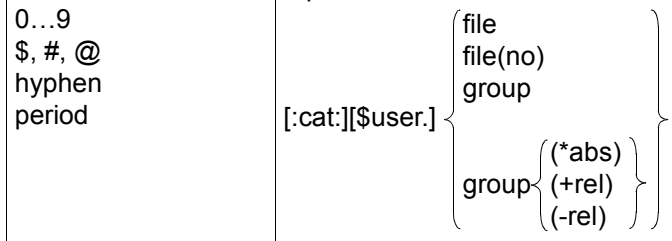
Data type	Character set	Special rules
filename	A…Z 0…9 $, #, @ hyphen period	Input format:  [:cat:][$user.] { file | file(no) | group | group { (*abs) | (+rel) | (-rel) } } :cat: optional entry of the catalog identifier; character set limited to A...Z and 0...9; maximum of 4 characters; must be enclosed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

Table 3: Data types (part 2 of 6)

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

Table 3: Data types (part 3 of 6)

Data type	Character set	Special rules
partial-filename	A…Z 0…9 $, #, @ $, #, @ hyphen	Input format: [:cat:][$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; max. 1023 characters; must be enclosed within single quotes if alternative data types are permitted, separators are used, or the first character is a ?, ! or ^.

Table 3: Data types (part 4 of 6)

Data type	Character set	Special rules
product-version	A…Z 0…9 period single quote	Input format: [[C]'][V][m]m.naso['] correction status release status where m, n, s and o are all digits and a is a letter. 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 4). 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 $, #, @ $, #, @	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 3: 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 3: Data types (part 6 of 6)

Suffixes for data types

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

Table 4: Data type suffixes (part 1 of 7)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with (contd.)</td><td></td></tr>
<tr><td>-under</td><td>Permits underscores (_) for the data type “name”.</td></tr>
<tr><td>-wild(n)</td><td>Parts of names may be replaced by the following wildcards.
n denotes the maximum input length when using wildcards. Due to the introduction of the data types posix-filename and posix-pathname, SDF now accepts wildcards from the UNIX world (referred to below as POSIX wildcards) in addition to the usual BS2000 wildcards. However, as not all commands support POSIX wildcards, their use for data types other than posix-filename and posix-pathname can lead to semantic errors.
Only POSIX wildcards or only BS2000 wildcards should be used within a search pattern. Only POSIX wildcards are allowed for the data types posix-filename and posix-pathname. If a pattern can be matched more than once in a string, the first match is used.
<table>
<tr><th>BS2000 wildcards</th><th>Meaning</th></tr>
<tr><td>*</td><td>Replaces an arbitrary (even empty) character string. 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.</td></tr>
<tr><td>Terminating period</td><td>Partially-qualified entry of a name.
Corresponds implicitly to the string “./*”, i.e. at least one other character follows the period.</td></tr>
<tr><td>/</td><td>Replaces any single character.</td></tr>
<tr><td><s_x:s_y></td><td>Replaces a string that meets the following conditions:
a) If s_x is shorter than or exactly as long as s_y:
– It is at least as long as s_x and no longer than s_y
– In the alphabetic collating sequence it lies in the range from s_x to s_y
– s_x may be empty (=1 character with the lowest coding)
b) If s_x is longer than s_y:
– It is at least as long as s_y and no longer than s_x
– In the alphabetic collating sequence it lies in the range from s_x to a string which begins with s_y and is filled with characters of the highest possible coding to the length of s_x
– s_y may be empty (=1 character with the highest coding)</td></tr>
</table>
</td></tr>
</table>

Table 4: Data type suffixes (part 2 of 7)

Suffix	Meaning	
with-wild(n) (contd.)	<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 page 148).
	-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 an arbitrary (even empty) character 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 4: Data type suffixes (part 3 of 7)

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

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

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

Table 4: Data type suffixes (part 5 of 7)

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

Table 4: Data type suffixes (part 6 of 7)

Suffix	Meaning
without (contd.)	
-temp-file	Specification of a temporary file is not permitted (see #file or @file under filename).
-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.
case-sensitive	When specifications are made for the data types command-rest and text, a distinction is made between uppercase and lowercase. This also applies for specifications which are not enclosed in single quotes.

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

7.3 Input rules

The LMS statements are read via the SDF user interface and processed by the command processor SDF (System Dialog Facility). Different forms of guided or unguided dialog exist with the facility to request help menus for the statements. See also [3].

Continuation lines

It is also possible for statements to extend over more than one record. Splitting is governed by the BS2000 command language conventions. A hyphen (-) is used as the separator character. Statement lines may be up to 32763 characters long.

Abbreviation options

When entering LMS statements it is permissible to abbreviate statement names, operand names and keywords.

The following rules then apply:

It is possible in each case to abbreviate from right to left so long as *uniqueness* is maintained. This applies both to the name as a whole and to subnames (beginning with a hyphen) and allows for the possibility of the subname being omitted entirely.

The guaranteed abbreviation options for all statements, operands and operand values are indicated in the syntax descriptions of the statements by boldface print. It is however possible to abbreviate beyond these (so long as uniqueness is maintained within a structure).

In order to avoid ambiguity resulting from functional extensions in future versions and to ensure readability for other users, abbreviations should be avoided in procedures.

Positional operands

SDF allows the optional specification of operands as keyword operands or positional operands. However, the possibility of an operand position changing in a subsequent version cannot be completely ruled out. It is therefore advisable to avoid using positional operands in procedures.

7.4 Statement aliases

The following table shows the aliases of some of the most frequently used statements. These names cannot be abbreviated further.

Statement	Alias
COPY-ELEMENT	CP / CPE
COPY-LIBRARY	CPL
EDIT-ELEMENT	EDE
EDIT-ELEMENT-ATTRIBUTES	EDEA
EDIT-ELEMENT-PROTECTION	EDEP
MODIFY-ELEMENT	MDE
MODIFY-ELEMENT-ATTRIBUTES	MD / MDEA
MODIFY-ELEMENT-PROTECTION	MDEP
MODIFY-LIBRARY-ATTRIBUTES	MDLA
MODIFY-LMS-DEFAULTS	MDD / MDLMSD
MODIFY-LOGGING-PARAMETERS	MDLGP
MODIFY-TYPE-ATTRIBUTES	MDTA
RESET-LMS-DEFAULTS	RSD / RSLMSD
RESET-LOGGING-PARAMETERS	RSLGP
RESET-TYPE-ATTRIBUTES	RSTA
SHOW-ELEMENT	SHE
SHOW-ELEMENT-ATTRIBUTES	SH / SHEA
SHOW-LIBRARY-ATTRIBUTES	SHLA
SHOW-LIBRARY-STATUS	SHLS
SHOW-LMS-DEFAULTS	SHD / SHLMSD
SHOW-LOGGING-PARAMETERS	SHLGP
SHOW-STATISTICS	SHST
SHOW-TYPE-ATTRIBUTES	SHTA
SHOW-USER-EXITS	SHUE

Table 5: Statement aliases

7.5 Description of the LMS statements

This section contains a tabular overview of the LMS statements followed by a description of all the statements in alphabetical order, organized as follows:

– statement name and function
– description of function
– representation of format
– description of operands
– required access rights
– notes
– examples

The substatements for the statement MODIFY-ELEMENT are described in alphabetical order by name following the statement (page 287ff) since together they constitute a logical unit.

The same is true of the make substatements, which are described following the BEGIN-MAKE statement starting on page 182.

The *LMS-DEFAULT keyword is no longer described in the individual statements. It generally signifies the value set with the MODIFY-LMS-DEFAULTS statement.

The following additionally applies for the member type:
The type specification *LMS-DEFAULT leads to an error if neither a global nor a library-specific default type is set.

Overview

Statement	Function
ACTIVATE-USER-EXIT	Activate a user exit
ADD-ELEMENT	Add files to libraries as members
BEGIN-MAKE	Initiates make substatements
CALL-EDT	Call up EDT
CLOSE-LIBRARY	Close a library
COMPARE-ELEMENT	Compare members
COPY-ELEMENT	Copy members
COPY-LIBRARY	Copy libraries
DEACTIVATE-USER-EXIT	Deactivate a user exit

Statement	Function
DELETE-ELEMENT	Delete members
EDIT-ELEMENT	Edit members using EDT
EDIT-ELEMENT-ATTRIBUTES	Start guided dialog for MODIFY-ELEMENT-ATTRIBUTES
EDIT-ELEMENT-PROTECTION	Start guided dialog for MODIFY-ELEMENT-PROTECTION
END	Terminate the LMS run
EXTRACT-ELEMENT	Store members in files
FIND-ELEMENT	Find member in records using wildcards
MODIFY-ELEMENT	Modify members using substatements
MODIFY-ELEMENT-ATTRIBUTES	Modify member attributes
MODIFY-ELEMENT-PROTECTION	Modify the member protection
MODIFY-LIBRARY-ATTRIBUTES	Modify the library attributes
MODIFY-LMS-DEFAULTS	Modify the LMS default values
MODIFY-LOGGING-PARAMETERS	Modify the LMS logging parameters
MODIFY-TYPE-ATTRIBUTES	Modify the type attributes
OPEN-LIBRARY	Open a library
PROVIDE-ELEMENT	Reserve members
REORGANIZE-LIBRARY	Reorganize a library
RESET-LMS-DEFAULTS	Reset the LMS defaults
RESET-LOGGING-PARAMETERS	Reset the LMS logging parameters
RESET-TYPE-ATTRIBUTES	Reset the type attributes
RETURN-ELEMENT	Cancel reservation of deleted members
SHOW-ELEMENT	List the contents of a member
SHOW-ELEMENT-ATTRIBUTES	List the directory of a library
SHOW-LIBRARY-ATTRIBUTES	List the library attributes
SHOW-LIBRARY-STATUS	Display the assigned libraries
SHOW-LMS-DEFAULTS	Output the current LMS default values
SHOW-LOGGING-PARAMETERS	Output the current LMS logging parameters
SHOW-STATISTICS	List statistics
SHOW-TYPE-ATTRIBUTES	List the type attributes
SHOW-USER-EXITS	Display the active user exits
WRITE-COMMENT	Write comments

ACTIVATE-USER-EXIT

This statement readies the user exits for the SHOW-ELEMENT or COMPARE-ELEMENT function.

ACTIVATE-USER-EXIT permits LMS to branch to a user routine prior to processing a member record.

Before LMS processes the member record, the following actions are possible:

– update the current member record

– insert records via the user routine

– exclude the current member record from processing

The user routine is informed of start and end of member so as to enable the user to insert records before the first and after the last member record.

For COMPARE-ELEMENT, two user exits are provided: one for the primary member and one for the secondary member. The user exits for SHOW-ELEMENT and COMPARE-ELEMENT can be defined simultaneously.

If several ACTIVATE-USER-EXIT statements with the same user exit are defined, the last one specified applies.

The activated user exits can be deactivated again by means of the DEACTIVATE-USER-EXIT statement.

The ACTIVATE-USER-EXIT statement requires specification of the function for which the user exit is to be activated, and of the entry point of the user program. All other parameters are preset, i.e. the user program is normally dynamically linked from the TASKLIB library.

ACTIVATE-**USER-EXIT**

FUNCTION = ***SHOW-ELEM**ENT / ***COMP**ARE-**ELEM**ENT(...)

***COMP**ARE-**ELEM**ENT(...)

ELEMENT = ***PRIM**ARY / ***SEC**ONDARY

,**ENTRY** = <name 1..8>

,**LIB**RARY = ***TASKLIB** / ***OMF** / <filename 1..54>

,**INTERF**ACE-**VERS**ION = ***V1** / ***V2**

FUNCTION = *SHOW-ELEMENT / *COMPARE-ELEMENT(...)
Defines the LMS statement for which the user exit is to be activated.

FUNCTION = *SHOW-ELEMENT
User exit for the SHOW-ELEMENT function. Valid for all text member types R, F, H, U and member types derived from them.

FUNCTION = *COMPARE-ELEMENT
User exit for the COMPARE-ELEMENT function. Valid for all text member types R, F, H, U and member types derived from them.

ELEMENT = *PRIMARY / *SECONDARY
If the user exit is activated for the COMPARE-ELEMENT statement, it is also necessary to specify whether the member is a primary member or a secondary member.

ENTRY = <name 1..8>
Name up to 8 characters in length of the entry point of the user routine. The name must not commence with the character string 'LMS'.

LIBRARY = *TASKLIB / *OMF / <filename 1..54>
Source containing the user module.

LIBRARY = *TASKLIB
The desired module is dynamically linked from the TASKLIB (see "Dynamically loading the user program" on page 160).

LIBRARY = *OMF
The desired module is dynamically linked from *OMF.

LIBRARY = <filename 1..54>
The desired module is dynamically linked from the specified library.

INTERFACE-VERSION = *V1 / *V2
Two interface versions exist. For reasons of compatibility the values BOE, REC and EOE are offered as standard. When specifying V2, the interface is expanded by the values C'BOS' (beginning of statement) and C'EOS' (end of statement). The user program is thus informed of the beginning and end of the statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Required access rights

The user program is linked in dynamically, which means that the access rights depend on the behavior of the BIND macro.

Dynamically loading the user program

The user program is always loaded into user-own class 6 memory. If the specification of a library is omitted under LIBRARY, the user program will first be sought in a private TASKLIB (assigned with /SET-TASKLIB LIBRARY=library), if present, and then in the system TASKLIB ($TASKLIB). If LIBRARY = *OMF or a file is specified but the user program is not found there, the user program is sought in the library assigned with LINK=BLSLIB or in the libraries assigned with LINK= BLSLIBnn ($00 \leq nn \leq 99$), with ascending number “nn”.

User exit interface

– Register conventions

When the user program is called, LMS will take account of the following register conventions:

Register 1: Address of a parameter list

Register 13: Address of the save area (18 words)

Register 14: Return address

Register 15: Address of entry point

– Structure of the parameter list

The parameter list consists of 5 words:

```
DC A(job description)
DC A(response description)
DC A(record to be transferred)
DC A(library name)
DC A(member designation)
```

The first two addresses are provided by LMS. This means that the user can only supply the response description to the address that is specified by LMS. The record address can be supplied by both LMS and the user.

1st word: `Job description`

The job description for the user subroutine consists of 3 bytes and can have the following contents:

C’BOE’	Beginning of member (element)
C’REC’	Record ready to be processed
C’EOE’	End of member (element)

If the expanded interface has been selected, the job description can have the following contents:

C’BOS’	Beginning of statement
C’BOE’	Beginning of member (element)
C’REC’	Record ready to be processed
C’EOE’	End of member (element)
C’EOS’	End of statement

2nd word: `Response description`

The response description issued by the user subroutine consists of 3 bytes and may have the following contents:

C’CON’	LMS processes the record whose address is in the parameter list on return from the user program, and then submits the next member record or EOE.
C’DEL’	LMS bypasses the last submitted member record and branches back to the user subroutine at the next member record or at EOE.
C’INS’	LMS first processes the record submitted by the user, and then returns to the user subroutine at the member record submitted previously or at EOE. This is repeated until the user responds with DEL or CON. This means that, if LMS submits record i, the records returned with INS are processed before record i.

The following job responses are possible:

Job	Response
BOS	Irrelevant
BOE	Irrelevant
REC	CON, DEL, INS
EOE	CON, INS
EOS	Irrelevant

If the response is incorrect, the LMS function is aborted and an error message is issued.

The diagram below illustrates the communication between LMS and the user subroutine.

3rd word: `Record`

This contains the address of the record that is passed by LMS to the user subroutine. When the user subroutine returns with the response CON or DEL, the same record address may be used. If the user wants to insert records, he must use a record buffer he has defined himself.
The records exchanged between LMS and the user subroutine start with a 4-byte record length field.
In order to list P-type members with SHOW-ELEMENT, the 5th byte must be a valid feed control character.

4th word: `Library name`

This word contains the address of the library name of the library currently being processed by LMS. The library name starts with a two-byte record length field.

5th word: `Member designation`

This word contains the address of the member designation of the member currently being processed by LMS. The member designation has the following format: two-byte record length field, followed by
`(type)membername/version[(variantnumber)]/date`

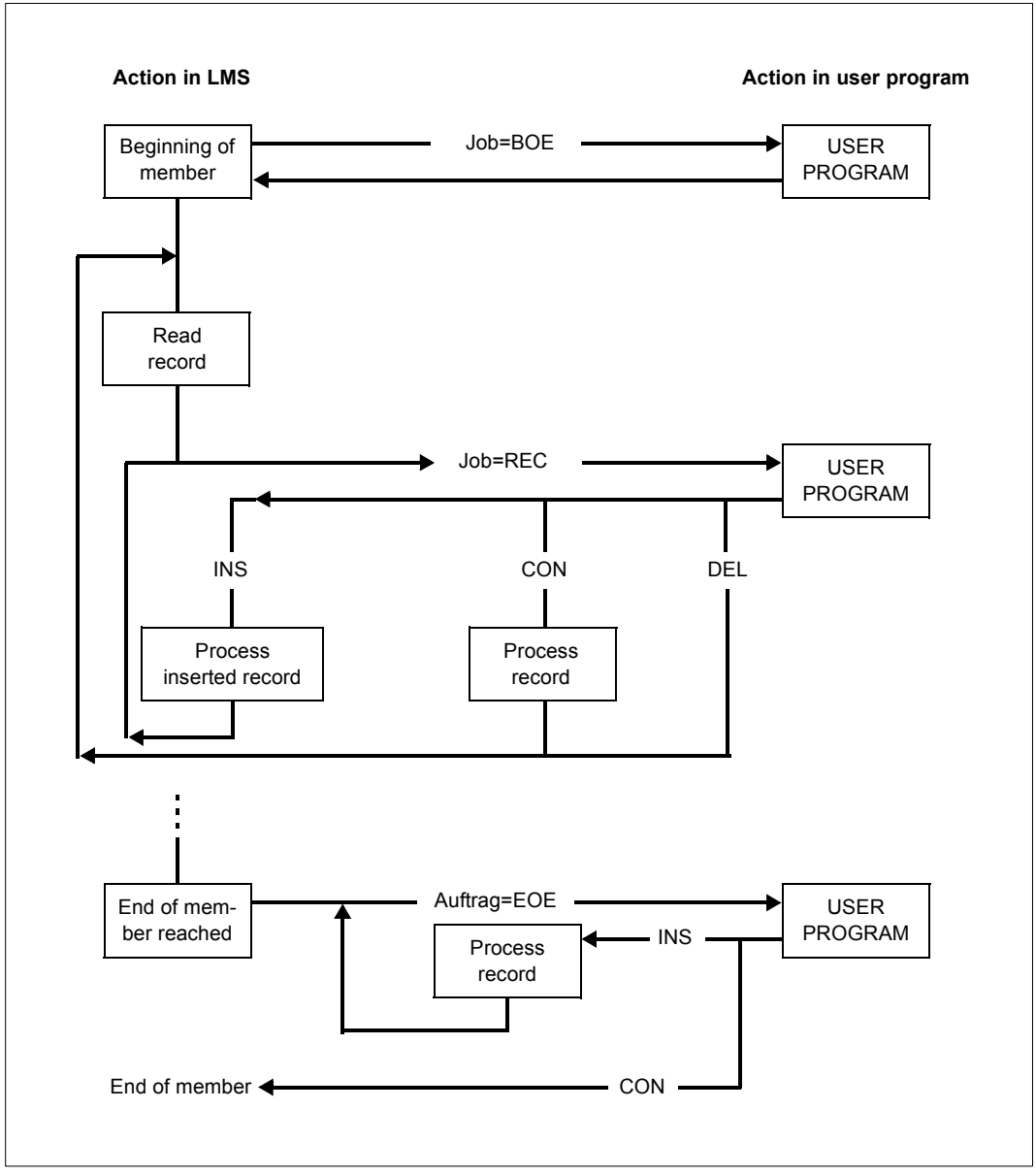


Figure 10: Communication between LMS and user programs

For an executable example involving a user exit, see page 483.

Examples

– Minimum specification for the statement ACTIVATE-USER-EXIT.
  The user program “module1” is readied as a user exit for the SHOW-ELEMENT function.

```
//activate-user-exit function=*show-element, entry=module1
```

– The desired user program is contained in the *OMF library.

```
//activate-user-exit function=*show-element, entry=module1,library=*omf
```

ADD-ELEMENT

ADD-ELEMENT adds files as non-delta members or delta members to a library. The member data is read from SYSDTA as standard. It can, however, also be read from an explicitly specified file or *OMF. However, if the statement is entered in the command line of EDT, the member data is automatically read from the current EDT work file.

The files are always added as a member to a library without a prefix, i.e. with catalog ID or user ID, unless the user has explicitly specified a prefix in the construction specification.

Files cataloged with RECORD-FORMAT=*UNDEFINED can also be incorporated in libraries. Files having RECORD-FORMAT=*FIXED can only be stored using SOURCE-ATTRIBUTES=*KEEP.
The record formats FIXED and UNDEFINED are converted into the VARIABLE record format, i.e. are given a 4-byte record header. The record length, including record header, must not exceed 32 Kbytes.
File generation groups can only be incorporated using link names and a valid LMS member designation.

In the case of the ADD-ELEMENT statement, LMS adopts the catalog attribute CCS of the file as a member attribute. If the data is read from SYSDTA, the member generated is given the CCS name set for SYSDTA as an attribute. If the data is read from *OMF, the member is assigned “no code”. If the data is read from an EDT work file, the member generated is given the currently set CCS name of EDT as an attribute.

ADD-ELEMENT

FROM-FILE = ***STD** / ***SYSDTA**(...) / ***ALL** / <filename 1..80 without-vers with-wild> / ***LINK**(...) / ***OMF**

***SYSDTA**(...)

END = ´***END**´ / <c-string 1..8>

***LINK**(...)

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

,**TO-E**LEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / ***LINK**(...) / <filename 1..54 without-vers>

***LINK**(...)

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

,**ELEM**ENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)

***BY-SOUR**CE(...)

VERSION = ***LMS-DEF**AULT / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..132 with-under with-wild-constr>(...)

VERSION = ***LMS-DEF**AULT / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / <alphanum-name 1..8>

,**USER-DATE** = ***TODAY** / <date 8..10 with-compl>

,**STOR**AGE-**FORM** = ***LMS-DEF**AULT / ***STD** / ***FULL** / ***DELTA**

,**ELEM**ENT-**ATTR**IBUTES = ***LMS-DEF**AULT / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

SOURCE-**ATTR**IBUTES = ***LMS-DEF**AULT / ***STD** / ***IGNORE** / ***KEEP**

,**DEL**ETE-**SOUR**CE = ***LMS-DEF**AULT / ***NO** / ***Y**ES

,**PROT**ECTION = ***LMS-DEF**AULT / ***STD** / ***BY-SOUR**CE

,**WR**ITE-**MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**

,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

FROM-FILE = *STD / *SYSDTA(...) / *ALL / <filename 1..80 without-vers with-wild>/ *LINK(..) / *OMF
Specifies the file to be added to the library as a member.

FROM-FILE = *STD
Data records are read from the default file, i.e. the system file SYSDTA. If the statement is entered from the EDT command line, however, the default file from which the data records are read is the current EDT work file.
Permissible member types:
– for non-delta members: S, M, P, J, D, X or types derived from them
– for delta members: analogous

FROM-FILE = *SYSDTA(...)
The records are read with RDATA from system file SYSDTA. The records must directly follow the ADD-ELEMENT statement.
Permissible member types:
– for non-delta members: S, M, P, J, D, X, R or types derived from them
– for delta members: S, M, P, J, D, X or types derived from them

END = '*END' / <c-string 1..8>
End criterion for the input. The sequence of records must be concluded with '*END' or a self-defined end criterion (see example, page 175). If the input data contains no end criterion, reading continues to EOF.

Note

If records are read from the system file SYSDTA, they must not begin with "/". The reason for this is that the RDATA macro interprets such records as commands and thus passes the return code for EOF. Therefore it is not possible to pass system commands as records.

FROM-FILE = *ALL
LMS attempts to incorporate all files of the ID into the library. If an error occurs in a file, this file is skipped and the process continues with the next one.

FROM-FILE = <filename 1..80 without-vers with-wild>
The data is read from the specified file.
Permissible member types:
– for non-delta members: S, M, P, J, D, X, R or types derived from them
– for delta members: S, M, P, J, D, X or types derived from them

Files of the PAM type can be stored only as non-delta members under the member type X or member types derived from it.

FROM-FILE = *LINK(...)
The data is read from the file specified via the link name.

LINK-NAME = <structured-name 1..8>
Link name referencing the file.

FROM-FILE = *OMF
Applies only to R-type members.
The data is read from the OMF file. All modules from the OMF file are incorporated. If the EAM area contains more than one module of the same name, LMS adds the last module processed to the library.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the destination and name under which the member is to be added.

LIBRARY = *STD / *LINK(...) / <filename 1..54 without-vers>
Specifies the library to which the member is to be added.

LIBRARY = *STD
The library opened globally by OPEN-LIBRARY.

LIBRARY = *LINK(...)
The library assigned via a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the file is to be added as a member.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wildcard-constr>(...)
Name that the new member to be added is to receive. A construction specification refers to the file name.

ELEMENT = *BY-SOURCE(...)
The member name corresponds to the file name, or to the module name in the case of *OMF.

VERSION = *LMS-DEFAULT / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version that the new member to be added is to receive.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, the highest version appropriate to BASE among the existing members of the same type and name is overwritten; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated (see also page 55).

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. If the BASE operand is entered in the form <composed-name 1..23>*, it will be interpreted as a prefix.
For further information on the effects of BASE, see page 50.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Specifies the name under which the member is stored.

VERSION = *LMS-DEFAULT / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version that the new member to be added is to receive.
For description of operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / <alphanum-name 1..8>
Type that the new member to be added is to receive.

USER-DATE = *TODAY / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *TODAY
The current date is given.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

STORAGE-FORM = *LMS-DEFAULT / *STD / *FULL / *DELTA
Storage form for the new member to be added. The storage form must not contradict the settings made by means of the MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES statements, and all members of a given type and name must have the same storage form.

STORAGE-FORM = *STD
The member is generated in accordance with the storage form required for the member scope. Contradictory requirements result in errors. If nothing special is required, full storage is selected.

STORAGE-FORM = *FULL
The new member is generated as a non-delta member (if this is not possible, an error message is issued).

STORAGE-FORM = DELTA
The member is generated as a delta member (if this is not possible, an error message is issued). This entry is permissible for member types: S, P, D, J, M, X and members types derived from them.

ELEMENT-ATTRIBUTES = *LMS-DEFAULT / *PARAMETERS(...)
Determines whether the file characteristics and also the ISAM key are incorporated into the output member.

ELEMENT-ATTRIBUTES = *PARAMETERS(...)

SOURCE-ATTRIBUTES = *LMS-DEFAULT / *STD/ *IGNORE / *KEEP
Stores file attributes. This operand has no effect if the data is read from SYSDTA, *OMF or an EDT work file. Original attributes are not stored.
If the data is read from a file of the type UPAM, this entry has no effect; it is always as though *KEEP had been specified.

SOURCE-ATTRIBUTES = *STD

No file attributes are stored. In the case of ISAM files, it is only possible to include in the member ISAM files using KEY-POSITION = 5, KEY-LENGTH <= 16 and RECORD-FORMAT = VARIABLE. A warning will be issued stating that the ISAM keys were not included.

SOURCE-ATTRIBUTES = *IGNORE
The same as for SOURCE-ATTRIBUTES = *STD, but no warning is issued.

SOURCE-ATTRIBUTES = *KEEP
The following file attributes are stored unchanged in the new member being added: ACCESS-METHOD, RECORD-FORMAT, RECORD-SIZE, BUFFER-LENGTH, PERFORMANCE, USAGE, ACCESS and USER-ACCESS. If ACCESS-METHOD=ISAM, LMS also stores the PADDING-FACTOR, LOGICAL-FLAG-LENGTH, VALUE-FLAG-LENGTH, PROPAGATE-VALUE-FLAG, the ISAM keys and information on ISAM secondary keys.

DELETE-SOURCE = *LMS-DEFAULT / *NO / *YES
Here, the user can specify whether the original file is to be retained or deleted. This operand has no effect if the data is read from SYSDTA, *OMF or an EDT work file.

DELETE-SOURCE=*NO
The original file will not be deleted.

DELETE-SOURCE=*YES
The original file will be deleted.

PROTECTION= *LMS-DEFAULT / *STD / *BY-SOURCE
Member protection for the member being added. This operand has no effect if the data is read from SYSDTA, *OMF or an EDT work file.

PROTECTION=*STD
If the member already exists, its member protection remains unchanged. If the member does not yet exist and initial member protection has been specified for the library and/or the member type, the member will receive that protection.

PROTECTION=*BY-SOURCE
The member is provided member protection according to the file protection attributes of the access mechanism activated for the file. ADD-ELEMENT is rejected with an error message if the file is protected by the access mechanism Access Control List (ACL).

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.
If the member to be stored is a delta member, it is necessary to ensure that the member is a leaf of the delta tree. Only leaves of a delta tree may be overwritten.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
A member will only be overwritten if a member having the same name is already present. Otherwise ADD-ELEMENT will be rejected with an error message.

WRITE-MODE = *EXTEND
A member will however only be extended if no ISAM keys are stored in the member and the file attributes stored in the member match the attributes of the file, except for the file name. Otherwise ADD-ELEMENT will be rejected with an error message.
EXTEND is not permitted for delta members or when input is from SYSDTA.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0064	GCCSN macro error; no CCS name specified
2	0	LMS0071	XHCS not loaded
2	0	LMS0095	Input records missing
2	0	LMS0102	Incomplete module in EAM file
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0301	File not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For FROM-FILE: read authorization for the file

If more than one file is affected by the statement, files without read authorization are excluded from the statement.

For TO-ELEMENT: read and write authorization for LIBRARY administer authorization where the specified member designation is new. Otherwise, only write authorization for the member existing under the specified member designation (administer authorization no longer required).

For STORAGE-FORM=*DELTA, read authorization must be granted for the member defined by BASE.

If WRITE-CONTROL is active and a base version exists, the USERID of the user must be entered as the HOLDER of the member specified by BASE. Only if write authorization has been granted can a new version be generated or this base version overwritten. In this case, administer authorization is no longer required.

If PROTECTION=*BY-SOURCE is specified, only the owner of the library can use this functionality. The library owner must also have ownership of the file which is to be added as a member.

Notes

– When adding a file to a library, it is the creation date of the member and not that of the file which applies.

– If SOURCE-ATTRIBUTES=*KEEP is specified, the following should be noted: Should any ISAM keys be present, this can impair subsequent processing such as language processing and /CALL-PROCEDURE. This parameter value is particularly suited to archiving.

– When creating a member, be sure to observe the convention applicable to the member type.

– When temporary files with wildcards are being added, no construction specification of the target member name is permitted, i.e. only ELEM = *BY-SOURCE is permitted.

– When temporary files are being added with ELEM = *BY-SOURCE, the member receives the internal file name. This member cannot be output to a file again under another task without explicit specification of a file name.

– If WRITE-CONTROL is active in the output library, the access method adds a comment (record type 2) to the member which is to be written. The comment logs the HOLDER, DATE and TIME of the process. The record is written as the first record of the record type. If, in addition, the member is written to the base of a different version (i.e. not the first version under a name), the member attributes STATE and HOLDER and all the rights of the base version are adopted for the new version. The CCSN is adopted from the source file. The USER-DATE is determined anew.

– If PROTECTION=*BY-SOURCE is specified, bear in mind the following:

  The BACL, GUARDS and standard access control mechanisms can include the data protection attributes of the access protection mechanism activated for a file in corresponding member proctection.

If the access protection mechanism activated for the file includes the access rights (read, write, execute), the member receives the following, corresponding protection mechanism:

Protection mechanism of file		Protection mechanism of member
Standard access control (ACCESS / USER-ACCESS)	→	Protection by BACL (without password)
BACL	→	Protection by BACL (without password)
GUARDS [1]	→	Protection by GUARD

[1] Special case:
If one of the access rights for the file has a value of NONE (no access possible) in the GUARDS protection mechanism, the corresponding right for the member is set to BACL protection with USER=NONE (no access possible).

Existing file passwords are not included in the corresponding member protection. When files are added to existing members, any existing member protection settings for the access rights (read, write, execute) are lost.

The hold right is handled in the same way as when PROTECTION=STD, i.e. if the member already exists, its member protection remains unchanged in regard to the hold right. If the member does not yet exist and initial protection regarding the hold right has been specified for the library or the member type, the member receives that setting.

Examples

– Adding a member

The member “testelem” is added to library LIB1 under the same name. The type specification must be specified explicitly here in the ADD-ELEMENT statement since the type is preset to *NONE as standard.

```
/show-file-attributes
TESTELEM
.
//start-lms
//open-library lib1,*update
//add-element from-file=testelem,to-elem=*lib(type=d)
.
.
```

– Reading from SYSDTA and definition of the end criterion

```
/start-lms ——————————————————————————————————————————————— (1)
//open-library lib1,*update ———————————————————————————————— (2)
//modify-lms-defaults (type=d) ————————————————————————————— (3)
//add-element *sysdta(end=c'stop'),to-element=*lib(element=letter.a) ———— (4)
* Dear... ———————————————————————————————————————————————— (5)
.
.
.
*STOP ————————————————————————————————————————————————————— (6)
//end ——————————————————————————————————————————————————————— (7)
```

(1) LMS is started.

(2) The library with the name lib1 is opened for reading and writing.

(3) The LMS default value *NONE for the type specification is changed to “D” in the member type specification. This setting applies to the entire LMS run unless a new MODIFY-LMS-DEFAULTS statement affecting this member type is issued or the type is changed locally in a statement.

(4) Records are to be read in from SYSDTA, where the word STOP is defined as the end criterion. The records are to be stored under the member name letter.a in the library lib1 opened by OPEN-LIBRARY. The member type need no longer be specified.

(5) Text input. The text ’Dear ...’ is stored exactly as keyed.

(6) The addition of records is terminated by specifying stop.

(7) LMS is terminated.

BEGIN-MAKE

The BEGIN-MAKE statement initiates a sequence of make substatements. A sequence of make substatements is concluded by the END-MAKE statement.

The BEGIN-MAKE statement can be used to specify the following:

– the target which is to be generated with the make run,

– the type of make processing (required) or all actions to be performed and

– the continuation processing.

If a procedure is being executed at the same time, LMS terminates it.

BEGIN-MAKE

TARGET = ***FIRST-TARGET** / <filename 1..54 without-vers> / ***LIB**RARY**-ELEM**ENT(...)

***LIB**RARY**-ELEM**ENT(...)

LIBRARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = <composed-name 1..64 with-under>(...)

<composed-name 1..64 with-under>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

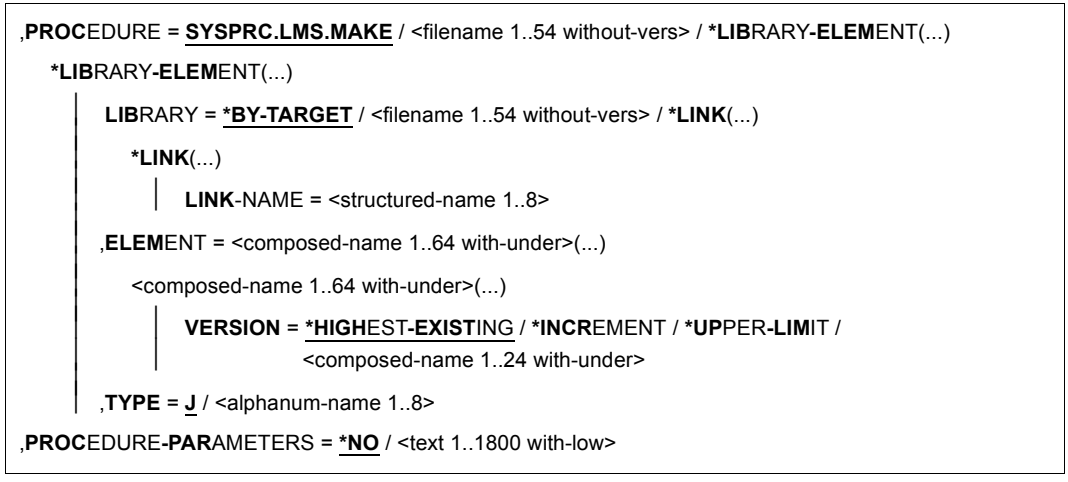
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = <alphanum-name 1..8>

,**SEL**ECT = ***MOD**IFIED / ***ALL**

,**SUCC**ESS-**PROC**ESSING = ***INCL**UDE-**PROC**EDURE / ***CALL-PROC**EDURE / ***ENTER-PROC**EDURE / ***CRE**ATE-**PROC**EDURE / ***TOUCH**

(part 1 of 2)



(part 2 of 2)

TARGET = *FIRST-TARGET / <filename 1..54 without-vers> / *LIBRARY-ELEMENT(...)
The target to be generated.

TARGET = *FIRST-TARGET
The first target (library member or file) among the dependency definitions is the target of the entire make run.

TARGET = <filename 1..54 without-vers>
The target is a file. The initial value for the default library in MODIFY-MAKE-DEFAULTS is *NONE.

TARGET = *LIBRARY-ELEMENT(...)
The target is located in a library member. A target library specified here is the initial value for the default library MODIFY-MAKE-DEFAULTS.

LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Library of the target member.

LIBRARY = <filename 1..54 without-vers>
Library of the target member.

LIBRARY = *LINK(...)
Library of the target member.

LINK-NAME = <structured-name 1..8>
Link name of the library of the target member.

ELEMENT = <composed-name 1..64 with-under>(...)
Name of the target member.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the target member.

VERSION = *HIGHEST-EXISTING
The highest existing version is taken as the target.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is taken as the target.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as a version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Specifies the base of the target member.

TYPE = <alphanum-name 1..8>
Type of the target member.

SELECT = *MODIFIED / *ALL
Components which are used for the generation of targets.

SELECT = *MODIFIED
Modified components (=components of more recent date) are to be used.

SELECT = *ALL
All components are to be used. It is simulated that all components that lead to the target are not current.

SUCCESS-PROCESSING = *INCLUDE-PROCEDURE / *CALL-PROCEDURE / *ENTER-PROCEDURE / *CREATE-PROCEDURE / *TOUCH
Type of continuation processing. At the end of the make run (END-MAKE statement), the processing specified here is initiated. If the target in TARGET is current, the continuation processing is not initiated.

SUCCESS-PROCESSING = *INCLUDE-PROCEDURE
The procedure specified for PROCEDURE is generated and executed synchronously, using the current variables.

SUCCESS-PROCESSING = *CALL-PROCEDURE
The procedure specified for PROCEDURE is generated and executed synchronously, using a new variable environment.

SUCCESS-PROCESSING = *ENTER-PROCEDURE
The procedure specified for PROCEDURE is generated and executed asynchronously, using a new variable environment.

SUCCESS-PROCESSING = *CREATE-PROCEDURE
The procedure is merely generated.

SUCCESS-PROCESSING = *TOUCH
Touches the components of the programming system which have to be generated, i.e. gives each of them a new modification time as a time stamp. The entire program system, up to and including TARGET, is then current. No procedure is generated. Files which are to be handled with *TOUCH must be neither empty nor protected against UPAM accesses.

PROCEDURE = SYSPRC.LMS.MAKE / <filename 1..54 without-vers> / *LIBRARY-ELEMENT(...)
Procedure that is generated by make and may be started when make terminates.

PROCEDURE = *LIBRARY-ELEMENT(...)
The procedure file that is generated is a member of a library.

LIBRARY = *BY-TARGET / <filename 1..54 without-vers> / *LINK(...)
Library in which the procedure is stored.

LIBRARY = *BY-TARGET
The library is that of TARGET.

LIBRARY = <filename 1..54 without-vers>
Library of the procedure member.

LIBRARY = *LINK(...)
Library of the procedure member.

LINK-NAME = <structured-name 1..8>
Link name of the library of the procedure member.

ELEMENT = <composed-name 1..64 with-under>(...)
Name of the procedure member.

VERSION = *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the procedure member.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, the highest version appropriate to BASE among the existing members of the same type and name is overwritten; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as a version designation.

TYPE = J / <alphanum-name 1..8>
Type of the procedure member.

PROCEDURE-PARAMETERS = *NO / <text 1..1800 with-low>
Parameters supplied to the procedure when it is called. These parameters correspond to the PROCEDURE-PARAMETERS operands of CALL-PROCEDURE, ENTER-PROCEDURE and INCLUDE-PROCEDURE, and are passed along unchanged.

PROCEDURE-PARAMETERS = *NO
No procedure parameters.

PROCEDURE-PARAMETERS = <text 1..1800 with-low>
Procedure parameters which are to be passed unchanged.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Note

A generated procedure member which is not of the highest version can be executed only if SDF-P is available in the system (and only with CALL-PROCEDURE and INCLUDE-PROCEDURE).

Example

The part of a procedure shown below generates the target from the parameter TARGET with the aid of the definitions in the file MAKEFILE if the target is not current. The make substatements are located in the MAKEFILE file. The procedure, which is generated by default, is then executed, and LMS is then terminated.

```
...
/START-LMS
/BEGIN-BLOCK DATA-INSERTION=YES
//BEGIN-MAKE TARGET=&(TARGET),       "MAKE CALL"
//MODIFY-MAKE-DEFAULTS LIBRARY=MYLIB "DEFAULT SETTINGS"
/INCLUDE-PROCEDURE NAME=MAKEFILE     "CALL OF MAKEFILE"
/END-BLOCK
//END-MAKE                           "END OF MAKE"
...
```

The framework of the procedure and the file MAKEFILE are shown in the example of make functionality on page 498.

make substatements

Generally, the make substatements result in a procedure containing all the actions required for the generation of the specified target.

Overview of make substatements

END-MAKE	Concludes the make substatements
MODIFY-MAKE-DEFAULTS	Specifies global parameters
SET-DEPENDENCY	Specifies dependencies between components
SET-POSTPROCESSING	Specifies actions to be performed before the generated procedure is executed
SET-PREPROCESSING	Specifies actions to be performed after the generated procedure is executed
SET-STD-ACTION	Specifies standard actions which process one member type to generate a different one

Note

Standard SDF statements may also be used as make substatements.

END-MAKE

END-MAKE signifies the end of the sequence of make substatements. The procedure generated by the make substatements is not actually processed until after the END-MAKE statement occurs.

END-MAKE

If, in BEGIN-MAKE, synchronous execution of the generated procedure was specified for SUCCESS-PROCESSING (*INCLUDE-PROCEDURE, *CALL-PROCEDURE), LMS is terminated.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0712	Touch not possible
2	0	LMS0714	Touch not possible on empty file
2	0	LMS0721	The specified target is already current
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

MODIFY-MAKE-DEFAULTS

The MODIFY-MAKE-DEFAULTS statement is used for global control of the make run. Values can be changed again and again with MODIFY-MAKE-DEFAULTS. The modified values are valid for the subsequent statements.

The MODIFY-MAKE-DEFAULTS operands . . . -VAR define the make S variables, which are set anew before each action.

The variables are declared through LMS. User declarations in the preprocessing section are not overwritten.

```
MODIFY-MAKE-DEFAULTS

 LIBRARY = *UNCHANGED / *NONE / <filename 1..54 without-vers> / *LINK(...)
    *LINK(...)
       | LINK-NAME = <structured-name 1..8>
,CURRENT-TARGET-VAR = *UNCHANGED / *NONE / <structured-name 1..20>
,FROM-OBJECTS-VAR = *UNCHANGED / *NONE / <structured-name 1..20>
,MODIFIED-OBJECTS-VAR = *UNCHANGED / *NONE / <structured-name 1..20>
,SUPPRESS-ERRORS = *UNCHANGED / *NONE / *ALL
```

LIBRARY = *UNCHANGED / *NONE / <filename 1..54 without-vers> / *LINK(...)
Default library.
Library which is to be used for *MAKE-DEFAULT for library members. The initial value is the library that was specified for TARGET in the BEGIN-MAKE statement (when the target is a file, the value is *NONE).

LIBRARY = *NONE
No default library. Use of the default library in subsequent statements results in errors.

LIBRARY = <filename 1..54 without-vers>
Specification of a default library.

LIBRARY = *LINK(...)
Specification of a default library.

LINK-NAME = <structured-name 1..8>
Link name of the default library.

CURRENT-TARGET-VAR = *UNCHANGED / *NONE / <structured-name 1..20>
S structure variable to which the current target is assigned.

CURRENT-TARGET-VAR = *NONE
The current target is not to be assigned to an S structure variable.

CURRENT-TARGET-VAR = <structured-name 1..20>
The current target is to be assigned to the specified S structure variable.

FROM-OBJECTS-VAR = *UNCHANGED / *NONE / <structured-name 1..20>
S list variable to which all the source components for the current target are assigned.

FROM-OBJECTS-VAR = *NONE
The source components are not to be assigned to an S list variable.

FROM-OBJECTS-VAR = <structured-name 1..20>
The source components are to be assigned to the specified S list variable.

MODIFIED-OBJECTS-VAR = *UNCHANGED / *NONE / <structured-name 1..20>
S list variable to which newer source components are assigned. These are all of the source components which are newer than the current target or all of the components if BEGIN-MAKE SELECT=*ALL was specified.

MODIFIED-OBJECTS-VAR = *NONE
The newer components are not to be assigned to an S list variable.

MODIFIED-OBJECTS-VAR = <structured-name 1..20>
The newer source components are to be assigned to the specified S list variable.

SUPPRESS-ERRORS = *UNCHANGED / *NONE / *ALL
Default for error suppression during actions.

SUPPRESS-ERRORS = *NONE
Errors result in the generated procedure being aborted (spin-off).

SUPPRESS-ERRORS = *ALL
Errors occurring during the action are suppressed, i.e. do not result in the generated procedure being aborted. In the case of lists of actions, errors are suppressed only at the end. The procedure resumes after the action(s) related to that dependency.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Example

The make S variables and the standard library are to be redefined for the rest of the make run. The variable CURT is in each case to contain the target processed, and the variable ALLOBJ all of the source components.

```
//MODIFY-MAKE-DEFAULTS LIBRARY =BSPLIB,-
  CURRENT-TARGET-VAR=CURT,FROM-OBJECTS-VAR=ALLOBJ
```

SET-DEPENDENCY

The SET-DEPENDENCY statement is used to define the dependencies between objects.

```
SET-DEPENDENCY

 TARGET-OBJECT = *VARIABLE(...) / list-poss(2000): *LIBRARY-ELEMENT(...) /
                      <filename 1..54 without-vers with-wild(80)>
    *VARIABLE(...)
        VARIABLE-NAME = <composed-name 1..255>
    *LIBRARY-ELEMENT(...)
        LIBRARY = *MAKE-DEFAULT / <filename 1..54 without-vers> / *LINK(...)
            *LINK(...)
                LINK-NAME = <structured-name 1..8>
        ,ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
            <composed-name 1..64 with-under with-wild(132)>(...)
                VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
                ,BASE = *STD / <composed-name 1..24 with-under with-wild>
        ,TYPE = <alphanum-name 1..8 with-wild(20)>
,FROM-OBJECT = *NONE / *VARIABLE(...) / list-poss(2000): *LIBRARY-ELEMENT(...) /
                    <filename 1..54 without-vers with-wild-constr>
    *VARIABLE(...)
        NAME = <composed-name 1..255>
    *LIBRARY-ELEMENT(...)
        LIBRARY = *MAKE-DEFAULT / <filename 1..54 without-vers> / *LINK(...)
            *LINK(...)
                LINK-NAME = <structured-name 1..8>
        ,ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
            <composed-name 1..132 with-under with-wildcard-constr>(...)
                VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
                ,BASE = *STD / <composed-name 1..24 with-under with-wild>
        ,TYPE = <alphanum-name 1..20 with-wild-constr>
,ACTION = *STD / list-poss(2000): <c-string 1..1800 with-low>
,SUPPRESS-ERRORS = *MAKE-DEFAULT / *NONE / *ALL
```

TARGET-OBJECT = *VARIABLE(...) / list-poss(2000): *LIBRARY-ELEMENT(...) / <filename 1..54 without-vers with-wild(80)>
Target component.
Target of the dependency definition. Wildcards are corresponding to the specifications for the FROM-OBJECT operand (see also page 133, “Selection specifications in make”).

TARGET-OBJECT = *VARIABLE (...)
The target objects are listed in an S list variable.
The individual list members must be available in the format of LMS. The version *HIGHEST-EXISTING is represented by *HIGH-EXIST.

VARIABLE-NAME = <composed-name 1..255>
Name of the S list variable.

TARGET-OBJECT = list-poss(2000): *LIBRARY-ELEMENT(...)
The target component is a library member and must be entered explicitly.

LIBRARY = *MAKE-DEFAULT / <filename 1..54 without-vers> / *LINK(...)
Library of the target member.

LIBRARY = *MAKE-DEFAULT
The library defined as the default by means of the make substatement MODIFY-MAKE-DEFAULTS.

LIBRARY = <filename 1..54 without-vers>
Library of the target member.

LIBRARY = *LINK(...)
Library of the target member.

LINK-NAME = <structured-name 1..8>
Link name of the library of the target member.

ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
Name of the target member (for wildcards, see page 133, “Selection specifications in make”).

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the target member.

VERSION = *HIGHEST-EXISTING
The highest existing version is overwritten. If no such version exists, LMS generates an analogous default version.

VERSION = *UPPER-LIMIT
The highest possible version X’FF’ is generated.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member.

TYPE = <alphanum-name 1..8 with-wild(20)>
Type of the target member.

TARGET-OBJECT = <filename 1..54 without-vers with-wild(80)>
Target of the dependency definition (for wildcards, see page 133, “Selection specifications in make”).

FROM-OBJECT = *NONE / *VARIABLE(...) / list-poss(2000): *LIBRARY-ELEMENT(...) / <filename 1..54 without-vers with-wild-constr>
Source component of the dependency definition. The wildcards are corresponding to the specifications for the TARGET-OBJECT operand (for wildcards, see page 133, “Construction specifications in make”).

FROM-OBJECT = *NONE
There are no source components. The target component is never current.

FROM-OBJECT = *VARIABLE (...)
The required source objects are listed in an S list variable. The individual list members must be available in the LMS format. The version *HIGHEST-EXISTING is represented by *HIGH-EXIST.

NAME = <composed-name 1..255>
Name of the S list variable.

FROM-OBJECT = list-poss(2000): *LIBRARY-ELEMENT(...)
The source components are library members.

LIBRARY = *MAKE-DEFAULT / <filename 1..54 without-vers> / *LINK(...)
Library of the source member.

LIBRARY = *MAKE-DEFAULT
The library defined as the default by means of the make substatement MODIFY-MAKE-DEFAULTS.

LIBRARY = <filename 1..54 without-vers>
Library of the source member.

LIBRARY = *LINK(...)
Library of the source member.

LINK-NAME = <structured-name 1..8>
Link name of the library of the source member.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Name of the source member ((for wildcards, see page 133, “Construction specifications in make”).

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the source member.

VERSION = *HIGHEST-EXISTING
The highest existing version is selected.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is selected.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the source member.

TYPE = <alphanum-name 1..20 with-wild-constr>
Type of the source member.

FROM-OBJECT = list-poss(2000): <filename 1..54 without-vers with-wild-constr>
Source component of the dependency definition (for wildcards, see page 133, "Construction specifications in make").

ACTION = *STD / list-poss(2000): <c-string 1..1800 with-low>
Action performed in order to generate the target component.

ACTION = *STD
Executes the action specified in the make substatement SET-STD-ACTION for the combination of the source and the target types. If lists are specified for TARGET-OBJECT or FROM-OBJECT, the type of the first member in the list is taken in each case to determine the standard action. No standard actions can be specified for files.

ACTION = list-poss(2000): <c-string 1..1800 with-low>
One or more actions that generate the target components from the source components. Each action has its own line in the procedure generated by BEGIN-MAKE.

SUPPRESS-ERRORS = *MAKE-DEFAULT / *NONE / *ALL
Suppression of errors occurring during the execution of actions.

SUPPRESS-ERRORS = *MAKE-DEFAULT
The default for error suppression set by means of MODIFY-MAKE-DEFAULTS.

SUPPRESS-ERRORS = *NONE
Errors result in the generated procedure being aborted (spin-off).

SUPPRESS-ERRORS = *ALL
Errors occurring during execution of the action are suppressed, i.e. do not result in the generated procedure being aborted. In the case of lists of actions, errors are suppressed only at the end. The procedure resumes after the action(s) related to that dependency.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Notes

– The first target of the first dependency definition can be specified as the overall target of the make run (presetting) under the name *FIRST-TARGET in the BEGIN-MAKE statement.

– At the end of the action, no program is to be loaded.

Example

The phase (load module) PROG in the standard library consists of the object modules PART1 and PART2. PROG is generated using the procedure *LIB(BSPLIB,LINK). The object modules (member type R) are also generated from sources of the same names (type S) using the procedure *LIB(BSPLIB,COMPILE). The selection specification in make *LIB(,*,R) acts on the type-R members present in the program system, i.e. PART1 and PART2. The modules are to be generated from sources of the same names. The variable ALLOBJ contains all of the source components, and the variable CURT (short for “current target”) in each case the target produced.

```
//SET-DEPENDENCY TARGET-OBJECT=PROG, -
//   FROM-OBJECT=(*LIB(,TEIL1,R),*LIB(,TEIL2,R)), -
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,LINK),(OBJVAR=ALLOBJ)'
//SET-DEPENDENCY -
//   TARGET-OBJECT=*LIB(,*,R),-
//   FROM-OBJECT=(*LIB(,*,S),*LIB(,GLOBALDATA,M),*LIB(,HEADERS,M)),-
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,COMPILE),(&&(CURT.ELEM))'
```

In abbreviated form similar to the UNIX make:

```
//SET-DEPENDENCY PROG, (*LIB(,TEIL1,R),*LIB(,TEIL2,R)), -
//   ´/CALL-PROCEDURE *LIB(BSPLIB,LINK),(OBJVAR=ALLOBJ)´
//SET-DEPENDENCY *LIB(,*,R), *LIB(,*,S),-
//   ´/CALL-PROCEDURE *LIB(BSPLIB,COMPILE),(&&(CURT.ELEM))´
```

SET-POSTPROCESSING

The SET-POSTPROCESSING statement is used to specify the actions that conclude the generated procedure. The SET-POSTPROCESSING statement must occur only once in any sequence of make statements.

SET-POSTPROCESSING
ACTION = list-poss(2000): <c-string 1..1800 with-low>

ACTION = list-poss(2000): <c-string 1..1800 with-low>
The specified actions are incorporated as the final actions in the generated procedure.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Note

The actions are not executed if the target of the make run is current.

Example

The command /CALL-PROCEDURE *LIB(BSPLIB,STOP) is to be executed at the end of the generated procedure.

```
//SET-POSTPROCESSING ACTION='/CALL-PROCEDURE *LIB(BSPLIB,STOP)'
```

SET-PREPROCESSING

The SET-PREPROCESSING statement is used to specify the actions that are to be executed at the beginning of the generated procedure. The SET-PREPROCESSING statement must occur only once in any sequence of make statements.

SET-PREPROCESSING
ACTION = list-poss(2000): <c-string 1..1800 with-low> **SUPPRESS-ERR**ORS = ***MAKE-DEF**AULT / ***NONE** / ***ALL**

ACTION = list-poss(2000): <c-string 1..1800 with-low>
The specified actions are incorporated as the first actions in the generated procedure.

SUPPRESS-ERRORS = *MAKE-DEFAULT / *NONE / *ALL
Default for error suppression during actions.

SUPPRESS-ERRORS = *MAKE-DEFAULT
The default for error suppression set by means of MODIFY-MAKE-DEFAULTS.

SUPPRESS-ERRORS = *NONE
Errors result in the generated procedure being aborted (spin-off).

SUPPRESS-ERRORS = *ALL
Errors occurring during execution of the action are suppressed, i.e. do not result in the generated procedure being aborted. In the case of lists of actions, errors are suppressed only at the end. The procedure resumes after the action(s) related to that dependency.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Note

The actions are not executed if the target of the make run is current.

Example

The command /CALL-PROCEDURE *LIB(BSPLIB,INIT) is to be executed at the beginning of the generated procedure.

```
//SET-PREPROCESSING ACTION='/CALL-PROCEDURE *LIB(BSPLIB,INIT)'
```

SET-STD-ACTION

The SET-STD-ACTION statement is used to specify standard actions which process source components of one type to generate a target components of another type. These standard actions can then be referenced in the SET-DEPENDENCY statement.

The SET-STD-ACTION statement has effect only for dependencies between library members and must be specified only once for each combination of source type and target type. Files cannot be assigned to standard actions because they have no type.

SET-STD-ACTION
TARGET-TYPE = <alphanum-name 1..8> ,**FROM-TYPE** = <alphanum-name 1..8> ,**ACTION** = list-poss(2000): <c-string 1..1800 with-low>

TARGET-TYPE = <alphanum-name 1..8>
Type of the target component.

FROM-TYPE = <alphanum-name 1..8>
Type of the source component.

ACTION = list-poss(2000): <c-string 1..1800 with-low>
Standard action for generating components of the target type from components of the source type.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Note

No program should be loaded at the end of the action.

Example

A SET-DEPENDENCY statement with no explicit action is to process type-S members to generate type-R members with the aid of the specified standard actions.

```
//SET-STD-ACTION -
//   TARGET-TYPE=R,FROM-TYPE=S,-
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,COMPILE,(&&(CURT.ELEM))'
//SET-DEPENDENCY -
//   TARGET-OBJECT=*LIB(,*,R),FROM-OBJECT=*LIB(,*,S)
```

In abbreviated form similar to the UNIX make (//SET-STD-ACTION is equivalent to using the suffixes of the UNIX make):

```
//SET-STD-ACTION R,S,-
//   '/CALL-PROCEDURE *LIB(BSPLIB,COMPILE),(&&(CURT.ELEM))'
//SET-DEPENDENCY *LIB(,*,R), *LIB(,*,S)
```

CALL-EDT

The CALL-EDT statement calls up the editor EDT and opens work file 0.

LMS supports EDT versions as of V16.2A (see [10]).

For the editing of Unicode members EDT V17.0 or higher is necessary (see [11]).

Terminating EDT:

EDT can be terminated with RETURN or HALT, and the EDT data (files in virtual memory, variables, etc.) will be retained. Only the occurrence of a serious EDT error will cause this data to be lost and an LMS message to be issued.

```
CALL-EDT

 EDT-MODE = *LMS-DEFAULT / *COMPATIBLE / *UNICODE
,EDITOR-COMMANDS = *NONE / <c-string 1..251> / *LOWER-CASE(...)
    *LOWER-CASE(...)
      | EDITOR-COMMANDS = <c-string 1..251 with-low>
```

EDT-MODE = *LMS-DEFAULT / *COMPATIBLE / *UNICODE
Specifies the mode that EDT is to be called in.

EDT-MODE = *COMPATIBLE
EDT is called in compatibility mode.

EDT-MODE = *UNICODE
EDT is called in Unicode mode.

EDITOR-COMMANDS = *NONE / <c-string 1..251> / *LOWER-CASE(...)
No sequence of editor commands is specified.

EDITOR-COMMANDS = <c-string 1..251>
Sequence of editor commands which are to be executed after EDT is called. In the entry, the commands must be separated from one another by a semicolon (;). With the exception of EDIT and RETURN, it is possible to specify any commands which are permitted both in F mode and in L mode of EDT. The EDIT command is permitted only in the form EDIT ONLY. It should be noted that the HALT command in a sequence of EDT commands causes EDT to terminate and so also causes the EDT data to be released (files in virtual memory, variables, etc.).
Lowercase letters are converted to uppercase.

EDITOR-COMMANDS = *LOWER-CASE(...)
Lowercase letters are not converted to uppercase.

EDITOR-COMMANDS = <c-string 1..251 with-low>
Sequence of editor commands as described above, except that lowercase letters are not converted to uppercase.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0297	Change of operation mode not possible
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Notes

After CALL-EDT, files or members can be opened and read into EDT work areas with the EDT OPEN F= or OPEN L= statement. These objects also remain locked after a return is made to LMS via HALT or RETURN. The OPEN indicator is only reset when LMS is terminated. However, if the CLOSE command is issued before HALT, the member or file is written and closed. Another task can then access it without terminating LMS.

If an EDT-MODE is specified explicitly in the CALL-EDT statement, but EDT cannot start in the specified mode, the statement is aborted and the message LMS0297 is output.

If no EDT-MODE is specified explicitly in the CALL-EDT statement, EDT starts in the default EDT mode, but can switch modes by itself if necessary. The default EDT mode is COMPATIBLE at the beginning of the LMS run and can be changed by the MODIFY-LMS-DEFAULTS statement.

CLOSE-LIBRARY

This statement closes the specified library/libraries.

If the LMS output is held in a library member, then the associated library will not be closed and an error message is issued.

If this statement is specified without parameters, all open libraries are closed.

```
CLOSE-LIBRARY

LIBRARY = *ALL / *STD / <filename 1..54 without-vers> / *LINK(...)
  *LINK(...)
    | LINK-NAME = <structured-name 1..8>
```

LIBRARY = *ALL / *STD/ <filename 1..54 without-vers> / *LINK(...)
Specifies the library or libraries to be closed.

LIBRARY = *ALL
All open libraries are closed.

LIBRARY = *STD
The library opened by OPEN-LIBRARY is closed.

LIBRARY = <filename 1..54 without-vers>
Name of the library to be closed.

LIBRARY = *LINK(...)
The library assigned via a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to the LMS run.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
1	0	LMS0036	Library not assigned
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Examples

– All open libraries are closed.

```
//CLOSE-LIBRARY
```

– The library that was assigned via the link name lib2 is closed.

```
//CLOSE-LIBRARY LIBRARY=*LINK(LINK-NAME=LIB2)
```

COMPARE-ELEMENT

COMPARE-ELEMENT permits text members to be compared one record at a time, where the scope of the comparison operation can be defined with the RECORD-PART operand. The differences established are listed in a comparison log and in the comparison statistics. The members may be located in different libraries.

The COMPARE-ELEMENT statement is also executed even if only one of the comparison members is found in the specified libraries. This allows the counting of records in members.

If two members are compared with one another, LMS uses the terms primary member and secondary member. The user is free to select the new or the old member as the base member. LMS always considers the secondary member to be the base for the comparison. This means that LMS identifies missing records in the secondary member as inserted records and missing records in the primary member as deleted records.

The primary and secondary member base types may differ if text members are compared.

The COMPARE-PARAMETERS operand is used to define the type of comparison (formal or logical) and control logging (scope and format).

COMPARE-ELEMENT always produces comparison statistics in the internal memory C0. After execution of the statement, C0 is added to memory C1. Memory C0 is reinitialized before each COMPARE-ELEMENT statement. These memories can be output by means of the SHOW-STATISTICS statement.

ACTIVATE-USER-EXIT permits the member records to be accessed via a user program prior to the actual comparison.

COMPARE-**ELEM**ENT

PRIMARY-**ELEM**ENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

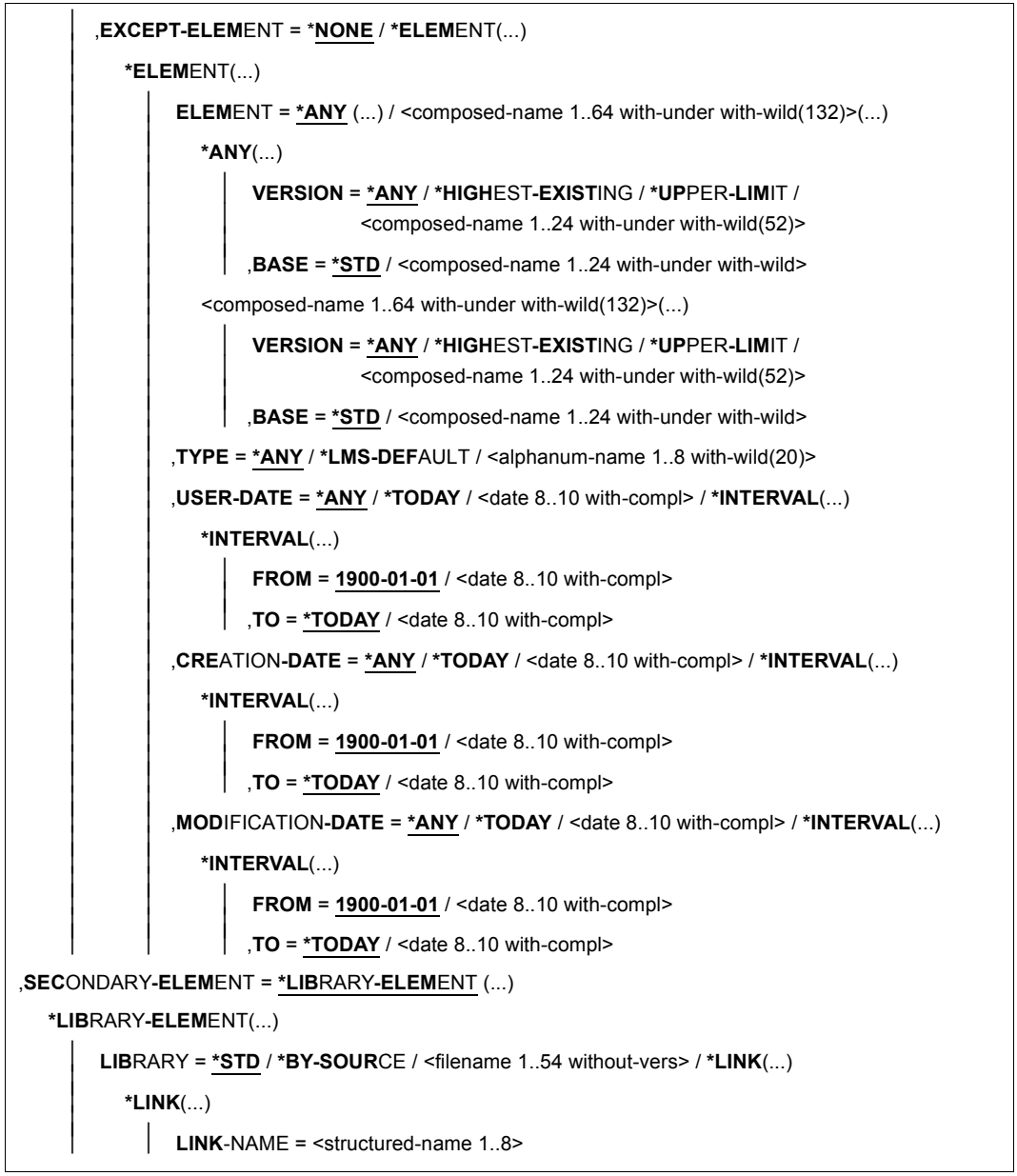
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 1 of 3)


,EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
*ELEMENT(...)
ELEMENT = *ANY (...) / <composed-name 1..64 with-under with-wild(132)>(...)
*ANY(...)
VERSION = *ANY / *HIGHEST-EXISTING / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
,BASE = *STD / <composed-name 1..24 with-under with-wild>
<composed-name 1..64 with-under with-wild(132)>(...)
VERSION = *ANY / *HIGHEST-EXISTING / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
,BASE = *STD / <composed-name 1..24 with-under with-wild>
,TYPE = *ANY / *LMS-DEFAULT / <alphanum-name 1..8 with-wild(20)>
,USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>
,CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>
,MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>
,SECONDARY-ELEMENT = *LIBRARY-ELEMENT (...)
*LIBRARY-ELEMENT(...)
LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
*LINK(...)
LINK-NAME = <structured-name 1..8>

(part 2 of 3)

,**ELEM**ENT = ***BY-SOUR**CE (...) / ***ALL**(...) /
<composed-name 1..132 with-under with-wild-constr>(...)

***BY-SOUR**CE(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***BY-SOUR**CE / ***ALL** / ***UP**PER-**LIM**IT /
<composed-name 1..52 with-under with-wild-constr>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

***ALL**(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***BY-SOUR**CE / ***ALL** / ***UP**PER-**LIM**IT /
<composed-name 1..52 with-under with-wild-constr>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..132 with-under with-wild-constr>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***BY-SOUR**CE / ***ALL** / ***UP**PER-**LIM**IT /
<composed-name 1..52 with-under with-wild-constr>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / ***ALL** / <alphanum-name 1..20 with-wild-constr>

,**COMP**ARE-**PAR**AMETERS = ***LMS-DEF**AULT / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

RECORD-**PART** = ***LMS-DEF**AULT / ***ALL** / ***PART**(...)

***PART**(...)

START = ***LMS-DEF**AULT / <integer 1..32764>

,**LEN**GTH = ***LMS-DEF**AULT / ***REST** / <integer 1..32764>

,**SPACE**S = ***LMS-DEF**AULT / ***STD** / ***IGNORED** / ***RELEV**ANT

,**INF**ORMATION = ***LMS-DEF**AULT / ***MED**IUM / ***MIN**IMUM / ***MAX**IMUM / ***SUMM**ARY / ***STATIS**TICS
/ ***NONE**

,**LAYOUT** = ***LMS-DEF**AULT / ***COMPATIBLE** / ***COMPR**ESSED

,**JOIN-ELEM**ENT-**SET**S = ***LMS-DEF**AULT / ***NO** / ***Y**ES

,**TEXT-OUT**PUT = ***LOGGING-PAR**AMETERS / ***NONE** / ***SYSOUT** / ***SYSLST**(...) / ***EDT**(...)

***SYSLST**(...)

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

***EDT**(...)

WRITE-**MODE** = ***EXTEND** / ***REPLACE**

,**STRUCTURE-OUT**PUT = ***SYSINF** / ***NONE** / <composed-name 1..255>(...)

<composed-name 1..255>>(...)

WRITE-**MODE** = ***REPLACE** / ***EXTEND**

(part 3 of 3)

PRIMARY-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the first comparison member (primary member).

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the primary member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the primary member.

LIBRARY = *LINK(..)
The library assigned via a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to the LMS run.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
All members of the library or the name of the member used as the primary member are compared.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the primary member.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is used as the primary member.

VERSION = *ALL
All versions are taken into account in the comparison.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is used as the primary member.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member that is used as the primary member.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=*HIGHEST-EXISTING, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
All types are taken into account in the comparison.

TYPE = <alphanum-name 1..8 with-wild(20)>
Type of the primary member.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The primary member has any date.

USER-DATE = *TODAY
The member with the current date is used as the primary member.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is used as the primary member.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are used as primary members.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see USER-DATE above.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see USER-DATE above.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded, i.e. all members selected by ELEMENT are used as primary members.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are not to be used as primary members. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see *LIBRARY-ELEMENT.

SECONDARY-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the second comparison member (secondary member). The member selected here is used as the base for the comparison.

LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the secondary member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *BY-SOURCE
The secondary member is contained in the same library as the primary member.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the secondary member.

LIBRARY = *LINK(..)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to the LMS run.

ELEMENT = *BY-SOURCE(...) / *ALL(...) / <composed-name 1..132 with-under with-wild-constr>(...)
Name of the member to be used as the secondary member.

VERSION = *HIGHEST-EXISTING / *BY-SOURCE / *ALL / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version of the secondary member.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is used as the secondary member.

VERSION = *BY-SOURCE
The version of the secondary member is the same as the version of the primary member, or X'FF' if this does not exist.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is used as the secondary member.

VERSION = <composed-name 1..52 with-under with-wild-constr>
Explicitly specifies the version of the member that is used as the secondary member.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the secondary member. For further information concerning specification of the base, see page 50.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Name of the member to be used as the secondary member.

VERSION = *HIGHEST-EXISTING / *BY-SOURCE / *ALL / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version of the secondary member.
For description of the operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

TYPE = *BY-SOURCE / *LMS-DEFAULT / *ALL / <alphanum-name 1..20 with-wild-constr>
Type of the secondary member.

TYPE = *BY-SOURCE
The secondary member has the same type as the primary member.

COMPARE-PARAMETERS = *LMS-DEFAULT / *PARAMETERS(...)
Defines the comparison parameters. Also used to specify the type of comparison (formal or logical) and the scope and format of logging.

RECORD-PART = *LMS-DEFAULT / *ALL / *PART(...)
Defines the comparison area in the record.

RECORD-PART = *ALL
The entire record is compared.

RECORD-PART = *PART(...)
Area specification for the part of the record to be compared.

START = <integer 1..32764>
Starting point of the area containing the part of the record to be compared. If no value is entered, the record is compared starting at the beginning.

LENGTH = *REST / <integer 1..32764>
Length of the area in the record to be compared. If no value is entered, the record is compared starting at the beginning.

SPACES = *LMS-DEFAULT / *STD / *IGNORED / *RELEVANT
Handling of space characters in the record.

SPACES = *STD
Has the same effect as *IGNORED for text members, otherwise as *RELEVANT.

SPACES = *IGNORED
Logical comparison. The comparison fields are compared one character at a time; spaces are ignored.

SPACES = RELEVANT
Formal comparison. The comparison fields are first checked for matching length. If the lengths match, the fields are compared in their entirety. If the lengths differ, the records are logged as being non-matching.

INFORMATION = *LMS-DEFAULT / *MEDIUM / *MINIMUM / *MAXIMUM / *SUMMARY / *STATISTICS / *NONE
Scope of logging.

INFORMATION = *MEDIUM
Standard comparison log. The comparison range of non-matching records is logged in its entirety. With matching records, only range specifications (record numbers) are logged. The comparison statistics are output.

INFORMATION = *MINIMUM
Minimum comparison log. For matching and non-matching records, only range specifications (record numbers) are logged. The comparison statistics are output.

INFORMATION = *MAXIMUM
Detailed comparison log.
All records are logged.
The comparison statistics are output.

INFORMATION = *SUMMARY
No comparison log. Only the comparison statistics are output.

INFORMATION = *STATISTICS
No comparison log. The comparison statistics are output in compressed form. The output is designed for lines with a length of 132.

INFORMATION = *NONE
No logging (no comparison log, no comparison statistics).
*NONE is meaningful only when the SHOW-STATISTICS statement is used.

LAYOUT = *LMS-DEFAULT / *COMPATIBLE / *COMPRESSED
Logging format.

LAYOUT = *COMPATIBLE
The comparison log is output in standard format. This format is compatible with earlier LMS versions.

LAYOUT = *COMPRESSED
The comparison log is output in a compressed format.

JOIN-ELEMENT-SETS = *LMS-DEFAULT / *NO / *YES
Defines the member set to be compared.

JOIN-ELEMENT-SETS = *NO
Only the primary members and the secondary members determined through construction are used for the comparison.

JOIN-ELEMENT-SETS = *YES
All primary and secondary members are used for the comparison.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST (...) / *EDT (...)
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified with MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=.

TEXT-OUTPUT = *NONE
The log output is suppressed, apart from error messages.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
Output is to the work file 9 of EDT. If an error occurs during log output, then the system switches to the default output stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output will be added to this data. If there is no data in the file, the output will be written at the beginning of the file.

WRITE-MODE = *REPLACE
The output will be written at the beginning of work file 9. Any data that is already in the file will be replaced.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Structured output.

STRUCTURE-OUTPUT = *SYSINF
The structured output is placed in the SYSINF stream assigned by /ASSIGN-STREAM (see the “SDF-P“ manual [12]).

STRUCTURE-OUTPUT = *NONE
No structured output.

STRUCTURE-OUTPUT = <composed-name 1..255>(...)
Specifies the S variable in which the structured output is to be placed. This variable must have been declared as a dynamic list variable.
(Command: `DECLARE-VARIABLE NAME=...(TYPE=*STRUCTURE),MULTIPLE-ELEMENTS=*LIST`)

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the list variable is to be overwritten or extended.

WRITE-MODE = *REPLACE
Overwrites the old contents of the list variable.

WRITE-MODE = *EXTEND
Appends the new list members to the existing list.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
2	0	LMS0201	Only the comparison range is logged
2	0	LMS0313	Overflow in statistic counter
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	130	LMS0041	System address space exhausted
	130	LMS0412	Member locked

Notes

– The INFORMATION operand has no influence on the structured output.

– A list member is generated for each comparison of two members. The individual variable members are described in chapter “Format of LMS output in S variables” on page 445.

– In the comparison statistics, the maximum value for element-count fields is 99,999. For line-count fields, it is 999,999,999. If the 9-digit limit overflows, the message LMS0313 will be shown and the affected counters will continue counting modulo 10^9.

Required access rights

For PRIMARY-ELEMENT:	Read authorization for LIBRARY and ELEMENT
For SECONDARY-ELEMENT:	Read authorization for LIBRARY and ELEMENT

Example

The members “TEST1” from libraries BIBU and PLIB are compared. The comparison area comprises the 5th through 30th bytes of the member record.

```
/ADD-FILE-LINK FILE=BIBU;LINK-NAME=LIB%
/START-LMS
//OPEN-LIBRARY LIBRARY=PLIB
//COMPARE-ELEMENT PRIMARY-ELEMENT=*LIB-ELEM(ELEM-TEST1,TYPE=S),-
                SECONDARY-ELEMENT=*LIB-ELEM(LIB=*LINK=LIB5)),-
                COMPARE-PARAMETERS=*PARAMETERS(INFORMATION=*MAXIMUM,-
                RECORD-PART=*PART(START=5;LENGTH=26))
...
//END
```

COPY-ELEMENT

COPY-ELEMENT copies members and libraries one to one. The copied members may receive new member designations. The source and target member base types may differ if text members are copied.
The following copy options are available:

– copying one or more members in the same library

– copying one or more members to a different library

– copying a complete library (see example on page 225)

The copied members can be stored as either non-delta or delta members. If the input library and the output library are the same when copying delta members, the copied delta members must be given new member names. Leaves of delta trees may be overwritten.

The source member can be deleted after copying using DELETE-SOURCE = *YES, thus allowing you to move a member.

Copy with structure (STORAGE-FORM = *BY-SOURCE)

When this format is used, LMS recognizes the form in which members are stored in the PLAM libraries. Correspondingly, delta trees are copied as delta trees and all other members are copied to the output file as non-delta members.

Notes

– If the copying process is aborted, the copied part of a delta tree is retained.

– VERSION, BASE and all date operands must be set to their default values. The complete name range, i.e. as version=*, is always copied.

– Specification of library lists causes errors.

Example

Members A/1 and A/2 are maintained in the library. The statement “Copy member A to B” causes member B/2 to be generated when STORAGE-FORM=*STD is specified, and members B/1 and B/2 when STORAGE-FORM=*BY-SOURCE is specified.

Overwriting the target name range (WRITE-MODE=*SUBSTITUTE)

Specifying WRITE-MODE=*SUBSTITUTE makes the copied member the only member in the target library with its type and name. Before copying the member into the target library, LMS deletes all members having the same type and name as the target member. This means that all user specifications in TO-ELEMENT (such as VERSION = *INCREMENT) are applied only to the empty target name range. If, for example, *INCREMENT is specified, the default version is generated.

Restrictions

1. The STORAGE-FORM= operand must not have the value *BY-SOURCE.
2. The input library must not be the same as the output library.

If an error occurs during deletion of the target name range (because, for example, a member is write-protected), the COPY-ELEMENT statement is aborted.

```
COPY-ELEMENT

ELEMENT = *LIBRARY-ELEMENT (...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
      *ALL(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
      <composed-name 1..64 with-under with-wild(132)>(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
    ,TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
```

(part 1 of 3)

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 2 of 3)

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**TO-E**LEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / ***BY-SOUR**CE / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)

***BY-SOUR**CE(...)

VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..132 with-under with-wild-constr>(...)

VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / <alphanum-name 1..20 with-wild-constr>

,**USER-DATE** = ***BY-SOUR**CE / ***TODAY** / <date 8..10 with-compl>

,**STOR**AGE-**FORM** = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***STD** / ***FULL** / ***DELTA**

,**PROT**ECTION = ***LMS-DEF**AULT / ***STD** / ***BY-SOUR**CE

,**DEL**ETE-**SOUR**CE = ***NO** / ***YES**

,**WR**ITE-**MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***EXTEND** / ***SUBST**ITUTE / ***ANY**

,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

(part 3 of 3)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the members to be copied.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library from which the members are to be copied.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL (...) / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be copied.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be copied.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is copied.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is copied.

VERSION = <composed-name 1..24 with-under with-wild(52)>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=*HIGHEST-EXISTING, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be copied.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be copied has any date.

USER-DATE = *TODAY
The member with the current date is copied.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is copied.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are copied.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded from copying.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are to be excluded from copying. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the destination and name under which the member is to be added.

LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
Specifies the new library name or library to which the member is to be added.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *BY-SOURCE
The member is copied to the library which contains the member being copied.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the file is to be added as a member. If the library does not yet exist, it will be created.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wild-constr>(...)
Name that the new member to be added is to receive.

ELEMENT = *BY-SOURCE(...)
The new name is the same as the old name.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the new member to be added is to receive.

VERSION = *BY-SOURCE
The new member receives the same version as the original member. If the original member has no version specification, the new member receives X'FF' as the version specification.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated (see also section "Version conventions" on page 53).

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..52 with-under with-wild-constr>
The new member receives the version specified here.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Name of the new member to be added. It can also be entered using wildcards.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the new member to be added is to receive.
For description of operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=*HIGHEST-EXISTING, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *BY-SOURCE / *LMS-DEFAULT / <alphanum-name 1..20 with-wild-constr>
Type that the new member to be added is to receive.

TYPE = *BY-SOURCE
The new member receives the same type designation as the original member.

USER-DATE = *BY-SOURCE / *TODAY / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *BY-SOURCE
The new member receives the same date as the original member.

USER-DATE = *TODAY
The current date is given.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

STORAGE-FORM = *LMS-DEFAULT / *BY-SOURCE / *STD / *FULL / *DELTA
Storage form for the member being created. The storage form must not contradict the settings made by means of the MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES statements, and all members of a given type and name must have the same storage form.

STORAGE-FORM = *BY-SOURCE
VERSION, BASE and all date operands must be set to their default values. The complete name range, i.e. as version=*, is always copied.
The member being added is copied with the same structure, i.e. delta trees are again stored as such and full members are copied as full members. If delta trees are copied with the same structure, the target name must not yet exist in the target type (i.e. WRITE-MODE has no effect).

STORAGE-FORM = *STD
The member is generated in accordance with the storage form required for the member scope. Contradictory requirements result in errors. If nothing special is required, full storage is selected.

STORAGE-FORM = *FULL
The new member is generated as a full member (if this is not possible, an error message is issued).

STORAGE-FORM = *DELTA
The new member is generated as a delta member (if this is not possible, an error message is issued). This entry is permissible for member types: S, P, D, J, M, X and members types derived from them.

PROTECTION = *LMS-DEFAULT / *STD / *BY-SOURCE
Member protection for the copied member.

PROTECTION = *STD
If the member already exists, the member protection remains unchanged. If the member does not yet exist and if an initial member protection is defined for the library or type of member, then the copied member will receive this protection.

PROTECTION = *BY-SOURCE
The copied member receives the same protection as the original member.

DELETE-SOURCE = *NO / *YES
Here the user can specify whether the source member is to be kept, or whether it is to be deleted.

DELETE-SOURCE = *NO
The source member is not deleted.

DELETE-SOURCE = *YES
The source member is deleted.

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *EXTEND / *SUBSTITUTE / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.
If the member to be stored is a delta member, it is necessary to ensure that the member is a leaf of the delta tree. Only leaves of a delta tree may be overwritten.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is then replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise, it will be created as a new member. *EXTEND is not permitted for delta members,

WRITE-MODE = *SUBSTITUTE
All members having the same type and name as the source member are deleted from the target library. The source member is then copied into the library.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the DIALOG-CONTROL operand of the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For LIBRARY-ELEMENT: read authorization for LIBRARY and ELEMENT

If more than one member is affected by the statement, members without read authorization are excluded from the statement.

When copying deltas with STORAGE-FORM=*BY-SOURCE, the delta tree will only be copied if read authorization exists for all its members.

For TO-ELEMENT: Read and write authorization for LIBRARY

Administer authorization where the specified member designation is new. Otherwise, only write authorization for the member existing under the specified member designation (administer authorization no longer required).

For STORAGE-FORM=*DELTA, read authorization must be granted for the member defined by BASE.

If WRITE-CONTROL is active and a base version exists, the USERID of the user must be entered as the HOLDER of the member specified by BASE. Only if write authorization has been granted can a new version be generated or this base version overwritten. In this case, administer authorization is no longer required.

If PROTECTION=*BY-SOURCE is specified, only the owner of the library file can use this functionality.

Notes

– When creating a member, be sure to observe the convention applicable to the type of member involved. Especially when the target type has the convention STD-TREE, problems can occur if the source side contains side branch versions whose main branch version is deleted. In this case, the affected side branches cannot be copied; LMS does issue an error message, however.

– Exception: copying a complete delta tree with STORAGE-FORM=*BY-SOURCE is always possible.

– STATE and HOLDER of the source member are not applied to the target member even with PROTECTION=*BY-SOURCE.

– If WRITE-CONTROL is active in the output library, the access method adds a comment (record type 2) to the member which is to be written. The comment logs the HOLDER, DATE and TIME of the process. The record is written as the first record of the record type. Any existing comment records are copied after this first record. If, in addition, the member is written to the base of a different version (i.e. not the first version under a name), the member attributes STATE and HOLDER and all the rights of the base version are applied to the new version. The CCSN is adopted from the source file. The USER-DATE is determined anew.

*Examples with WRITE-MODE=*SUBSTITUTE*

– Input library X contains the S-type member A/1. Output library Y contains the S-type member A/2.

```
//COPY-ELEM ELEMENT= *LIB(LIB=X,ELEM=A,TYPE=S),-
          TO ELEMENT= *LIB(LIB=Y),WRITE-MODE=*SUBSTITUTE
```

Following this statement, member A/1 is the only member of type S and name A existing in the output library. Member A/2 has been deleted.

– All the members of a product version are located in input library X. These members are to be copied into an existing output library Y in such a way that, after the copying process is concluded, Y contains only the copied product version and no other version. This can be done with the following statement.

```
//COPY-ELEM ELEMENT= *LIB(LIB=X,ELEM=*,TYPE=*),-
          TO ELEMENT= *LIB(LIB=Y),WRITE-MODE=*SUBSTITUTE
```

Example of how to copy an entire library

Library lib1 is copied in its entirety and is given the name lib2. By specifying '*' for member and type, no knowledge is required of the members contained, i.e. all members are copied one to one to the newly created library lib2.

```
/START-LMS
//OPEN-LIBRARY LIB1
//SHOW-ELEMENT-ATTRIBUTES
 INPUT  LIBRARY= :1OSQ:$USER.LIB1
TYP NAME     VER (VAR#) DATE         NAME     VER (VAR#) DATE
(D) LETTER.A @   (0001) 2011-04-12   TESTELEM @   (0001) 2011-04-12
       2 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME  VER (VAR#) DATE
(S) TEST3 @   (0001) 2011-04-12
       1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
------------------------------------------------------------------------------
       3 ELEMENT(S) IN THIS TABLE OF CONTENTS
//COPY-ELEMENT (,ELEM=*,TYPE=*),TO-ELEM=*LIB(LIBRARY=LIB2)
//SHOW-LIBRARY-STATUS
 STATUS FILENAME                                    MODE    LINK DEF-TYPE
OPEN :1OSQ:$USER.LIB2
OPEN :1OSQ:$USER.LIB1                               READ
//SHOW-ELEMENT-ATTRIBUTES *LIB(LIBRARY=LIB2)
 INPUT  LIBRARY= :N:$USER.LIB2
TYP NAME     VER (VAR#) DATE         NAME     VER (VAR#) DATE
(D) LETTER.A @   (0001) 2011-04-12   TESTELEM @   (0001) 2011-04-12
       2 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME  VER (VAR#) DATE
(S) TEST3 @   (0001) 2011-04-12
       1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
------------------------------------------------------------------------------
       3 ELEMENT(S) IN THIS TABLE OF CONTENTS
//END
```

COPY-LIBRARY

The COPY-LIBRARY statement copies a library in its entirety, i.e. together with all its library, type and member attributes. The target library must either have FILE-STRUCTURE=NONE or not yet exist. The target library is given the library format corresponding to its value for BUFFER-LENGTH. The statement is thus suitable for converting a library format.

The file protection attributes of the source library can be applied to the target library. The statement is thus also suitable for reorganizing libraries. The target library is logically identical to the original and occupies only the minimum required disk space.

If an error occurs during processing of the COPY-LIBRARY statement (e.g. insufficient disk space), the target library is not complete.

COPY-LIBRARY

LIBRARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**TO-LIB**RARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**F**ILE-**ATTR**IBUTES = ***STD** / ***BY-SOUR**CE

LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Specifies the library which is to be copied.

LIBRARY = <filename 1..54 without-vers>
Copies the library with the specified name.

LIBRARY = *LINK(...)
Copies the library assigned by means of a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library, which was declared with a /ADD-FILE-LINK command.

TO-LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Specifies the target library.

TO-LIBRARY = <filename 1..54 without-vers>
Generates a library with the specified name.

TO-LIBRARY = *LINK(...)
Generates the library assigned by means of a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library, which was declared with a /ADD-FILE-LINK command.

FILE-ATTRIBUTES = *STD / *BY-SOURCE
Attributes of the target library file.

FILE-ATTRIBUTES = *STD
The file attributes of the target library are not changed. New files will be generated with the default values defined by the file management system.

FILE-ATTRIBUTES = *BY-SOURCE
The file protection attributes of the source library are applied to the target library (analogous to `/COPY-FILE ..,PROTECTION=*SAME`).

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked

Required access rights

For LIBRARY: read authorization
For TO-LIBRARY: read and write authorization

Ownership of LIBRARY and LIBRARY are not protected against TU-UPAM accesses or there are no protection attributes at library, type and member level.

Note

Library lists are not permitted.

Examples

– Copying a library to an NK4 pubset

```
/start-lms
//copy-library library=lib,to-library=:nk4:lib
//end
```

– Creating the NK4 library format in advance

```
/add-file-link file-name=nk4.lib,link-name=nk4,buffer-length=*std(2)
/start-lms
//copy-library library=nk2lib,to-library=*link(nk4)
//end
```

– Reorganizing a library with buffer

```
/delete-file file-name=tolib
/start-lms
//copy-library library=lib,to-library=tolib,file-attributes=*by-source
//end
/copy-file from-file=tolib,to-file=lib
/delete-file file-name=tolib
```

DEACTIVATE-USER-EXIT

The DEACTIVATE-USER-EXIT statement deactivates the user exits that were activated by ACTIVATE-USER-EXIT. The user exits are no longer used with the next corresponding SHOW-ELEMENT or COMPARE-ELEMENT. However, the user routine is not yet unloaded since it may be required elsewhere. This means that in the case of the next ACTIVATE-USER-EXIT with the same entry point, the user program does not need to be linked in again.

The DEACTIVATE-USER-EXIT statement requires specification of the function for which the user exit is to be deactivated.

```
DEACTIVATE-USER-EXIT

FUNCTION = *SHOW-ELEMENT / *COMPARE-ELEMENT(...)
  *COMPARE-ELEMENT(...)
    | ELEMENT = *PRIMARY / *SECONDARY
```

FUNCTION = *SHOW-ELEMENT / *COMPARE-ELEMENT(...)
Defines the LMS statement for which the user exit is to be deactivated.

FUNCTION = *SHOW-ELEMENT
Deactivate user exit for the SHOW-ELEMENT function.

FUNCTION = *COMPARE-ELEMENT(...)
Deactivate user exit for the COMPARE-ELEMENT function.

ELEMENT = *PRIMARY / *SECONDARY
If the user exit is deactivated for the COMPARE-ELEMENT statement, it is still necessary to define whether the member is a primary or secondary member.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Required access rights

No access rights are necessary.

DELETE-ELEMENT

The DELETE-ELEMENT statement deletes the specified members in the assigned library (logical deletion). The directory entries are thereby deleted and storage space is released.

A member of a library is deleted physically

– if the member contains a code for physical deletion

– if the operand DESTROY=*YES has been set

– if the class 2 option DESTLEV requires it.

Delta members are not physically deleted until the last delta member of a delta tree, i.e. the complete delta tree, is deleted.

The statement is executed only if a library has been specified explicitly in the statement or the library specified under OPEN-LIBRARY has been opened with MODE=*UPDATE.

The DELETE-ELEMENT statement is permitted for all member types.

DELETE-**ELEM**ENT

ELEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 1 of 2)

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**DESTROY-DATA** = ***LMS-DEF**AULT / ***NO** / ***Y**ES

,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES

(part 2 of 2)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the member to be deleted.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library from which the member is to be deleted.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library from which the member is to be deleted.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be deleted.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be deleted.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is deleted.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is deleted.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be deleted.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=*HIGHEST-EXISTING, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be deleted.

USER-DATE = *ANY /*TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be deleted has any date.

USER-DATE = *TODAY
The member with the current date is deleted.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is deleted.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are deleted.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded from deletion.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are to be excluded from deletion. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the LIBRARY-ELEMENT operand of this statement.

DESTROY-DATA = *LMS-DEFAULT / *NO / *YES
Deletes the data for all members defined by *LIBRARY-ELEMENT.

DESTROY-DATA = *NO
A member of a library is deleted physically only if it contains a flag for physical deletion or the class 2 option DESTLEV requires it.

DESTROY-DATA = *YES
Following logical deletion, the data, if present, is deleted physically, i.e. overwritten with X’00’.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement, where the value *ERROR which might have been set there has the same effect as *NO. Likewise, the value *ERROR which may have been set for DIALOG-CONTROL= in the /SEND-MSG message command has the same effect as *NO with DELETE-ELEMENT.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member does not exist
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

Read and write authorization for LIBRARY
Administer authorization and write authorization for ELEMENT.

Notes

– With each UPDATE to a delta tree the delta structure is reorganized, i.e. records no longer required are deleted and the storage space no longer required is released.

– Deleting a main branch version having the convention STD-TREE on which side branch versions are dependent can lead to problems in subsequent copying.

Example

Member TEST3 is deleted from library LIB1.

```
/START-LMS
//OPEN-LIBRARY LIBL,MODE=*UPDATE
//SHOW-ELEMENT-ATTRIBUTES
TYP NAME     VER (VAR#) DATE         NAME     VER (VAR#) DATE
(D) LETTER.A @   (0001) 2012-04-12   TESTELEM @   (0001) 2012-04-12
       2 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME  VER (VAR#) DATE
(S) TEST3 @   (0001) 2012-04-12
       1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
------------------------------------------------------------------------------
       3 ELEMENT(S) IN THIS TABLE OF CONTENTS
//DELETE-ELEMENT *LIB(ELEM=TEST§,TYPE=S)
//SHOW-ELEMENT-ATTRIBUTES
INPUT  LIBRARY= :1OSQ:$USER.LIB1
TYP NAME     VER (VAR#) DATE         NAME     VER (VAR#) DATE
(D) LETTER.A @   (0001) 2012-04-12   TESTELEM @   (0001) 2012-04-12
       2 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//END
```

EDIT-ELEMENT

The EDIT-ELEMENT statement calls the file editor EDT in order to create, correct or view text member (see also [10] and [11]).

The source and target base types may differ.

Creating and correcting text members

The EDIT-ELEMENT statement calls EDT and reads the specified member from the assigned library into work file 0. The contents of work file 0 are deleted beforehand. When EDT is terminated, the corrected or newly generated member is written to the assigned library.

LMS supports EDT versions higher than V16.2A.

As of EDT V16.4A, XHCS is supported. When it is called, LMS passes the coded-character-set name of the relevant member to EDT and writes the member back with the value set in EDT. If no input member is specified, “no code” is assumed as the input CCSN.

In the case of EDT-versions < V16.4A, the CCSN of the input member is entered as the CCSN of the target member. If no input member is specified, the target member is given the CCSN “no code”.

Editing is not possible if RECORD-FORMAT=FIXED, KEY-POSITION > 5 or KEY-LENGTH > 8 has been stored in the attribute record (record type 164) of the input member.

For the editing of Unicode members EDT V17.0 or higher is necessary.

Scratch file

When EDT is called, LMS generates a scratch file with the link name EDTISAM if the member contains ISAM keys.

Name of the scratch file:

S.LMS.TSNnnnn.date.time-of-day.member

where “member” can be up to 9 characters long. Member names exceeding this limit are truncated after the first 9 characters. If the “member” suffix would result in an illegal BS2000 file name (e.g. ’.’ as the ninth character), LMS forms a scratch file name without the member name:

Editor run

– LMS passes the member records on to EDT. The member is then available in virtual memory.

– If ISAM keys are stored in the member, the line number displayed by EDT gives the first six digits of the ISAM keys.

– The member being processed is locked to other users.

Termination of the EDT run

RETURN from work file 0:

If not empty, work file 0 is added as a member to the output library. The EDT data (files in virtual memory, variables, etc.) remains intact. This data is released only in the case of a severe EDT error.

HALT from work file 0:

The following query is issued:

```
LMS0420: EDITED ELEMENT (type)membername/version[(variantnumber)]/date TO
BE ADDED? REPLY (Y=YES, N=NO or R=RETURN TO EDITOR)?
```

The response determines whether the current work file is added as a member or returned to EDT. The EDT data remains intact. This data is released only in case of a severe EDT error.

HALT/RETURN from work file ≠ 0:

The following query is issued:

```
LMS0420: EDITED ELEMENT (type)membername/version[(variantnumber)]/date TO
BE ADDED? REPLY (Y=YES, N=NO or R=RETURN TO EDITOR)?
```

If 'N' is entered, the member is not added. If the reply is 'Y', the following dialog is then conducted with the user:

```
LMS0421: WORKFILE TO BE ADDED (0 = WORKFILE(0),..., N = NONE)
```

If the reply is 'N', LMS returns to the EDT work file currently being processed.

EDT in batch mode

@RETURN:

The corrected member is added from work file 0 to the output library, provided that the output library is not empty.

@HALT:

The corrected member is not added to the output library.

Displaying information from EDT on the member being edited

RETURN ? / HALT ?

EDT displays the names of the target library and the target member, along with the version and type of the member being edited in work file 0. EDT also displays part of the member which was edited. Editing then resumes.
If ELEMENT=*NONE or TO-ELEMENT=*NONE are specified, RETURN ? and HALT ? have no effect.

```
EDIT-ELEMENT

ELEMENT = *NONE / *LIBRARY-ELEMENT(...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
      *ALL(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
      <composed-name 1..64 with-under with-wild(132)>(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
    ,TYPE = *LMS-DEFAULT / <alphanum-name 1..8>
```

(part 1 of 3)

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 2 of 3)

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**TO-E**LEMENT = ***LIBRARY-ELEMENT**(...) / ***NONE**
***LIB**RARY-**ELEM**ENT(...)
LIBRARY = ***STD** / ***BY-SOUR**CE / <filename 1..54 without-vers> / ***LINK**(...)
***LINK**(...)
LINK-NAME = <structured-name 1..8>
,**ELEM**ENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)
***BY-SOUR**CE(...)
VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
<composed-name 1..132 with-under with-wild-constr>(...)
VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / <alphanum-name 1..8>
,**USER-DATE** = ***TODAY** / ***BY-SOUR**CE / <date 8..10 with-compl>
,**STOR**AGE-**FORM** = ***LMS-DEF**AULT / ***STD** / ***FULL** / ***DELTA**
,**EDT-MODE** = ***STD** / ***COMPATIBLE** / ***UNICODE**
,**EDIT**OR-**COM**MANDS = ***NONE** / <c-string 1..251> / ***LOW**ER-**CASE**(...)
***LOW**ER-**CASE**(...)
EDITOR-**COM**MANDS = <c-string 1..251 with-low>
,**INF**ORMATION = ***TEXT** / list-poss(2): ***TEXT** / ***COMMENT**
,**WR**ITE-**MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**
,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

(part 3 of 3)

ELEMENT = *NONE / *LIBRARY-ELEMENT(...)
Specifies the member to be edited.

ELEMENT = *NONE
No input member is specified. The member to be edited is created as a new member, or overwritten with new data.

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library from which the member is to be edited.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library from which the member is to be taken.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be edited.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be edited.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is used.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is edited.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be edited.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / <alphanum-name 1..8>
Type of the member to be edited.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be edited has any date.

USER-DATE = *TODAY
The member with the current date is edited.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is edited.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are edited.

FROM = 1900-01-01 / <date 8..10> with-compl
Beginning of interval.

TO = *TODAY / <date 8..10> with-compl
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded from editing.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are not to be excluded from editing. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see *LIBRARY-ELEMENT.

TO-ELEMENT = *LIBRARY-ELEMENT(...) / *NONE
Specifies the destination to which and the name under which the member is to be written back.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
Specifies the library to which the member is to be added.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *BY-SOURCE
The member is added to the library from which it was taken.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the file is to be added as a member. If the library does not yet exist, it will be created.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wild-constr>(...)
Name that the new member to be added is to receive.

ELEMENT = *BY-SOURCE(...)
The new name is the same as the old name.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the new member to be added is to receive.

VERSION = *BY-SOURCE
The new member receives the same version as the original member. If the original member has no version specification, the new member receives X'FF' as the version specification.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated (see also page 50).

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..52 with-under with-wild-constr>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Name of the new member to be added. It can also be entered using wildcards.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the new member to be added is to receive.
For description of operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

TYPE = *BY-SOURCE / *LMS-DEFAULT / <alphanum-name 1..8>
Type that the new member to be added is to receive.

TYPE = *BY-SOURCE
The new member receives the same type designation as the original member.

USER-DATE = *TODAY / *BY-SOURCE / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *TODAY
The current date is given.

USER-DATE = *BY-SOURCE
The new member is given the same as the source member.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

STORAGE-FORM = *LMS-DEFAULT / *STD / *FULL / *DELTA
Storage form for the member being added. The storage form must not contradict the settings made by means of the MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES statements, and all members of a given type and name must have the same storage form.

STORAGE-FORM = *STD
The member is generated in accordance with the storage form required for the member scope. Contradictory requirements result in errors. If nothing is specified, full storage is selected.

STORAGE-FORM = *FULL
The new member is generated as a full member (if this is not possible, an error message is issued).

STORAGE-FORM = *DELTA
The new member is generated as a delta member (if this is not possible, an error message is issued). This entry is permissible for member types: S, P, D, J, M, X and members types derived from them.

TO-ELEMENT = *NONE
The edited member is not written back.

EDT-MODE = *STD / *COMPATIBLE / *UNICODE
Specifies the mode that EDT is to be called in.

EDT-MODE = *STD
EDT is called in the mode specified by the MODIFY-LMS-DEFAULTS statement, but is allowed to switch modes if necessary.

EDT-MODE = *COMPATIBLE
EDT is called in compatibility mode.

EDT-MODE = *UNICODE
EDT is called in Unicode mode.

EDITOR-COMMANDS = *NONE / <c-string 1..251> / *LOWER-CASE(...)
Specifies a sequence of editor commands.

EDITOR-COMMANDS = *NONE
No sequence of editor commands is specified.

EDITOR-COMMANDS = <c-string 1..251>
Sequence of editor commands which are to be executed after EDT is called. In the entry, the commands must be separated from one another by a semicolon (;). With the exception of EDIT and RETURN, it is possible to specify any commands which are permitted both in F mode and in L mode of EDT. The EDIT command is permitted only in the form EDIT ONLY. It should be noted that the HALT command in a sequence of EDT commands causes EDT to terminate and so also causes the EDT data to be released (files in virtual memory, variables, etc.).
Following execution of this sequence of EDT commands, the generated or edited member is written back directly.
Lowercase letters are converted to uppercase.

EDITOR-COMMANDS = *LOWER-CASE(...)
Lowercase letters are not converted to uppercase.

EDITOR-COMMANDS = <c-string 1..251 with-low>
Sequence of editor commands as described above, except that lowercase letters are not converted to uppercase.

Example of an EDT command

```
EDITOR-COM = ´ON&C´´A´´TO´´B´´´
```

Example of an EDT command without conversion of lowercase letters

```
EDITOR-COM = (´(LOWER ON;ON&C´´a´´TO´´b´´´)
```

INFORMATION = *TEXT / list-poss(2): *TEXT / *COMMENT
The section of the member which is to be edited.

INFORMATION = *TEXT
The text itself, i.e. record type 1, is to be edited.

INFORMATION = *COMMENT
The separately stored comment, i.e. record type 2, is to be edited.

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a member having the same name.
If the source member is the same as the target member, the WRITE-MODE operand is ignored.
If the member to be stored is a delta member, it is necessary to ensure that the member is a leaf of the delta tree. Only leaves of a delta tree may be overwritten.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise it will be created as a new member. *EXTEND is not permitted for delta members,

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
2	0	LMS0163	At least one record truncated
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For LIBRARY-ELEMENT: read authorization for LIBRARY and ELEMENT

If more than one member is affected by the statement, members without read authorization are excluded from the statement.

For TO-ELEMENT: read and write authorization for LIBRARY

Administer authorization where the specified member designation is new. Otherwise, only write authorization for the member existing under the specified member designation (administer authorization no longer required).

For STORAGE-FORM=*DELTA, read authorization must be granted for the member defined by BASE.

If WRITE-CONTROL is active and a base version exists, the user ID of the user must be entered as the HOLDER of the member specified by BASE. Only if write authorization has been granted can a new version be generated or this base version overwritten. In this case, administer authorization is no longer required.

Notes

– See note under CALL-EDT.

– When creating a member, be sure to take into account the convention applicable to the member type.

– Specifying a list for INFORMATION results in each of the list entries being edited in work file 0, i.e. the individual sections of the member are edited one after the other. The EDT run continues until the last selected section has been edited and a decision has been made as to whether the edited member (all of the selected sections) is to be added to the library.

– If WRITE-CONTROL is active in the output library, the access method adds a comment (record type 2) to the member which is to be written. The comment logs the HOLDER, DATE and TIME of the operation. The record is written as the first record of the record type. Any existing comment records or those which have been edited by means of INFORMATION *COMMENT are copied after it. If, in addition, the member is written to the base of a different version (i.e. not the first version under a name), the member attributes STATE and HOLDER and all the rights of the base version are adopted for the new version. The CCSN is adopted from the source file. The USER-DATE is determined anew.

– If an EDT-MODE is specified explicitly in the EDIT-ELEMENT statement, but EDT cannot start in the specified mode, the statement is aborted and the message LMS0297 is output.

– If no EDT-MODE is specified explicitly in the EDIT-ELEMENT statement, EDT starts in the default EDT mode, but can switch modes by itself if necessary. The default EDT mode is COMPATIBLE at the beginning of the LMS run and can be changed by the MODIFY-LMS-DEFAULTS statement.

Examples

The member test4 is to be edited with EDT:

```
//edit-elem *lib(elem=test4,type=s)
```

The member test4 is to be edited with EDT, whereby the string 'old_word' becomes the string ’new_word’:

```
//edit-elem *lib(elem=test4,type=s),ed-com=*low-case('on 1-10
c''old_word''t''new_word''')
```

The member test4 is to be viewed with EDT but not written back:

```
//edit-elem (elem=test4,type=s),to-elem=*none
```

In the library LIBCCSN, the member UTFE is in the character set UTFE.
The member LONG-LINES is in the standard 7-bit character set EDF03IRV, but has lines that are longer than 256 characters. The default EDT mode is COMPATIBLE.

```
//modify-lms-defaults edt-mode=*compatible
//open-lib libccsn,*upd
//edit-elem (libccsn,utfe,s),edt-mode=*compatible
% LMS0297 CHANGE OF EDT OPERATING MODE NOT POSSIBLE ——————————————————  (1)
//edit-elem (libccsn,utfe,s)  ————————————————————————————————————————  (2)
```

```
1.00  LINE 1
2.00  LINE 2



22.00
23.00
ret                                                    0000.00:00001(00)
LTG                                                             TAST
```

```
//edit-elem (libccsn,long-lines,s),edt-mode=*unicode ——————————————————  (3)
```

```
1.00  LINE 1
2.00  very long line ... longer than 256 characters



22.00
23.00
ret                                                    0000.00:00001(00)
LTG                                                             TAST
```

(1) EDT fails to start because Unicode character sets are not supported in compatibility mode.

(2) EDT automatically switches to Unicode mode

(3) Unicode mode is specified explicitly

EDIT-ELEMENT-ATTRIBUTES

The EDIT-ELEMENT-ATTRIBUTES statement starts the guided dialog mechanism for the MODIFY-ELEMENT-ATTRIBUTES statement. Where technically possible and helpful, the predefined default operand values are each replaced by values currently applicable to the specified element.

```
EDIT-ELEMENT-ATTRIBUTES

ELEMENT = *LIBRARY-ELEMENT (...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = <composed-name 1..64 with-under>(...)
      <composed-name 1..64 with-under>(...)
        VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
                  <composed-name 1..24 with-under>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
    ,TYPE = *LMS-DEFAULT / <alphanum-name 1..8>
```

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(..)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = <composed-name 1..64 with-under>(...)
Name of the member whose attributes are to be modified.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the member.

VERSION = *HIGHEST-EXISTING
The attributes of the member with the highest existing version are modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' of the member in the library under the specified TYPE and name is used.

VERSION = <composed-name 1..24 with-under>
Explicitly specifies the version of the member.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

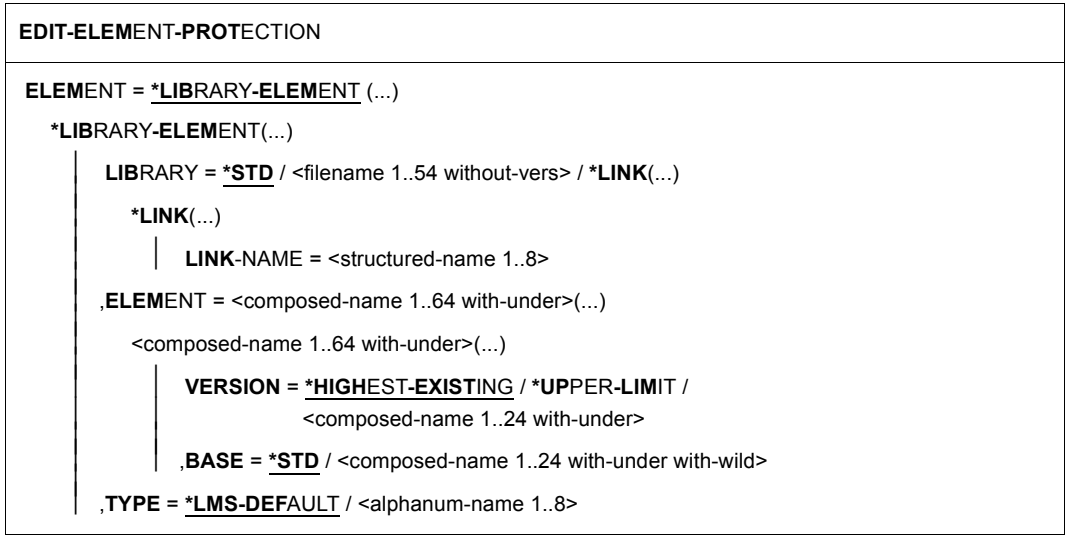
TYPE = *LMS-DEFAULT / <alphanum-name 1..8 with-wild(20)>
Type of the member.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0303	Member outside reference condition range
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

EDIT-ELEMENT-PROTECTION

The EDIT-ELEMENT-PROTECTION statement starts the guided dialog mechanism for the MODIFY-ELEMENT-PROTECTION statement. Where technically possible and helpful, the predefined default operand values are each replaced by values currently applicable to the specified element.



ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = <composed-name 1..64 with-under>(...)
Name of the member whose protection attributes are to be modified.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the member.

VERSION = *HIGHEST-EXISTING
The protection attributes of the member with the highest existing version are modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' of the member in the library under the specified TYPE and name is used.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / <alphanum-name 1..8>
Type of the member.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member not found
	64	LMS0303	Member outside reference condition range
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

END

END terminates the LMS program. All libraries that are still open are closed.

END

The END statement has no operands.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	LMS terminated normally
	64	CMD0205	LMS terminated abnormally

EXTRACT-ELEMENT

The EXTRACT-ELEMENT statement outputs library members to files, unless the user specifies otherwise. If the statement is entered in the EDT command line, the member data is written to the current EDT work file by default.

LMS creates the files in accordance with

– the entry in the task file table (TFT), if the file has been assigned via the link name,

– the stored file attributes and the FILE-ATTRIBUTES operand and

– the catalog entry.

The files can have RECORD-FORMAT=UNDEFINED and arbitrary BUFFER-LENGTH and RECORD-SIZE values. However, the maximum record length of 32 Kbytes (including the record header) must not be exceeded.

If the ISAM keys of an ISAM file have been included in the member, the ISAM keys are also output when EXTRACT-ELEMENT is issued.

If information on ISAM secondary keys was stored when the file was added, the secondary keys are recreated. If some or all of the secondary keys cannot be recreated, the file is generated without those keys.

The EXTRACT-ELEMENT statement is permitted for the member types S, M, R, J, P, D, X, C and member types derived from them.

C-type members, PAM files under type X and types derived from them are generated as PAM files.

The generated file contains the CCS name of the source member as its CCS catalog attribute. If the member data is written to an EDT work file, the EXTRACT-ELEMENT statement is permitted only for textual member types or types derived from them, whereby the CCSN of each member is passed to EDT.

Note

Valid member names are not always permitted as file names.

Generating ISAM files

When members are output to ISAM files, LMS generates the ISAM keys as follows:

– If the ISAM keys are also added when an ISAM file is included as a library member, LMS generates the ISAM file with those ISAM keys which have been stored.

– If no ISAM keys have been stored in the input member, an ISAM file with KEY-POSITION = 5 and KEY-LENGTH = 8 is created. LMS then normally generates ISAM keys with an initial value of 1000 and an increment of 1000. If the member is too large for this increment (more than 100,000 records), the increment will be calculated from the number of records.

Notes

– R-type members are output up to the END record. Any records which come afterwards are ignored.

– Correction journal records (TXTP) are not included in the output in the case of C-type members.

– RECORD-SIZE is supplied with values only with RECORD-FORMAT=FIXED; with RECORD-FORMAT=VARIABLE, the value is 0.

EXTRACT-ELEMENT

ELEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***HIGH**EST**-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***HIGH**EST**-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION**-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

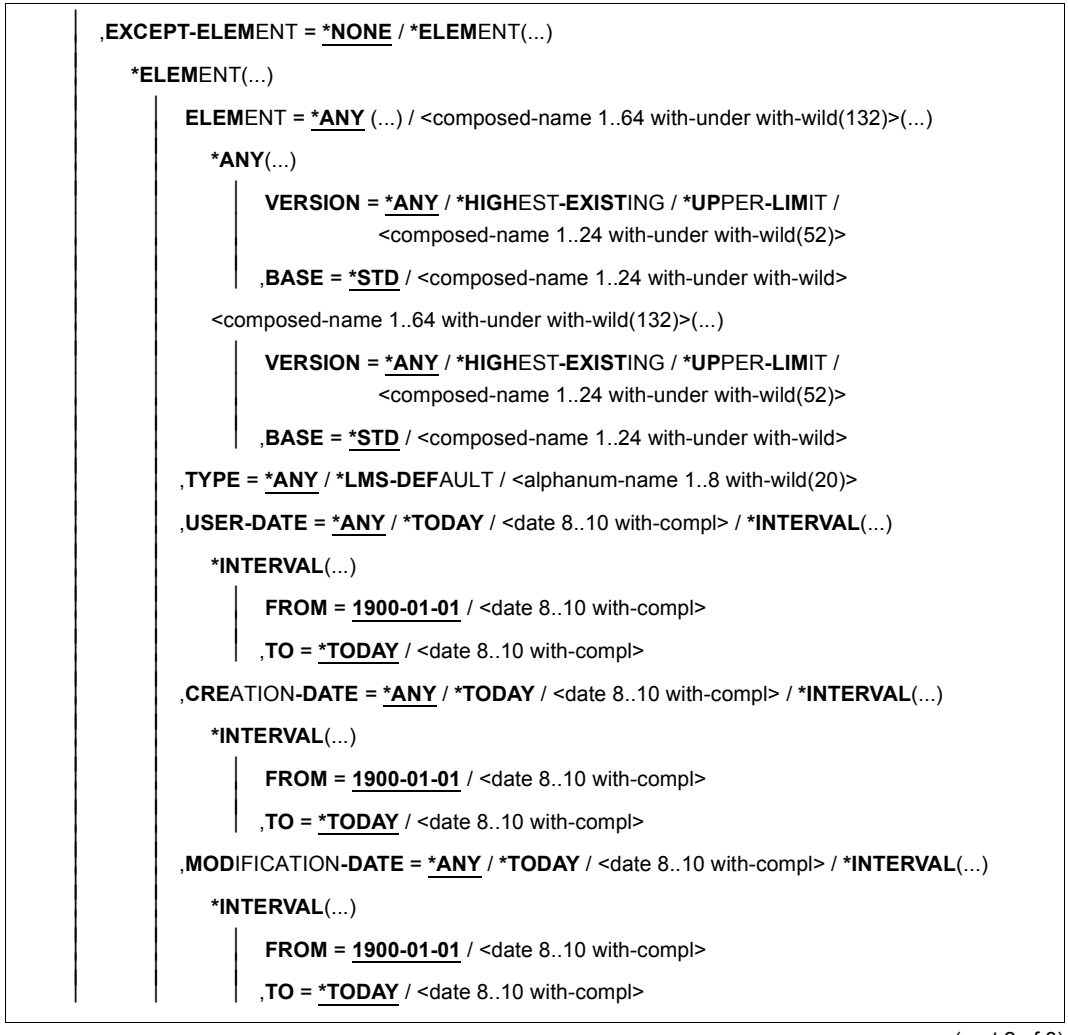
,**MOD**IFICATION**-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 1 of 3)


,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)
***ELEM**ENT(...)
ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)
***ANY**(...)
VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
<composed-name 1..64 with-under with-wild(132)>(...)
VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>
,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 2 of 3)

```
,TO-FILE = *STD / *BY-SOURCE / <filename 1..54 without-gen-vers with-wild-constr> / *LINK(...)
    *LINK(...)
        | LINK-NAME = <structured-name 1..8>
,FILE-ATTRIBUTES = *BY-ELEMENT / *BY-CATALOG / *LMS-DEFAULT / *PARAMETERS(...)
    *PARAMETERS(...)
        | ACCESS-METHOD = *LMS-DEFAULT / *ISAM / *SAM
,INFORMATION = *TEXT / list-poss(2): *TEXT / *COMMENT
,PROTECTION = *LMS-DEFAULT / *STD / *BY-SOURCE
,WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *EXTEND / *ANY
,DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
```

(part 3 of 3)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(..)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(..)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be extracted from the library and included in a file.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be output.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is used.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is output.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be output.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be output.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be output has any date.

USER-DATE = *TODAY
The member with the current date is output.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is output.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are output.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are not to be output. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the LIBRARY-ELEMENT operand of this statement.

TO-FILE = *STD / *BY-SOURCE /
<filename 1..54 without-gen-vers with-wild-constr> / *LINK(...)
Name of the target file. A construction specification references the member name.

TO-FILE = *STD
Unless otherwise specified, the member data is output to a file which is given the same name as the member. However, if the EXTRACT-ELEMENT statement comes from the EDT command line, the data is written by default to the current EDT work file.

TO-FILE = *BY-SOURCE
The file name is the same as the member name.

TO-FILE = *LINK(...)
The member is output to the file that was assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library which was declared with a /ADD-FILE-LINK command before LMS was called and which must be known to LMS.

FILE-ATTRIBUTES = *BY-ELEMENT / *BY-CATALOG / *LMS-DEFAULT / *PARAMETERS(...)
File attributes that are defined when the file is created. This operand has no effect if the member data is written to the current EDT work file. LMS defines the file attributes in accordance with the following hierarchy:

1. LINK entry
2. file attributes stored in the member
3. catalog entry
4. LMS default values

The following specifications take effect only when TO-FILE=*LINK has not been specified.

FILE-ATTRIBUTES = *BY-ELEMENT
The file attributes stored in the member take priority.

FILE-ATTRIBUTES = *BY-CATALOG
The attributes stored in the catalog entry take priority. If there is no catalog entry, specifying *BY-CATALOG has the same effect as *BY-ELEMENT.

FILE-ATTRIBUTES = *PARAMETERS(...)

ACCESS-METHOD = *LMS-DEFAULT / *ISAM / *SAM
Specifies the access method ISAM or SAM for the target file.

INFORMATION = *TEXT / list-poss(2): *TEXT / *COMMENT
The section of the member which is to be processed.

INFORMATION = *TEXT
The text itself, i.e. record type 1, is to be output.

INFORMATION = *COMMENT
The separately stored comment, i.e. record type 2, is to be output.

PROTECTION = *LMS-DEFAULT / *STD / *BY-SOURCE
Setting and activation of an access protection mechanism for the file created; this protection corresponds to the member protection in effect for the member. This operand has no effect if the member data is written to the current EDT work file.

PROTECTION = *STD
The member protection in effect for the member is not taken into account in setting the access protection mechanism for the file which is created.

PROTECTION = *BY-SOURCE
The file created is provided with an access protection mechanism corresponding to the member protection which is in effect for the member.

Note

With PROTECTION=*BY-SOURCE, the file is provided with the BACL access protection mechanism, even if it offers no additional protection to the member's access rights (read, write, execute).
If desired, the values of the ACCESS and USER-ACCESS file attributes (record type 164), which may have been stored in the member (see FILE-ATTRIBUTES), can be used to set the values of the default access control mechanism, regardless of the entry in the PROTECTION operand.

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a file having the same name. If the file does not exist under this name, it will be created as a new file. If the member data is written to the current EDT work file, this operand has no effect, i.e. if the current work file already contains data, the member data is appended to it, and if the current work file is empty, the member data is written at the beginning.

WRITE-MODE = *CREATE
The new file must not yet exist and is created as a new file.

WRITE-MODE = *REPLACE
The file must already exist and is replaced.

WRITE-MODE = *EXTEND
The file is extended if it already exists. Otherwise, it will be created as a new file.

WRITE-MODE = *ANY
The file is replaced if it already exists. Otherwise it will be created as a new file.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0053	Member and file attributes different
2	0	LMS0129	Statement aborted by user
2	0	LMS0199	Record length invalid with fixed record format
2	0	LMS0274	Block control value changed
2	0	LMS0286	File attributes not modified
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0035	Member protection not transferrable to file
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For LIBRARY-ELEMENT: read authorization for LIBRARY and ELEMENT

If more than one member is affected by the statement, members without read authorization are excluded from the statement.

For TO-FILE: read and write authorization for the file

With PROTECTION=*BY-SOURCE, members can be output as files only for the user’s own ID, and only the owner of the library file can use this functionality.

Note

If PROTECTION=*BY-SOURCE is specified, bear in mind the following:

This functionality is rejected if GUARD protection is set for one or more types of access (read, write, execute) to the member and no special protection or BACL protection deviating from USER=NONE is set for at least one type of access (r, w, x).

Existing passwords in member protection are not taken into account.

When PROTECTION=*BY-SOURCE is specified and members are output to existing files provided with additional protection by the protection attributes of a password, these attributes are set to NONE (any existing password is deleted).

Examples

– Member ELEM1 is output via EXTRACT-ELEMENT to file TEST with the specified file attributes.

```
/ADD-FILE-LINK FILE-NAME=TEST,LINK-NAME=OUT,ACCESS-METHOD=*SAM,-
/              RECORD-FORMAT=*VARIABLE
/START-LMS
//MOD-LOGG-PAR LOGG=*MAX
//OPEN-LIBRARY LIBRARY=LIBIN
//EXTRACT-ELEMENT *LIB(LIBIN,ELEM1,S),TO-FILE=*LINK(LINK-NAME-OUT)
INPUT  LIBRARY= :1OSQ:$USER.LIBIN
OUTPUT FILE
       EXTRACT (S)ELEM1/@(0001)/2011-03-27 AS :1OSQ:$USER.TEST
       , REPLACED EMPTY FILE
```

– If all the members in a library are to be output by name, the following statement must be specified:

```
//EXTRACT-ELEMENT *LIB(ELEM=*ALL,TYPE=*ALL)
INPUT  LIBRARY= :1OSQ:$USER.LIBIN
OUTPUT FILE
       EXTRACT (S)ELEM1/@(0001)/2011-03-27 AS :1OSQ:$USER.ELEM1
       EXTRACT (S)EXT.FILE.1/@(0001)/2011-03-27 AS :1OSQ:$USER.EXT.FILE.1
       EXTRACT (S)EXT.FILE.2/@(0001)/2011-03-27 AS :1OSQ:$USER.EXT.FILE.2
       EXTRACT (S)EXT.FILE.3/@(0001)/2011-03-27 AS :1OSQ:$USER.EXT.FILE.3
       EXTRACT (S)EXT.FILE.4/@(0001)/2011-03-27 AS :1OSQ:$USER.EXT.FILE.4
       EXTRACT (S)EXT.FILE.5/@(0001)/2011-03-27 AS :1OSQ:$USER.EXT.FILE.5
//END
```

FIND-ELEMENT

The FIND-ELEMENT statement uses a specified search string to search for textual members in the member records, and logs the members it finds, optionally with or without the hit records.

FIND-ELEMENT

ELEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

LINK-NAME = <structured-name 1..8> / <filename 1..8>

,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEFAULT** / ***ALL** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 1 of 3)

,**MODIFICATION-DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>

,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>

,**EXCEPT-ELEM**ENT = **<u>*NONE</u>** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = **<u>*ANY</u>** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = **<u>*ANY</u>** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = **<u>*STD</u>** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = **<u>*ANY</u>** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = **<u>*STD</u>** / <composed-name 1..24 with-under with-wild>

,**TYPE** = **<u>*ANY</u>** / ***LMS-DEFAULT** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>

,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>

,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>

,**MODIFICATION-DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>

,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>

(part 2 of 3)

```
,PATTERN = <c-string 1..256> / *LOWER-CASE(...)
    *LOWER-CASE(...)
      | PATTERN = <c-string 1..256 with-low>
,SHOW-RECORDS = *YES (...) / *NO
    *YES(...)
      | RECORD-NUMBER = *NO / *YES
,INFORMATION = *STD / *ALL / list-poss(2): *TEXT / *COMMENT
TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
    *SYSLST(...)
      | SYSLST-NUMBER = *STD / <INTEGER 1..99>
    *EDT(...)
      | WRITE-MODE = *EXTEND / *REPLACE
,STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
    <composed-name 1..255>(...)
      | WRITE-MODE = *REPLACE / *EXTEND
```

(part 3 of 3)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the member to be searched.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library to be searched.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library from which the member is to be taken.

LIBRARY = *LINK(..)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL (...)/ <composed-name 1..64 with-under with-wild(132)>(...)
Specifies the member to be searched.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be searched.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is used.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is searched.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be searched.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=*HIGHEST-EXISTING, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be searched.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be searched has any date.

USER-DATE = *TODAY
The member with the current date is searched.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is searched.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are searched.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification of the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT =*NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded from the search.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are to be excluded from the search. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see *LIBRARY-ELEMENT.

PATTERN = <c-string 1..256> / *LOWER-CASE(...)
Character string (regular expression) on which the search is based (see the „C Library Functions (BS2000/OSD) for POSIX Applications“ manual [16]). The use of upper and lower case characters is ignored.

PATTERN = *LOWER-CASE(...)
No conversion of lower case characters.

PATTERN = <c-string 1..256 with-low>
Character string as described above, but in this case the use of upper and lower case characters is distinguished.

SHOW-RECORDS = *YES(...) / *NO
Specifies whether the hit records are shown or not.

SHOW-RECORDS = *YES(...)
The hit records are shown. Every member found is output in a header, followed by the records in which it was found (similar to the SHOW-ELEMENT output). The number of records found and the number of records searched are also displayed for each member. Finally, the number of members found and the number of members searched are output.

RECORD-NUMBER = *NO / *YES
Specifies whether or not the record numbers are output.

RECORD-NUMBER = *NO
No record numbers are output.

RECORD-NUMBER = *YES
The record numbers are also output.

SHOW-RECORDS = *NO
The hit records are not shown. The members found are output as with SHOW-ELEMENT-ATTRIBUTES ... INFORMATION=*MEDIUM. The number of members found and the number of members searched are also output.

INFORMATION = *STD / *ALL / list-poss(2): *TEXT / *COMMENT
The section of the member which is to be searched.

INFORMATION = *STD
Has the same effect as *TEXT for text members, otherwise as *ALL.

INFORMATION = *ALL
The whole member, i.e. record types 1-159 and 164, is searched.

INFORMATION = *TEXT
The text itself, i.e. record type 1, is searched.

INFORMATION = *COMMENT
The separately stored comment, i.e. record type 2, is searched.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified using //MODIFY-LOGGING-PARAMETERS, TEXT-OUTPUT=.

TEXT-OUTPUT = *NONE
The log output is suppressed, except for any error messages.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
The output is written to work file 9 of EDT. In the event of a error during log output, the system switches to the default log stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output is added to this data. If not, then the oputput will be written starting at the beginning of the work file.

WRITE-MODE = *REPLACE
Output is written at the beginning of work file 9. Any data already contained in the work file will be replaced.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Structured output of the members found.

STRUCTURE-OUTPUT = *SYSINF
The structured output is placed in the SYSINF stream assigned by /ASSIGN-STREAM (see the “SDF-P“ manual [12]).

STRUCTURE-OUTPUT = *NONE
There is no structured output.

STRUCTURE-OUTPUT = <composed-name 1..255>(...)
Specifies the S variable in which the structured output is to be placed. This variable must be declared as a dynamic list variable.
(Command: `/DECLARE-VARIABLE NAME=...(TYPE =*STRUCTURE),MULTIPLE-ELEMENTS =*LIST`)

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the list variable is to be overwritten or extended.

WRITE-MODE = *REPLACE
The existing contents of the list variable are overwritten.

WRITE-MODE = *EXTEND
The new list members are appended to the existing list.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For ELEMENT: read authorization for LIBRARY and ELEMENT

If more than one member is affected by the statement, members without read authorization are excluded from the statement.

Examples

– The members of library X are to be searched for the string “abc”.

```
//FIND-ELEMENT *LIB(LIB=X,ELEM=*,TYPE=S),PATTERN='abc',SHOW-RECORDS=*YES
INPUT  LIBRARY= :1OSQ:$USER.X
INPUT  ELEMENT= (S)FIND/@(0001)/2012-10-09
 abc is in this record.
This is a record with abcdefg.
       2 OUT OF      127 RECORD(S) FOUND
INPUT  LIBRARY= :1OSQ:$USER.X
       1 OUT OF        2 (S)-ELEMENT(S) FOUND
//FIND-ELEMENT *LIB(LIB=X,ELEM=*,TYPE=S),PATTERN='abc', -
SHOW-RECORDS=*YES(,RECORD-NUMBER=*YES)
INPUT  LIBRARY= :1OSQ:$USER.X
INPUT  ELEMENT= (S)FIND/@(0001)/2012-10-09
      #7  > abc is in this record.
    #123  >This is a record with abcdefg.
       2 OUT OF      127 RECORD(S) FOUND
INPUT  LIBRARY= :1OSQ:$USER.X
       1 OUT OF        2 (S)-ELEMENT(S) FOUND
//FIND-ELEMENT *LIB(LIB=X,ELEM=*,TYPE=S),PATTERN='abc',SHOW-RECORDS=*NO
INPUT  LIBRARY= :1OSQ:$USER.X
TYP NAME VER (VAR#) DATE
(S) FIND @   (0001) 2012-10-09
       1 OUT OF        2 (S)-ELEMENT(S) FOUND
```

– L-type member LLM from library X is to be searched for the string ’SYSLNK’ and 5 characters in front of the hit, together with all characters after the hit, are to be displayed (one screen line maximum).

```
//FIND-ELEMENT *LIB(LIB=X,ELEM=LLM,TYPE=L),PATTERN=’.....SYSLNK.*’
INPUT  LIBRARY= :1SOSQ:$USER.X

INPUT  ELEMENT= (L)LLM/@(0001)/2012-10-09
RECORD-TYPE: 160
   $.SYSLNK.LMS.034
MSLMSSYSLNK.LMS
       2 OUT OF        7 RECORD(S) FOUND
INPUT  LIBRARY= :1OSQ:$USER.X
       1 OUT OF        1 (L)-ELEMENT(S) FOUND
```

MODIFY-ELEMENT

The MODIFY-ELEMENT statement initiates the modification of members. The modifications themselves are controlled by way of MODIFY-ELEMENT substatements (see page 287).

MODIFY-ELEMENT selects the members to be modified. The source and target base types may differ if text members are modified.

Once the MODIFY-ELEMENT statement has been sent, LMS expects a substatement as the next statement. If another statement is entered instead of a substatement, an error message is issued.

MODIFY-**ELEM**ENT

ELEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / <alphanum-name 1..8>

(part 1 of 3)

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 2 of 3)

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**TO-E**LEMENT = ***LIB**RARY-**ELEM**ENT (...)
***LIB**RARY-**ELEM**ENT(...)
LIBRARY = ***STD** / ***BY-SOUR**CE / <filename 1..54 without-vers> / ***LINK**(...)
***LINK**(...)
LINK-NAME = <structured-name 1..8>
,**ELEM**ENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)
***BY-SOUR**CE(...)
VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
<composed-name 1..132 with-under with-wildcard-constr>(...)
VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / <alphanum-name 1..8>
,**USER-DATE** = ***TODAY** / ***BY-SOUR**CE / <date 8..10 with-compl>
,**TEXT-PAR**AMETERS = ***NONE** / ***PAR**AMETERS(...)
***PAR**AMETERS(...)
INPUT-RECORD-**ID** = ***NONE** / ***RECORD-PART**(...)
***RECORD-PART**(...)
START = <integer 1..251>
,**LEN**GTH = <integer 1..16>
,**WR**ITE-**MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***ANY**
,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

(part 3 of 3)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD/ <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member to be modified.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member to be modified.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
Specifies the member to be modified.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be modified.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is modified.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be modified.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / <alphanum 1..8>
Type of the member to be modified.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be modified has any date.

USER-DATE = *TODAY
The member with the current date is modified.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is modified.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are modified.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded from correction.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are to be excluded from modification. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the destination to which and the name under which the corrected member is to be written back.

LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
Specifies the library to which the corrected member is to be written back.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *BY-SOURCE
The corrected member is written back to the original library.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the corrected member is to be added.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wild-constr>(...)
Name that the corrected member is to receive.

ELEMENT = *BY-SOURCE(...)
The new name is the same as the old name.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the corrected member is to receive.

VERSION = *BY-SOURCE
The corrected member receives the same version as the original member. If the original member has no version specification, the corrected member receives X'FF' as the version specification.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..52 with-under with-wild-constr>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Name of the corrected member. It can also be entered using wildcards.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the corrected member is to receive.
For a description of the operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

TYPE = *BY-SOURCE /*LMS-DEFAULT / <alphanum-name 1..8>
Type that the corrected member is to receive.

TYPE = *BY-SOURCE
The corrected member receives the same type designation as the original member.

USER-DATE = *TODAY / *BY-SOURCE / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *TODAY
The current date is given.

USER-DATE = *BY-SOURCE
The new member is given the same date as the source member.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

TEXT-PARAMETERS = *NONE / *PARAMETERS(...)
Specifies parameters for textual members.

TEXT-PARAMETERS = *NONE
No parameters are specified for textual members.

TEXT-PARAMETERS = *PARAMETERS(...)
Specifies parameters for textual members.

INPUT-RECORD-ID = *NONE / *RECORD-PART(...)
Specifies the location of the record ID (see substatements for textual members on page 287) in the input record.

INPUT-RECORD-ID = *NONE
No location is specified for the record ID of the input record.

INPUT-RECORD-ID = *RECORD-PART(...)
Specifies the beginning and length of the record ID area, where beginning + length ≤ 252

START = <integer 1..251>
Specifies the first character in the record ID to indicate the beginning of the record ID area.

LENGTH = <integer 1..16>
Specifies the length of the record ID.

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.
If the source member is the same as the target member, this operand is ignored.

WRITE-MODE = *CREATE
The name of the corrected member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The corrected member must already exist and is replaced.

WRITE-MODE = *ANY
The corrected member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement on page 330.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For ELEMENT: read authorization for LIBRARY and ELEMENT
If more than one member is affected by the statement, members without read authorization are excluded from the statement.

For TO-ELEMENT: read and write authorization for LIBRARY
Administer authorization where the specified member designation is new. Otherwise, only write authorization for the member existing under the specified member designation (administer authorization no longer required).

If WRITE-CONTROL is active and a base version exists, the USERID of the user must be entered as the HOLDER of the member specified by BASE. Only if write authorization has been granted can a new version be generated or this base version overwritten. In this case, administer authorization is no longer required.

Notes

– When creating a member, be sure to take into account the convention applicable to the member type.

– If WRITE-CONTROL is active in the output library, the access method adds a comment (record type 2) to the member which is to be written. The comment logs the HOLDER, DATE and TIME of the operation. The record is written as the first record of the record type. Any comment records which already exist or have been edited by means of INFORMATION=*COMMENT are copied after it. If, in addition, the member is written to the base of a different version (i.e. not the first version under a name), the member attributes STATE and HOLDER and all the rights of the base version are adopted for the new version. The CCSN is adopted from the source file. The USER-DATE is determined anew.

MODIFY-ELEMENT substatements for member types R, C and L

These substatements are valid for members of the R, C and L base types and must directly follow the MODIFY-ELEMENT statement. They are read from the statement stream until the END-MODIFY substatement is encountered.

The MODIFY-ELEMENT substatements make modifications to object modules, phases and link and load modules (LLM).

LMS initially collects these substatements and executes them only after the END-MODIFY substatement has been entered.

Overview of MODIFY-ELEMENT substatements

These substatements are dependent on the selected member type and are permitted only for members of types R, C or L.

MODIFY-ELEMENT statement	Member type	Function
ADD-REP-RECORD	R	Generate REP records
ADD-TEXT-MODIFICATION	R,C,L	Modify text records
DELETE-RECORD-TYPE	R,C,L	Delete record types
END-MODIFY	R,C,L	Terminate modifications
MODIFY-CSECT-ATTRIBUTES	R	Modify CSET attributes
MODIFY-MODIFICATION-DEFAULTS	R,C,L	Define global parameters in the MODIFY-ELEMENT statement
REMOVE-MODIFICATION	R,C,L	Cancel corrections
RENAME-SYMBOLS	R	Rename CSECTs, ENTRYs, EXTRNs and COMMONs

Note

The SDF standard statements (see page 137) are also permitted as MODIFY-ELEMENT substatements.

ADD-REP-RECORD

The MODIFY-ELEMENT substatement ADD-REP-RECORD adds REP records to the object module. These REP records are evaluated by the dynamic binder loader (DBL).

ADD-REP-RECORD is permitted only for object modules (R-type members).

```
ADD-REP-RECORD

 ADDRESS = <x-string 1..8>(...)
    <x-string 1..8>(...)
       | BASE-ADDRESS = *MODIFICATION-DEFAULT / <x-string 1..8>
,NEW-CONTENTS = <x-string 1..100> / <c-string 1..50 with-low>
```

ADDRESS = <x-string 1..8>(...)
Specifies the address at which the member selected by MODIFY-ELEMENT is to be modified.

BASE-ADDRESS = *MODIFICATION-DEFAULT / <x-string 1..8>
Base address. BASE-ADDRESS is added to ADDRESS. The resulting correction address must, in the case of prelinked modules, relate to the prelinked module (and not to the CSECT).

NEW-CONTENTS = <x-string 1..100> / <c-string 1..50 with-low>
Replacement text specified in character or hexadecimal form.
If the text is specified in character form, it must not exceed 50 characters in length. An apostrophe in the text must be duplicated.
If the text is specified in hexadecimal form, it must not exceed 100 characters in length.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

ADD-TEXT-MODIFICATION

The MODIFY-ELEMENT substatement ADD-TEXT-MODIFICATION corrects text records of an object module and phases. This substatement generates a correction journal record (TXTP record) that contains the original contents of the text area.
With the MODIFY-MODIFICATION-DEFAULTS statement, you can specify that no correction journal record is to be created. Corrections without a correction journal record cannot be reversed by the REMOVE-MODIFICATION substatement.

This substatement may be used for members of types R, C and L.

```
ADD-TEXT-MODIFICATION

 ADDRESS = <x-string 1..8>(...)
     <x-string 1..8>(...)
         | BASE-ADDRESS = *MODIFICATION-DEFAULT / <x-string 1..8>
,NEW-CONTENTS = <x-string 1..100>(...) / <c-string 1..50 with-low>(...)
     <x-string 1..100>(...)
         | OLD-CONTENTS = *ANY / <x-string 1..100> / <c-string 1..50 with-low>
     <c-string 1..50 with-low>(...)
         | OLD-CONTENTS = *ANY / <x-string 1..100> / <c-string 1..50 with-low>
,MODIFICATION-ID = *MODIFICATION-DEFAULT / *SPACES /  <c-string 1..12 with-low>
```

ADDRESS = <x-string 1..8>(...)
Specifies the address at which the member selected by MODIFY-ELEMENT is to be modified.

BASE-ADDRESS = *MODIFICATION-DEFAULT / <x-string 1..8>
Base address.

The base address is added to ADDRESS. The resulting correction address is as follows:

For ...	Relative to:
modules	start of CSECT (the desired CSECT is specified via the MODIFY-MODIFICATION-DEFAULTS substatement)
phases	start of phase
LLMs	start of CSECT if a CSECT has been specified

For ...	Relative to:
	start of sub-LLM if a sub-LLM has been specified (the desired sub-LLM is specified via the MODIFY-MODIFICATION-DEFAULTS substatement)
	start of slice if a slice has been specified (the desired slice is specified via the MODIFY-MODIFICATION-DEFAULTS substatement)
	start of LLM if nothing has been specified and the LLM only consists of one slice. If the LLM consists of more than one slice, you must specify a CSECT, a sub-LLM or a slice.

NEW-CONTENTS = <x-string 1..100>(...) / <c-string 1..50 with-low>(...)
Replacement text specified in character or hexadecimal form.

OLD-CONTENTS = *ANY / <x-string 1..100> / <c-string 1..50 with-low>
Original text of the member. The original text must always be specified the same length as the replacement text.

MODIFICATION-ID = *MODIFICATION-DEFAULT / *SPACES / <c-string 1..12 with-low>
Identification which is held in the correction journal record (TXTP record). If SPACES is specified, blanks are used as the identification.
Only 8 characters are allowed for member types R and C.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

DELETE-RECORD-TYPE

The MODIFY-ELEMENT substatement DELETE-RECORD-TYPE excludes the following record types from the input member:

– ISD records (applies only to R-type members)
– LSD records (applies only to R-type members)
– REP records (applies only to R-type members)
– INCLUDE records (applies only to R-type members)
– TXTP records (applies only to R-, C- and L-type members)
– DSDD records (applies only to R-type members)

This substatement may be used for members of types R, C and L.

DELETE-**REC**ORD-**TYPE**

TYPE = ***TXTP**(...) / list-poss(5): ***ISD** / ***LSD** / ***REP** / ***DSDD** / ***INCL**UDE

***TXTP**(...)

MODIFICATION-**ID** = ***ALL** / ***SPACE**S / <c-string 1..12 with-low>

TYPE = *TXTP(...) / list-poss(5): *ISD / *LSD / *REP / *DSDD / *INCLUDE
Defines the record type that is not to be transferred from the input member to the output member.

MODIFICATION-ID = *ALL / *SPACES / <c-string 1..12 with-low>
Only those TXTP records with the specified identification are deleted.
For member types R and C, only 8 characters are allowed.
This identification applies only to this DELETE-RECORD-TYPE.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Note

Deleted record types cannot be retrieved.

END-MODIFY

END-MODIFY concludes the sequence of MODIFY-ELEMENT substatements. LMS then checks all the statements for executability and executes the statement string.

END-MODIFY

This substatement has no operands.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

MODIFY-CSECT-ATTRIBUTES

The MODIFY-ELEMENT substatement MODIFY-CSECT-ATTRIBUTES modifies CSECT attributes.

This substatement must be used for object modules (R-type members) only.

MODIFY-**CSECT-ATTR**IBUTES
NAME = ***ALL** / <c-string 1..8 with-low> / <text 1..8>
,**VISIBLE** = ***UNCHA**NGED / ***Y**ES / ***NO**
,**READ-ONLY** = ***UNCHA**NGED / ***Y**ES / ***NO**
,**PAGE-ALIGN**MENT = ***UNCHA**NGED / ***Y**ES / ***NO**
,**RESID**ENCY-**MODE** = ***UNCHA**NGED / **24** / ***ANY**
,**ADDR**ESSING-**MODE** = ***UNCHA**NGED / **24** / **31** / ***ANY**

At the beginning of the MODIFY-ELEMENT statement, the operands are preset to the value immediately following *UNCHANGED.

NAME = *ALL / <c-string 1..8 with-low> / <text 1..8>
Name of the CSECT whose attributes are to be modified. All CSECTs or a specific CSECT can be specified.

VISIBLE = *UNCHANGED / *YES / *NO
Masking (visibility) of CSECTs.

VISIBLE = *YES
The specified control sections are not masked (see [4]).
A secondary name record is created for these sections, and the names are entered in the directory of secondary names.

VISIBLE = *NO
The specified control sections are masked. No secondary name record is created for these sections, and the names are not entered in the directory of secondary names. Any secondary name record which may exist is deleted.

If all control sections of an object module are masked, a library member without a secondary name entry is created. This object module can be located via the primary name only.

The module name can, however, be derived from the initial control section name with the aid of all ESD records, since masked control sections are also used in this case.

Note

The linkage editor cannot process object modules which only have masked control sections, e.g. when an object module is excluded with the autolink function.
The VISIBLE operand can also be used on ENTRYs.

READ-ONLY = *UNCHANGED / *YES / *NO
Write protection.

READ-ONLY = *YES
Indicates that only read access to the specified control sections is permitted while the program is executing.

READ-ONLY = *NO
Enables write access to the specified control sections even while the program is executing.

PAGE-ALIGNMENT = *UNCHANGED / *YES / *NO
Page alignment.

PAGE-ALIGNMENT = *YES
Indicates that the specified control sections are to be aligned on a page boundary, i.e. the load address should be a multiple of decimal 4096 or hexadecimal 1000.

PAGE-ALIGNMENT = *NO
Does not take page boundaries into account. The control sections always start at the next doubleword address produced during the linkage process.

RESIDENCY-MODE = *UNCHANGED / 24 / *ANY
Load mode.

RESIDENCY-MODE = 24
Indicates that the specified control sections are to be loaded to the address area below the 16-Mbyte limit.

RESIDENCY-MODE = *ANY
No limitation exists.

ADDRESSING-MODE = *UNCHANGED / 24 / 31 / *ANY
Execution mode.

ADDRESSING-MODE = 24
Indicates that the specified control sections are to be executable in 24-bit mode.

ADDRESSING-MODE = 31
Indicates that the specified control sections are to be executable in 31-bit mode.

ADDRESSING-MODE = *ANY
Any execution mode.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

MODIFY-MODIFICATION-DEFAULTS

The MODIFY-ELEMENT substatement MODIFY-MODIFICATION-DEFAULTS defines the global default values within the MODIFY-ELEMENT statement.

This substatement may be used for members of types R, C and L.

```
MODIFY-MODIFICATION-DEFAULTS

 CSECT-NAME = *UNCHANGED / *NONE / <c-string 1..32 with-low> / <text 1..32>
,PHASE-SEGMENT = *UNCHANGED / *ROOT /  <name 1..8>
,LLM-PART = *UNCHANGED / *NONE / *SLICE(...) / *SUB-LLM(...)
    *SLICE(...)
      | NAME = <structured-name 1..32>
    *SUB-LLM(...)
      | PATH-NAME = <c-string 1..255 with-low> / <text 1..255>
,MODIFICATION-LOGGING = *UNCHANGED / *YES(...) / *NO
    *YES(...)
      | MODIFICATION-ID = *UNCHANGED / *SPACES / <c-string 1..12 with-low>
,BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
```

At the beginning of the MODIFY-ELEMENT statement, the operands are preset to the value immediately following *UNCHANGED.

CSECT-NAME = *UNCHANGED / *NONE / <c-string 1..32 with-low> / <text 1..32>
Name of the CSECT to be corrected (relevant only for types R and L).

CSECT-NAME = *NONE
If no CSECT name is specified, in the case of R-type modules the first CSECT name is used.

PHASE-SEGMENT = *UNCHANGED / *ROOT / <name 1..8>
Specifies the phase segment to be corrected. If no segment is specified, the first segment (*ROOT) is used.

LLM-PART = *UNCHANGED / *NONE / *SLICE(...) / *SUB-LLM(...)
If no LLM part is specified, the entire LLM is used.

LLM-PART = *SLICE(...)
Specifies the slice to be corrected.

NAME = <structured-name 1..32>
Name of the slice to be corrected.

LLM-PART = *SUB-LLM(...)
Specifies the sub-LLM to be corrected.

PATH-NAME = <c-string 1..255 with-low> / <text 1..255>
The sub-LLM to be corrected is determined by way of its path name.

MODIFICATION-LOGGING = *UNCHANGED / *YES(...) / *NO
Defines TXTP record generation.

MODIFICATION-LOGGING = *YES(...)
TXTP records are to be generated.

MODIFICATION-ID = *UNCHANGED / *SPACES / <c-string 1..12 with-low>
Identification which is held in the correction journal record (TXTP record). If SPACES is specified, blanks are used as the identification.
For member types R and C, only 8 characters are allowed.

MODIFICATION-LOGGING = *NO
No TXTP records are to be generated.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
Hexadecimal specification of the base address. At the beginning of the MODIFY-ELEMENT statement, base address 0 is set.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

REMOVE-MODIFICATION

The MODIFY-ELEMENT substatement REMOVE-MODIFICATION cancels corrections from a previous correction run under the following preconditions:

A correction journal record was created with the MODIFY-ELEMENT substatement ADD-TEXT-MODIFICATION, i.e. the operand MODIFICATION-LOGGING = *YES (see page 297) was set.

This substatement may be used only for members of types R, C and L.

REMOVE-**MOD**IFICATION
MODIFICATION-**ID** = ***ALL** / ***SPACE**S / <c-string 1..8 with-low>

MODIFICATION-ID = *ALL / *SPACES / <c-string 1..12 with-low>
For member types R and C, only 8 characters are allowed.
Only those corrections with the specified identification are cancelled. If an identification is specified, it is necessary that correction journal records for it exist. If no identification (*ALL) is specified, all corrections for which a correction journal record exists are cancelled.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

RENAME-SYMBOLS

The MODIFY-ELEMENT substatement RENAME-SYMBOLS changes the name of a CSECT, ENTRY, COMMON, EXTRN, WXTRN or a V constant. Each renaming results in a modification of the ESD records. LMS checks for the uniqueness of names within all ESD records, rejecting a new name if that name already exists.

In LSD records, no renaming occurs. This is why, following a change in the CSECT name, AID can no longer be used for symbolic testing (see [13]). If testing is run near machine level, the new names must be used in any AID qualifications.

The MODIFY-ELEMENT substatement RENAME-SYMBOLS may be used only for object modules (R-type members).

RENAME-**SYMB**OLS
SYMBOL-NAME = <text 1..8> ,**SYMBOL-TYPE** = ***CSECT** / ***ENTRY** / **"COMMON** / ***EXTRN** / ***VCON** / ***WXTRN** ,**NEW-NAME** = <text 1..8>

SYMBOL-NAME = <text 1..8>
Defines the symbol name to be renamed.

SYMBOL-TYPE = *CSECT / *ENTRY / *COMMON / *EXTRN / *VCON / *WXTRN
Defines the type of symbol whose name is to be changed.

NEW-NAME = <text 1..8>
New symbol name.
The name should satisfy the BINDER conventions for the special data type <symbol> (see [4]). LMS does not check for this convention, however.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

Note

Masked (invisible) CSECT/ENTRY names can also be renamed.

MODIFY-ELEMENT substatements for textual members

These substatements make changes to text members. They are read from the statement stream until the MODIFY-ELEMENT substatement END-MODIFY is encountered.

Overview of MODIFY-ELEMENT substatements

These substatements may be used only for textual members.

MODIFY-ELEMENT substatement	Function
ADD-RECORD	Inserts records
END-MODIFY	Concludes the modifications
REMOVE-RECORD	Removes records

Notes

– Standard SDF statements are also permitted as MODIFY-ELEMENT substatements (see page 137).

– Only member records with lengths ≤ 251 are processed. Longer records will be truncated. In this case, LMS issues a warning.

Definition of record identification for textual members

Record identification may be a record number or a record ID.

Record number: The record number indicates the relative position of the member record in relation to the beginning of the member. If the record number specified is greater than the member's highest record number, the changes continue after the last member record, i.e. records are appended to the member.

Record ID: The location and length of the record ID are specified by means of the INPUT-RECORD-ID operand (see the MODIFY-ELEMENT statement). This is why it is not permissible to specify a record ID in substatements except when INPUT-RECORD-ID has a value other than *NONE. If specified, a record ID must have the length declared in INPUT-RECORD-ID. Only leading zeros may be omitted.
If the record ID does not occur in the input member, the changes are inserted in front of the first record with a higher record ID.

Record numbers and record IDs may be mixed within substatements. In substatements and data records, they must always be specified in ascending order.
If an error is detected in interactive mode, the correction must be terminated with END-MODIFY and then restarted. After an ADD-RECORD substatement, an *END must also be entered.

ADD-RECORD

The MODIFY-ELEMENT substatement ADD-RECORD inserts the records following the statement at the specified position. The records to be inserted must be concluded by an *END record.

ADD-RECORD
RECORD-**ID** = ***NONE** / <integer 0..99999999> / <c-string 1..16 with-low>

RECORD-ID = *NONE / <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the data record to be added.

RECORD-ID = *NONE
If the INPUT-RECORD-ID operand of the MODIFY-ELEMENT statement is set to a value other than *NONE, the data records following the ADD-RECORD substatement are inserted in the member being modified in accordance with their record IDs.
If a specified record ID designates a record which already exists, the data record is written over the existing record. If no record with the specified record ID yet exists, the data record is inserted in front of the first record with a higher record ID. If INPUT-RECORD-ID=*NONE is set or no record ID is specified for a data record, the record is inserted at the current position.

RECORD-ID = <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the member position after which the data records following the statement are to be inserted. If the specified record number or record ID does not exist, the data records are each inserted in front of the first record with a higher record number/record ID.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

END-MODIFY

Each sequence of MODIFY-ELEMENT substatements is concluded by an END-MODIFY substatement.

END-MODIFY

This substatement has no operands.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

REMOVE-RECORD

The MODIFY-ELEMENT substatement REMOVE-RECORD deletes the specified record or record area from the member.

REMOVE-**REC**ORD

RECORD-**ID** = <integer 0..99999999> / <c-string 1..16 with-low> / ***RANGE**(...)

***RANGE**(...)

FROM = <integer 0..99999999> / <c-string 1..16 with.low>

,TO = <integer 0..99999999> / <c-string 1..16 with.low>

RECORD-ID = <integer 0..99999999> / <c-string 1..16 with-low> / *RANGE(...)
Record number or record ID of the record to be deleted.

RECORD-ID = *RANGE(...)
Specifies the record area to be deleted.

FROM = <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the first record number or record ID of the area which is to be deleted.

TO = <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the last record number or record ID of the area which is to be deleted.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error

MODIFY-ELEMENT-ATTRIBUTES

This statement can be used to modify the member name, version, user date, CCS name and/or member state. The statement may be used for all member types.

The source and target member base types may differ if text members are processed.

It is not permitted to change a member designation for delta members or when WRITE-CONTROL is active.

MODIFY-**ELEM**ENT-**ATTR**IBUTES

ELEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***HIGH**EST-**EXIST**ING / ***ALL** / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CODED-CHAR**ACTER-**SET** = ***ANY** / ***NONE** / <name 1..8 with-wild(20)>

,**STATE** = ***ANY** / ***FREE** / ***IN-HOLD**(...)

***IN-HOLD**(...)

HOLDER = ***ANY** / <name 1..8 with-wild(20)>

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>

,**NEW-ATTR**IBUTES = **PAR**AMETERS (...)
***PAR**AMETERS(...)
ELEMENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)
***BY-SOUR**CE(...)
VERSION = ***BY-SOUR**CE / ***LMS-DEF**AULT / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
<composed-name 1..132 with-under with-wild-constr>(...)
VERSION = ***BY-SOUR**CE / ***LMS-DEF**AULT / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..52 with-under with-wild-constr>
,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>
,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / <alphanum-name 1..20 with-wild-constr>
,**USER-DATE** = ***BY-SOUR**CE / ***TODAY** / <date 8..10 with-compl>
,**MODIFICATION-DATE** = ***BY-SOUR**CE / ***SYSTEM-DATE**
,**CODED-CHAR**ACTER-**SET** = ***BY-SOUR**CE / ***LIB**RARY-**DEF**AULT / ***NONE** / <name 1..8>
,**STATE** = ***BY-SOUR**CE / ***FREE** / ***IN-HOLD**
,**WR**ITE-**MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***ANY**
,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(..)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL (...)/ <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member whose attributes are to be modified.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member.

VERSION = *HIGHEST-EXISTING
The attributes of the member with the highest existing version are modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' of the member in the library under the specified TYPE and name is used.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member has any date.

USER-DATE = *TODAY
The member with the current date is used.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is used.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are used.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

CODED-CHARACTER-SET = *ANY / *NONE / <name 1..8 with-wild(20)>
Character set assigned to the member.

CODED-CHARACTER-SET = *ANY
Selects members without regard to their assigned character set.

CODED-CHARACTER-SET = *NONE
Selects members which have not been assigned a character set.

CODED-CHARACTER-SET = <name 1..8> / <composed-name 1..8 with-wild(20)>
Selects the members to which the specified character set has been assigned.

STATE = *ANY / *FREE / *IN-HOLD(...)
State assigned to the member.

STATE = *ANY
Selects members without regard to their respective STATEs.

STATE = *FREE
Selects only members with STATE=FREE

STATE = *IN-HOLD(...)
Selects only members with STATE=IN-HOLD.

HOLDER = *ANY / <name1..8 with-wild(20)>
HOLDER assigned to the member.

HOLDER = *ANY
Selects members without regard to their respective HOLDERs.

HOLDER = <name 1..8> / <composed-name 1..8 with-wild(20)>
Selects only members which are assigned a HOLDER matching the pattern.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members whose attributes are to be excluded from modification. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

NEW-ATTRIBUTES = *PARAMETERS(...)
Specifies the attributes that the target member is to receive.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wild-constr>(...)
New name that the member is to receive.

ELEMENT = *BY-SOURCE(...)
The new name is the same as the old name. The member is not renamed.

VERSION = *BY-SOURCE / *LMS-DEFAULT / *HIGHEST-EXISTING /
***INCREMENT / *UPPER-LIMIT /**
<composed-name 1..52 with-under with-wild-constr>
Version that the member is to receive. Only the version is renamed, not the member.

VERSION = *BY-SOURCE
The new member receives the same version as the original member.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated (see also page 55).

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..52 with-under with-wild-constr>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
The new name of the member can also be entered using wildcards.

VERSION = *BY-SOURCE / *LMS-DEFAULT / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..52 with-under with-wild-constr>
Version that the member to be renamed is to receive. For description of operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

TYPE = *BY-SOURCE / *LMS-DEFAULT / <alphanum-name 1..20 with-wild-constr>
Type which the member being renamed is to receive.

TYPE = *BY-SOURCE
The member to be renamed receives the same type designation as the original member.

USER-DATE = *BY-SOURCE / *TODAY / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *BY-SOURCE
The member to be renamed receives the same date as the original member.

USER-DATE = *TODAY
The current date is given.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

MODIFICATION-DATE = *BY-SOURCE / *SYSTEM-DATE
Controls updating of the modification date.

MODIFICATION-DATE = *BY-SOURCE
The modification date is not to be updated.

MODIFICATION-DATE = *SYSTEM-DATE
Updates the modification date.

CODED-CHARACTER-SET = *BY-SOURCE / *LIBRARY-DEFAULT / *NONE / <name 1..8>
Character set assigned to the member.

CODED-CHARACTER-SET = *BY-SOURCE
The member is assigned the character set of the source member.

CODED-CHARACTER-SET = *LIBRARY-DEFAULT
The member is assigned the character set of the library containing the member.

CODED-CHARACTER-SET = *NONE
No character set is assigned to the member.

CODED-CHARACTER-SET = <name 1..8>
Specifies the character set which is to be assigned to the member.

STATE = *BY-SOURCE / *FREE / *IN-HOLD
State assigned to the member.

STATE = *BY-SOURCE
Leaves the state of the member unchanged.

STATE = *FREE
The new state of the member is FREE. Only the HOLDER and the owner of the library are permitted to change the state to FREE.

STATE = *IN-HOLD
The new state of the member is IN-HOLD. The USERID of the person who changed the state from FREE to IN-HOLD is entered as the HOLDER. If the state was already IN-HOLD, the statement is rejected. Only persons with hold authorization can be HOLDERs.

WRITE-MODE =
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member. If the type, name and version of the member remain unchanged, WRITE-MODE has the same effect as WRITE-MODE=*ANY.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

Read and write authorization for LIBRARY

Administer authorization is also required in order to change the member designation.

If an existing member designation is to be changed, the person making the change must have write authorization for the member which will be overwritten in the process.

Hold authorization is required in order to change a member’s state from FREE to IN-HOLD. Only the member’s HOLDER or the library owner are permitted to change the state of the member back to FREE.

Other attributes may be changed by persons having administer authorization or write authorization for the member involved, but not if STATE=*IN-HOLD is specified and WRITE-CONTROL is activated.

Notes

– When creating a member, be sure to take into account the convention applicable to the member type.

– If WRITE-CONTROL is activated, changes to member designations will not be allowed. This prevents members from being generated or overwritten without it also being documented when changes were made, and by whom.

Examples

The following examples assume that the member type is already preset.

– Modifying the member name from OLD to NEW:

```
//modify-element-attributes *lib(library=lib,elem=old), -
                                  new-attr=*par(elem=new)
```

– Modifying the version designation of the member “old” from V1 to V2:

```
//modify-element-attributes *lib(library=lib,elem=old(version=v1)), -
                                  new-attr=*par(*by-source(version=v2))
```

MODIFY-ELEMENT-PROTECTION

This statement modifies the protection for the specified members of the assigned library. The statement is permissible for all member types.

*UNCHANGED means that the respective member protection attributes remain unchanged.

```
MODIFY-ELEMENT-PROTECTION

ELEMENT = *LIBRARY-ELEMENT (...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
      *ALL(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
      <composed-name 1..64 with-under with-wild(132)>(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
    ,TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
    ,USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
      *INTERVAL(...)
        FROM = 1900-01-01 / <date 8..10 with-compl>
        ,TO = *TODAY / <date 8..10 with-compl>
```

(part 1 of 4)

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **1900-01-01** / <date 8..10 with-compl>
,**TO** = ***TODAY** / <date 8..10 with-compl>
,**PROT**ECTION = ***ANY** / ***NONE** / ***PAR**AMETERS(...)
***PAR**AMETERS(...)
READ = ***ANY** / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
***PAR**AMETERS(...)
USER = ***ANY** / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***ANY** / ***Y**ES / ***NO**
,**WR**ITE = ***ANY** / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
***PAR**AMETERS(...)
USER = ***ANY** / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***ANY** / ***Y**ES / ***NO**
,**EXEC** = ***ANY** / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
***PAR**AMETERS(...)
USER = ***ANY** / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***ANY** / ***Y**ES / ***NO**

(part 2 of 4)

,**HOLD** = ***ANY** / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)

***BY-GUARD**(...)

GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>

***PAR**AMETERS(...)

USER = ***ANY** / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**

,**PASS**WORD = ***ANY** / ***Y**ES / ***NO**

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

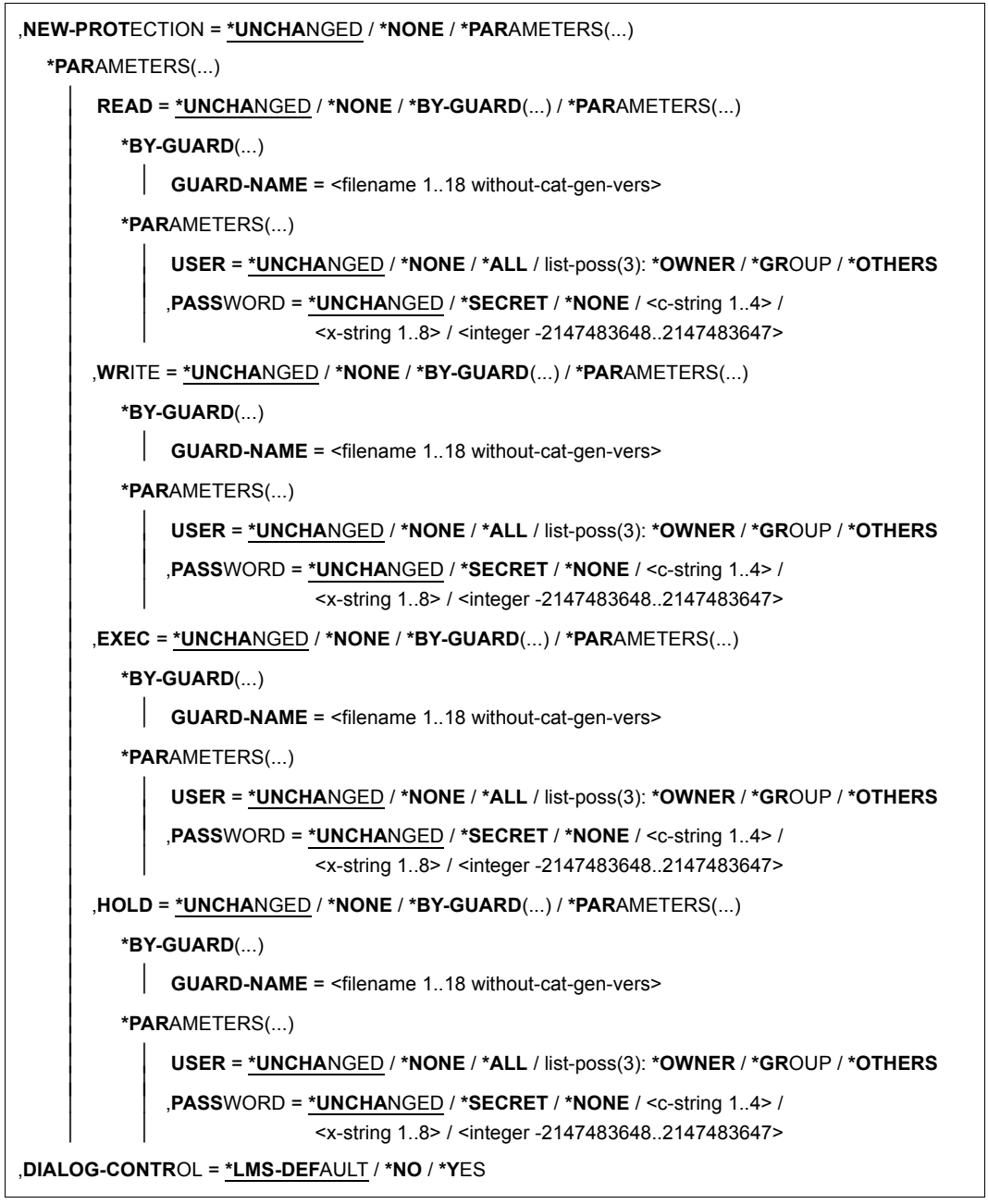
,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 3 of 4)


,**NEW-PROT**ECTION = ***UNCHA**NGED / ***NONE** / ***PAR**AMETERS(...)
***PAR**AMETERS(...)
READ = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
***PAR**AMETERS(...)
USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***UNCHA**NGED / ***SECRET** / ***NONE** / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
,**WR**ITE = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
***PAR**AMETERS(...)
USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***UNCHA**NGED / ***SECRET** / ***NONE** / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
,**EXEC** = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
***PAR**AMETERS(...)
USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***UNCHA**NGED / ***SECRET** / ***NONE** / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
,**HOLD** = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
***BY-GUARD**(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
***PAR**AMETERS(...)
USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
,**PASS**WORD = ***UNCHA**NGED / ***SECRET** / ***NONE** / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES

(part 4 of 4)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member whose protection attributes are to be modified.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member.

VERSION = *HIGHEST-EXISTING
The protection attributes of the member with the highest existing version are modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' of the member in the library under the specified TYPE and name is used.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*. For further information concerning specification of the base, see page 50.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member has any date.

USER-DATE = *TODAY
The member with the current date is used.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is used.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are used.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

PROTECTION = *ANY / *NONE / *PARAMETERS(...)
Member protection for the selected members.

PROTECTION = *ANY
The members have any member protection.

PROTECTION = *NONE
The members have no additional member protection.

PROTECTION = *PARAMETERS(...)
Specifies the protection with which the members to be selected are provided.

READ = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Read protection setting assigned to the member.

READ = *ANY
Selects members without regard to their respective read protection settings.

READ = *NONE
Selects only members which have no read protection.

READ = *BY-GUARD(...)
Selects only members which have GUARD read protection.

GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
Selects only members which have read protection by a GUARD-NAME matching the pattern.

READ = *PARAMETERS(...)
Selects only members which have read protection by BACL and/or password.

USER = *ANY / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
The read-authorized user group which is assigned to the member.

USER = *ANY
Selects members without regard to their respective read-authorized user groups.

USER = *NONE
Selects only members for which no read authorization has been granted.

USER = *OWNER
Selects only members for which the owner of the library file has read authorization.

USER = *GROUP
Selects only members for which the library file owner's group has read authorization.

USER = *OTHERS
Selects only members for which OTHERS read authorization has been granted.

PASSWORD = *ANY / *YES / *NO
Read password assigned to the member.

PASSWORD = *ANY
Selects members without regard to their respective passwords.

PASSWORD = YES
Selects only members which are protected by a read password.

PASSWORD = *NO
Selects only members which are not protected by a read password.

WRITE = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Write authorization. Selects only members for which this authorization has been granted in the specified manner.
The operands are analogous to those described above for READ.

EXEC = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Execute authorization. Selects only members for which this authorization has been granted in the specified manner.
The operands are analogous to those described above for READ.

HOLD = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Hold authorization. Selects only members for which this authorization has been granted in the specified manner.
The operands are analogous to those described above for READ.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members whose protection attributes are to be excluded from modification. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded. For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

NEW-PROTECTION = *UNCHANGED / *NONE / *PARAMETERS(...)
New member protection for the selected members.

NEW-PROTECTION = *NONE
The member receives no new or additional protection. It is protected only by the protection for the library file.

NEW-PROTECTION = *PARAMETERS(...)
Specifies which protection should now apply for the member.

READ = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
New read protection granted for the member.

READ = *NONE
No further access restriction is provided.

READ = *BY-GUARD(...)
Specifies the read guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Name of the guard.

READ = *PARAMETERS(...)
Modifies the user circles for the read authorization.

USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
The circle of those with read authorization is explicitly listed.

USER = *NONE
None may access in the specified manner.

USER = *ALL
All may access in the specified manner (full listing).

USER = *OWNER
The owner of the library file may access.

USER = *GROUP
Those belonging to the group of the owner of the library file may access.

USER = *OTHERS
All others may access.

PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
The circle of authorized persons is further restricted. In addition to the necessary access right, the correct password is required. Specifying 0 or X'00000000' does not result in any change of the last value.
Specifying *SECRET or ^ allows the desired password to be entered invisibly. If the "secret" value is entered as a c-string, it must be enclosed in apostrophes. If it is entered as an x-string, it must similarly be enclosed in apostrophes and also be prefixed by an X.

WRITE = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
New write authorization granted for the member.
The description of the operands is analogous to that for READ (see page 321).

EXEC = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
New execute authorization granted for the member.
The description of the operands is analogous to that for READ (see page 321).

HOLD = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
New hold authorization granted for the member.
The operands are analogous to those described for READ (see page 321).

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement, where the value *ERROR which might have been set there has the same effect as *NO. Likewise, the value *ERROR which may have been set for DIALOG-CONTROL= in the /SEND-MSG message command has the same effect as *NO with MODIFY-ELEMENT-PROTECTION.

Note

The *ERROR value, which may perhaps have been set in the MODIFY-LMS-DEFAULTS statement, has the same effect as *NO.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

Read and write authorization for LIBRARY

Protection attributes can only be modified by the owner of the library file.

Examples

– In addition to the read authorization, the group is also given the write authorization for all members to which the group previously only had read access.

```
//modify-element-protection -
//                   *lib(elem=*,type=*,protection=(read=(user=*group))),-
//                     new-protection=(write=(user=*group))
```

When specifying new-protection there is no need to redefine the read authorization. The previous setting (UNCHANGED) for the read authorization still applies.

– Entry of this password is to be made non-visible.

```
//modify-element-protection *lib(elem=test,type=s), -
//           new-protection=(exec=(user=*group,password=*secret))

%ENTER SECRET OPERAND (NEW-PROTECTION=:EXEC=:PASSWORD):
```

MODIFY-LIBRARY-ATTRIBUTES

This statement modifies the attributes of the specified library. These attributes are:

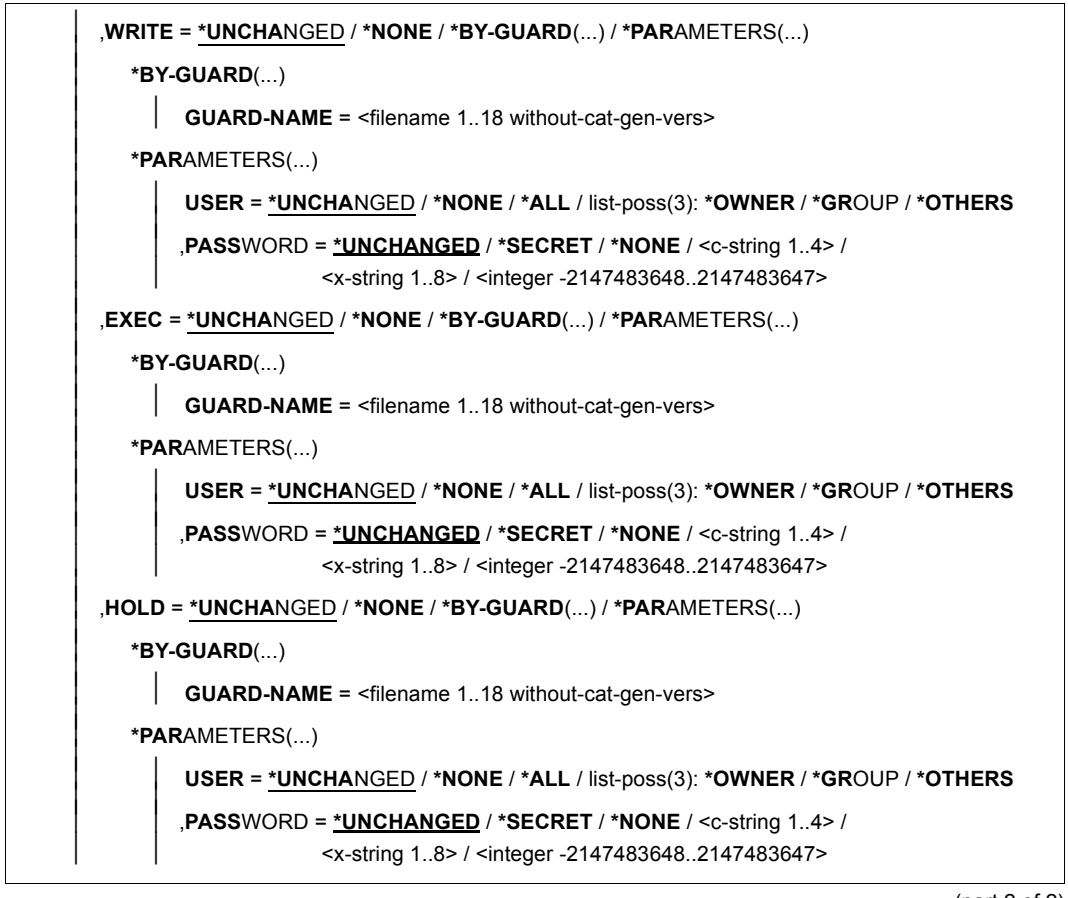
– protection attributes,
– the preferred storage form and
– the recording of the access date

*UNCHANGED means that the respective library attributes remain unchanged.

MODIFY-**LIB**RARY-**ATTR**IBUTES

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)
  ***LINK**(...)
    LINK-NAME = <structured-name 1..8>
,**STOR**AGE-**FORM** = ***UNCHA**NGED / ***NONE** / ***STD** / ***FULL** / ***DELTA**
,**WR**ITE-**CONTROL** = ***UNCHA**NGED / ***NONE** / ***DEACT**IVATE / ***ACTIV**ATE
,**ACCESS-DATE** = ***UNCHA**NGED / ***NONE** / ***KEEP**
,**ADMIN**ISTRATION = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
  ***BY-GUARD**(...)
    GUARD-NAME = <filename 1..18 without-cat-gen-vers>
  ***PAR**AMETERS(...)
    USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
    ,**PASS**WORD **= *UNCHANGED** / ***SECRET** / ***NONE** / <c-string 1..4> /
      <x-string 1..8> / <integer -2147483648..2147483647>
,**INIT-ELEM-PROT**ECTION = ***UNCHA**NGED / ***NONE** / ***PAR**AMETERS(...)
  ***PAR**AMETERS(...)
    READ = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)
      ***BY-GUARD**(...)
        GUARD-NAME = <filename 1..18 without-cat-gen-vers>
      ***PAR**AMETERS(...)
        USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**
        ,**PASS**WORD = ***UNCHANGED** / ***SECRET** / ***NONE** / <c-string 1..4> /
          <x-string 1..8> / <integer -2147483648..2147483647>

(part 1 of 2)



(part 2 of 2)

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library whose protection attributes are to be modified.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose protection attributes are to be modified.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

STORAGE-FORM = *UNCHANGED / *NONE / *STD / *FULL / *DELTA
Permissible storage form for members of the library, provided that nothing different is required for the member type by MODIFY-TYPE-ATTRIBUTES. Also, all members of the same type and name must have the same storage form.

STORAGE-FORM = *NONE
Same effect as STD.

STORAGE-FORM = *STD
Both full storage and delta storage are permitted.

STORAGE-FORM = *FULL
Only full storage is permitted.

STORAGE-FORM = *DELTA
Only delta storage is permitted.

WRITE-CONTROL = *UNCHANGED / *NONE / *DEACTIVATE / *ACTIVATE
Attribute for controlling additional checks.

WRITE-CONTROL = *NONE
No additional checks are performed when generating or overwriting versions.

WRITE-CONTROL = *DEACTIVATE
No additional checks are performed when generating or overwriting versions.

WRITE-CONTROL = *ACTIVATE
A version can be written only

– if the USERID of the user wanting to write it is entered as the HOLDER in the relevant base version and

– if either a new version is generated or the base version is overwritten.

For the first version of a name, no base yet exists; it can be generated only by persons with ADMIN authorization. Whenever versions are generated or overwritten, LMS automatically adds a type-2 record documenting the HOLDER=author and the DATE and TIME of the operation. In addition, the STATE and HOLDER attributes and all rights are applied to the new version, provided the current statement does not require different values.

ACCESS-DATE = *UNCHANGED / *NONE / *KEEP
Attribute for recording the access date.

ACCESS-DATE = *NONE
No access dates for the members are recorded in the library.

ACCESS-DATE = *KEEP
Access dates for members are recorded in the library. Only after this point in time can a member have an access date and time.

ADMINISTRATION = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Administer authorization. The circle of those with administer authorization for the library is specified explicitly. Only these persons may create, delete and rename members.

ADMINISTRATION = *NONE
No administer authorization is granted.

ADMINISTRATION = *BY-GUARD(...)
The administer authorization of the specified library is controlled by means of a guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Name of the guard.

ADMINISTRATION = *PARAMETERS(...)
Specifies the user circle and, where applicable, a password for management of the specified library.

USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
The circle of those with management authorization is explicitly listed.

USER = *NONE
None may manage (empty list).

USER = *ALL
All may manage (full listing).

USER = *OWNER
Only the owner of the library file may manage it.

USER = *GROUP
Those belonging to the group of the owner of the library file may manage.

USER = *OTHERS
All others may manage.

PASSWORD =*UNCHANGED / *SECRET / *NONE / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
The circle of authorized persons is further restricted. In addition to the necessary access right, the correct password is required. Specifying 0 or X’00000000’ does not result in any change of the last value.
Specifying *SECRET or ^ allows the desired password to be entered invisibly. If the “secret” value is entered as a c-string, it must be enclosed in apostrophes. If it is entered as an x-string, it must similarly be enclosed in apostrophes and also be prefixed by an X.

INIT-ELEM-PROTECTION = *UNCHANGED / *NONE / *PARAMETERS(...)

INIT-ELEM-PROTECTION = *NONE
No initial protection for the members contained in the library is defined.

INIT-ELEM-PROTECTION = *PARAMETERS(...)
Specifies the protection rights that the newly created members are to receive.

READ = *UNCHANGED /*NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial read authorization is explicitly specified.

READ = *NONE
No further access restriction is provided.

READ = *BY-GUARD(...)
Specifies the read guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Name of the guard.

READ = *PARAMETERS(...)
Specifies the user circles for the read authorization.

USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
The circle of those with read authorization is explicitly listed.

USER = *NONE
None may access in the specified manner.

USER = *ALL
All may access in the specified manner (full listing).

USER = *OWNER
The owner of the library file may access.

USER = *GROUP
Those belonging to the group of the owner of the library file may access.

USER = *OTHERS
All others may access.

PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
The circle of authorized persons is further restricted. In addition to the necessary access right, the correct password is required.
Specifying 0 or X’00000000’ does not result in any change of the last value.
Specifying *SECRET or ^ allows the desired password to be entered invisibly. If the “secret” value is entered as a c-string, it must be enclosed in apostrophes. If it is entered as an x-string, it must similarly be enclosed in apostrophes and also be prefixed by an X.

WRITE = *UNCHANGED /*NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial write authorization is explicitly defined.
The description of the operands is analogous to that for READ

EXEC = *UNCHANGED /*NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial execute authorization is explicitly defined.
The description of the operands is analogous to that for READ

HOLD = *UNCHANGED /*NONE / *BY-GUARD(...) / *PARAMETERS(...)
Grants the hold authorization for the member.
The operands are analogous to those described for READ.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked

Required access rights

Read and write authorization for LIBRARY

Only the owner of the library file can modify attributes of the library.

MODIFY-LMS-DEFAULTS

The MODIFY-LMS-DEFAULTS statement allows modification of the default values. If an explicit value is used locally in an LMS statement, this value will take priority over the default.

The reference in the LMS statements to the values set here is the specification of *LMS-DEFAULT.

At the beginning of the LMS run the values immediately following *UNCHANGED are valid. If one of these values is changed by the MODIFY-LMS-DEFAULTS statement, this new setting becomes the current setting. This remains valid for the LMS run (*UNCHANGED) until a new MODIFY-LMS-DEFAULTS statement for this value or RESET-LMS-DEFAULTS is issued.

MODIFY-**LMS-DEF**AULTS

ELEMENT-**ATTR**IBUTES = ***UNCHA**NGED / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

TYPE = ***UNCHA**NGED / ***NONE** / <alphanum-name 1..8>

,**ELEM**ENT-**VERS**ION = ***UNCHA**NGED / ***ALL** / ***HIGH**EST-**EXIST**ING

,**TO-ELEM**ENT-**VERS**ION = ***UNCHA**NGED / ***BY-SOURCE** / ***HIGH**EST-**EXIST**ING / ***INCREMENT**

,**STOR**AGE-**FORM** = ***UNCHA**NGED / ***STD** / ***FULL** / ***DELTA**

,**SOUR**CE-**ATTR**IBUTES = ***UNCHA**NGED / ***STD** / ***IGNORE** / ***KEEP**

,**F**ILE-**ATTR**IBUTES = ***UNCHA**NGED / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

ACCESS-METHOD = ***UNCHA**NGED / ***ISAM** / ***SAM**

,**DESTROY-DATA** = ***UNCHA**NGED / ***NO** / ***Y**ES / ***BY-SOUR**CE

,**WR**ITE-**MODE** = ***UNCHA**NGED / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**

,**DIALOG-CONTR**OL = ***UNCHA**NGED / ***NO** / ***Y**ES / ***ERROR**

(part 1 of 5)

,**INF**ORMATION = ***UNCHA**NGED / ***MED**IUM / ***MIN**IMUM / ***MAX**IMUM /
***SUMM**ARY / ***DELTA-STRUCTURE /** ***PAR**AMETERS(...)

***PAR**AMETERS(...)

GENERAL = ***UNCHA**NGED / ***NO** / ***YES**

,HISTORY = ***UNCHA**NGED / ***NO** / ***YES**

,SECURITY = ***UNCHA**NGED / ***NO** / ***YES**

,**LAYOUT** = ***UNCHA**NGED / ***VAR**IABLE / ***FIXED**

,**SORT** = ***UNCHA**NGED / ***BY-NAME** / ***BY-VERSION** / ***BY-USER-DATE** / ***BY-CREATION-DATE** /
***BY-MODI**FICATION-**DATE** / ***BY-ACCESS-DATE** / ***BY-ELEM**ENT-**SIZE** / ***BY-SECONDARY-NAME**

,**OUT**PUT-**FORM** = ***UNCHA**NGED / ***STD** / ***CHAR**ACTER / ***HEX**ADECIMAL / ***DUMP**

,**DEL**ETE-**SOUR**CE = ***UNCHA**NGED / ***NO** / ***Y**ES

,**PROT**ECTION = ***UNCHA**NGED / ***STD** / ***BY-SOUR**CE

,**MAX-ERROR-WEIGHT** = ***UNCHA**NGED / ***SERIOUS** / ***SIGNIFICANT** / ***RECOV**ERABLE

,**EDT-MODE** = ***UNCHA**NGED / ***COMPATIBLE** / ***UNICODE**

,**NEXT-ATTEM**PT = ***UNCHA**NGED / ***NO** / ***Y**ES(...)

***Y**ES(...)

NUMBER-**OF-ATTEM**PTS = ***UNCHA**NGED / <integer 1..2147483647>

,**PER**IOD = ***UNCHA**NGED / <integer 1..21599>

,**COMP**ARE-**PAR**AMETERS = ***UNCHA**NGED / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

RECORD-**PART** = ***UNCHA**NGED / ***ALL** / ***PART**(...)

***PART**(...)

START = ***UNCHA**NGED / <integer 1..32764>

,**LEN**GTH = ***UNCHA**NGED / ***REST** / <integer 1..32764>

,**SPACE**S = ***UNCHA**NGED / ***STD** / ***IGNORED** / ***RELEV**ANT

,**INF**ORMATION = ***UNCHA**NGED / ***MED**IUM / ***MIN**IMUM / ***MAX**IMUM / ***SUMM**ARY / ***STATIS**TICS
/ ***NONE**

,**LAYOUT** = ***UNCHA**NGED / ***COMPATIBLE** / ***COMPR**ESSED

,**JOIN-ELEM**ENT-**SET**S = ***UNCHA**NGED / ***NO** / ***Y**ES

(part 2 of 5)

,**TEXT-INF**ORMATION = ***UNCHA**NGED / ***ALL** / ***STATIS**TICS / ***F**ILE-**ATTR**IBUTES / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

INFORMATION = ***UNCHA**NGED / ***ALL** / list-poss(2): ***TEXT** / ***COMMENT**

,**REC**ORD-**RANGE** = ***UNCHA**NGED / ***ALL** / ***RANGE**(...)

***RANGE**(...)

FROM = ***UNCHA**NGED / <integer 1..2147483647>

,**TO** = ***UNCHA**NGED / ***LAST** / <integer 1..2147483647>

,**REC**ORD-**PART** = ***UNCHA**NGED / ***ALL** / ***PART**(...)

***PART**(...)

START = ***UNCHA**NGED / <integer 1..262144>

,**LEN**GTH = ***UNCHA**NGED / ***REST** / <integer 1..262144>

,**REC**ORD-**NUM**BER = ***UNCHA**NGED / ***BY-OUT**PUT / ***Y**ES / ***NO**

,**MOD**ULE-**INF**ORMATION = ***UNCHA**NGED / ***ALL** / ***STATIS**TICS / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

INFORMATION = ***UNCHA**NGED / ***ALL** / ***TXT**(...) / ***TXTP**(...) /
list-poss(9): ***ESD** / ***ISD** / ***LSD** / ***RLD** / ***REP** / ***INCL**UDE / ***DSDD** / ***REF** / ***END**

***TXT**(...)

CSECT-NAME = ***UNCHA**NGED / ***ALL** / <c-string 1..32 with-low> / <text 1..32>

,**ADDR**ESS = ***UNCHA**NGED (...) / <x-string 1..8>(...)

***UNCHA**NGED(...)

BASE-ADDRESS = ***UNCHA**NGED / <x-string 1..8>

<x-string 1..8>(...)

BASE-ADDRESS = ***UNCHA**NGED / <x-string 1..8>

,**LEN**GTH = ***UNCHA**NGED / ***REST** / <integer 1..2147483647> / <x-string 1..8>

***TXTP**(...)

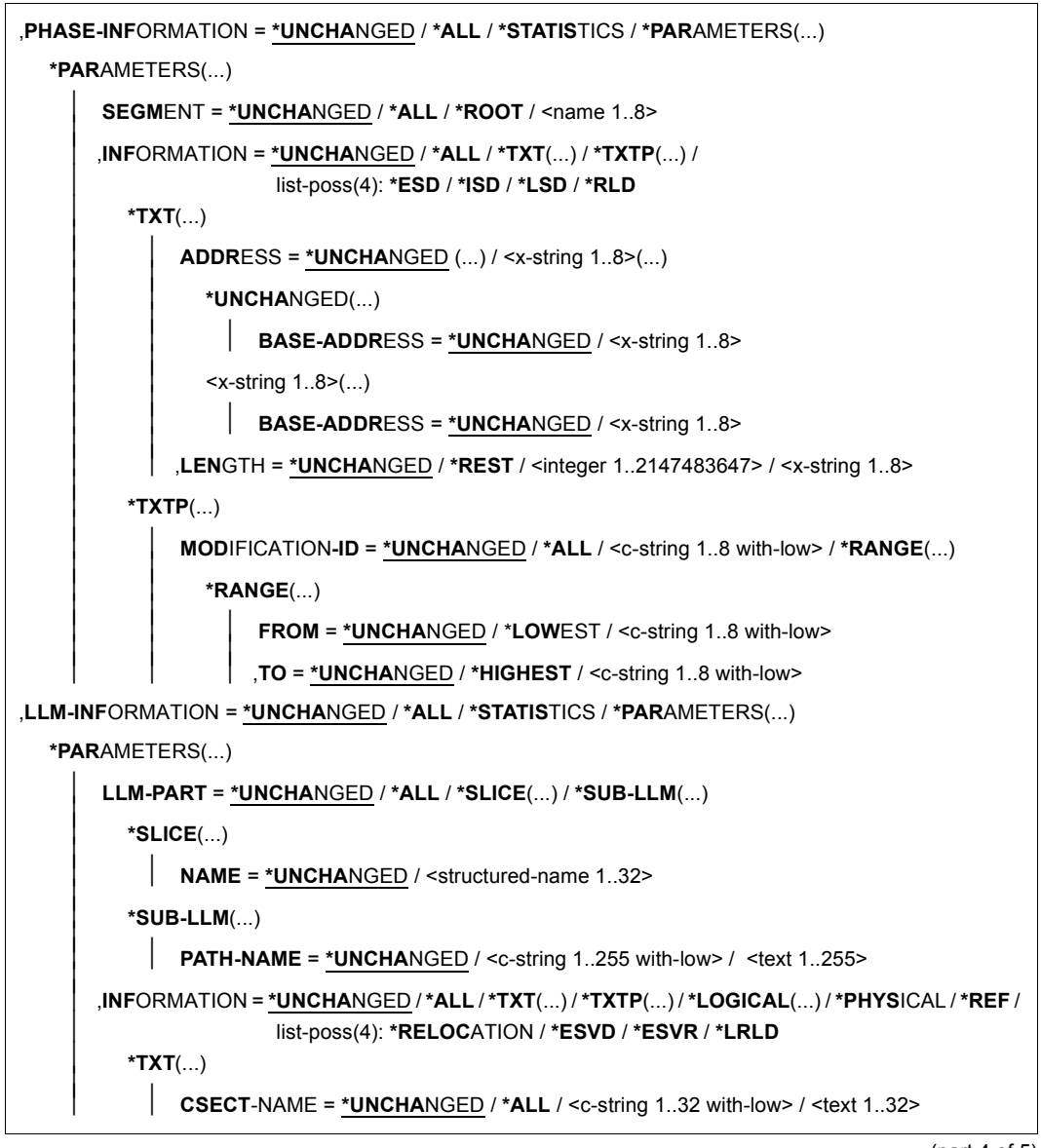
MODIFICATION-**ID** = ***UNCHA**NGED / ***ALL** / <c-string 1..8 with-low> / ***RANGE**(...)

***RANGE**(...)

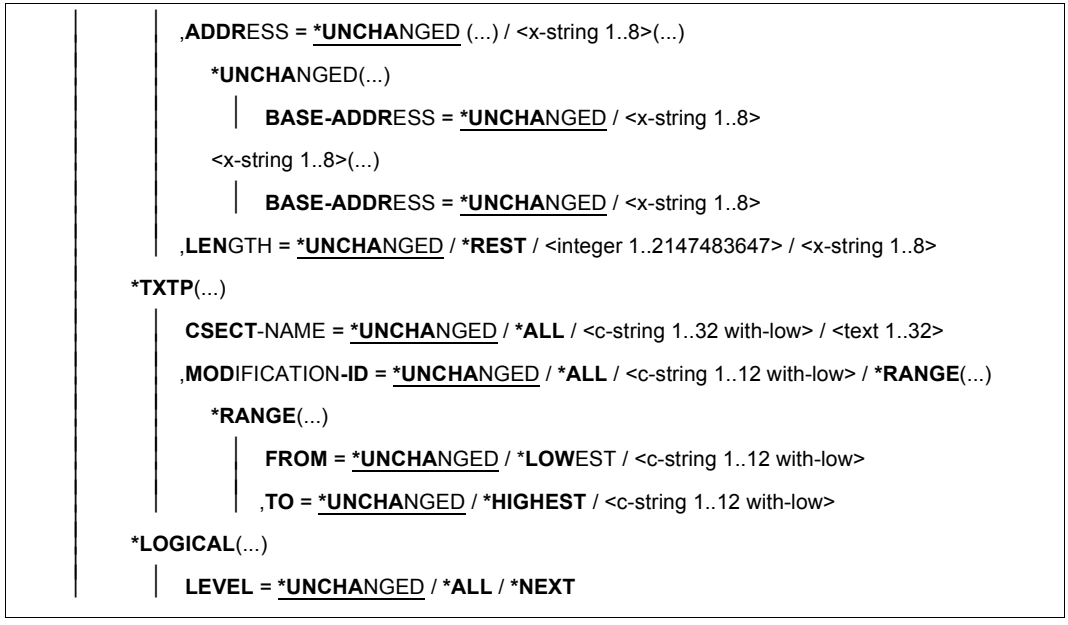
FROM = ***UNCHA**NGED / ***LOW**EST / <c-string 1..8 with-low>

,**TO** = ***UNCHA**NGED / ***HIGHEST** / <c-string 1..8 with-low>

(part 3 of 5)


,PHASE-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
*PARAMETERS(...)
SEGMENT = *UNCHANGED / *ALL / *ROOT / <name 1..8>
,INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) /
list-poss(4): *ESD / *ISD / *LSD / *RLD
*TXT(...)
ADDRESS = *UNCHANGED (...) / <x-string 1..8>(...)
*UNCHANGED(...)
BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
<x-string 1..8>(...)
BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
,LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
*TXTP(...)
MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> / *RANGE(...)
*RANGE(...)
FROM = *UNCHANGED / *LOWEST / <c-string 1..8 with-low>
,TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
,LLM-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
*PARAMETERS(...)
LLM-PART = *UNCHANGED / *ALL / *SLICE(...) / *SUB-LLM(...)
*SLICE(...)
NAME = *UNCHANGED / <structured-name 1..32>
*SUB-LLM(...)
PATH-NAME = *UNCHANGED / <c-string 1..255 with-low> / <text 1..255>
,INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF /
list-poss(4): *RELOCATION / *ESVD / *ESVR / *LRLD
*TXT(...)
CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>

(part 4 of 5)



(part 5 of 5)

ELEMENT-ATTRIBUTES = *UNCHANGED / *PARAMETERS(...)
Defines the member type, member version, storage form and file attributes.

TYPE = *UNCHANGED / *NONE / <alphanum-name 1..8>
Defines the member type.

TYPE = *NONE
No global member type is defined, i.e. the type specifications must be made locally in statements if the library specified in the statement has no library-specific default type.

TYPE = <alphanum-name 1..8>
The name specified here is used as the type in the statements if the library specified in the statement has no library-specific default type. Otherwise, this library-specific default type replaces the local *LMS-DEFAULT in the statement.

ELEMENT-VERSION = *UNCHANGED / *ALL / *HIGHEST-EXISTING
Defines the member version for SHOW-ELEMENT-ATTRIBUTES.

ELEMENT-VERSION = *ALL
All versions of a member are output.

ELEMENT-VERSION = *HIGHEST-EXISTING
Only the highest existing version of a member with reference to BASE is output.

TO-ELEMENT-VERSION = *UNCHANGED / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT
Defines the version of the target member.

TO-ELEMENT-VERSION = *BY-SOURCE
The target member receives the same version as the source member.

TO-ELEMENT-VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

TO-ELEMENT-VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated.

STORAGE-FORM = *UNCHANGED / *STD / *FULL / *DELTA
Storage form for the member being added. The storage form must not contradict the settings made by means of the MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES statements, and all members of a given type and name must have the same storage form.

STORAGE-FORM = *STD
The member is generated in accordance with the storage form required for the member scope. Contradictory requirements result in errors. If nothing is specified, full storage is selected.

STORAGE-FORM = *FULL
The new member is generated as a full member (if this is not possible, an error message is issued).

STORAGE-FORM = *DELTA
The new member is generated as a delta member (if this is not possible, an error message is issued). This entry is permissible for member types: S, P, D, J, M, X and members types derived from them.

SOURCE-ATTRIBUTES = *UNCHANGED / *STD/ *IGNORE / *KEEP
Evaluated only for the ADD-ELEMENT statement.

SOURCE-ATTRIBUTES = *STD
No file attributes and no ISAM key are stored. For ISAM files, a warning is output stating that the ISAM keys were not stored.

SOURCE-ATTRIBUTES = *IGNORE
As for **SOURCE-ATTRIBUTES = *STD**, but no warning is output.

SOURCE-ATTRIBUTES = *KEEP
The following file attributes are stored unchanged in the member being added: ACCESS-METHOD, RECORD-FORMAT, RECORD-SIZE, BUFFER-LENGTH, PERFORMANCE, USAGE, ACCESS and USER-ACCESS.
If ACCESS-METHOD=ISAM, then PADDING-FACTOR, LOGICAL-FLAG-LENGTH, VALUE-FLAG-LENGTH, PROPAGATE-VALUE-FLAG, the ISAM keys and information on ISAM secondary keys are also stored.

FILE-ATTRIBUTES = *UNCHANGED / *PARAMETERS(...)
File attributes that are defined on file creation.

ACCESS-METHOD = *UNCHANGED / *ISAM / *SAM
Defines the file access method.

ACCESS-METHOD = *ISAM
Creates an ISAM file.

ACCESS-METHOD = *SAM
Creates a SAM file.

DESTROY-DATA = *UNCHANGED / *NO / *YES / *BY-SOURCE
Determines whether the data is physically deleted, i.e. overwritten with X'00'.

DESTROY-DATA = *NO
A member of a library is deleted physically only if it contains a flag for physical deletion or the class 2 option DESTLEV requires it.

DESTROY-DATA = *YES
After logical deletion the data, if present, is physically deleted.

DESTROY-DATA = *BY-SOURCE
The code for overwriting the data is taken from the source member or the source file and assigned to the target member or the target file. If no source exists, *BY-SOURCE then has the same effect as DESTROY-DATA=*NO.

WRITE-MODE = *UNCHANGED / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.
If the member to be stored is a delta member, it is necessary to ensure that the member is a leaf of the delta tree. Only leaves of a delta tree may be overwritten.
If the value selected for WRITE-MODE is not possible for a particular statement, the setting WRITE-MODE=*CREATE applies.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise it will be created as a new member.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *UNCHANGED/ *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement. (This operand has no effect in procedures or in batch mode.)

DIALOG-CONTROL = *NO
All members are processed without dialog queries.

DIALOG-CONTROL = *YES
LMS inquires as to how it should act for each member, e.g. whether the member is to be processed, skipped or the statement terminated.

DIALOG-CONTROL = *ERROR
If a recoverable error occurs during member processing, e.g. a member being overwritten, the user is asked how LMS should act.
The user has the following intervention options, where applicable:

YES	The member is to be processed.
NO	The member is not to be processed.
ALL	The statement is to be completed without a dialog.
TERMINATE	The statement is to be terminated.

After pressing the K2 key and entering “/SEND-MSG”, the user has the option of changing the value of the DIALOG-CONTROL operand. If LMS is in the midst of member processing, the user can control further processing as follows with the /SEND-MSG command shown below:

```
/SEND-MSG PROG,´[N-I / N-E / C][,DIALOG-CONTROL= NO / YES / ERROR]´
```

```
SEND-MSG PROG, NEXT-INPUT (N-I)
```
The current statement is aborted; LMS reads in another statement as soon as it is active again. Entering an unknown text or omitting the text has the same effect as NEXT-INPUT.

```
SEND-MSG PROG, NEXT-ELEMENT (N-E)
```
Processing of the current member in the current statement in aborted; LMS continues processing with the next member if there is one. If there is none, NEXT-ELEMENT has the same effect as NEXT-INPUT.

`SEND-MSG PROG, CONTINUE (C)`
LMS processing continues normally.

`C,NO`	The member is to be processed but without activating the dialog.
`N-E/N-I,NO`	Member processing or the statement is to be terminated but without activating the dialog.
`C,YES/ERROR`	The member is to be processed and the dialog subsequently activated.
`N-E/N-I, YES/ERROR`	Member processing or the statement is to be terminated and the dialog subsequently activated.

Note

The value set with /SEND-MSG for DIALOG-CONTROL applies only to the current statement.

INFORMATION = *UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM / *SUMMARY / *DELTA-STRUCTURE / *PARAMETERS(...)
Specifies the scope of the directory to be output.

INFORMATION = *MEDIUM
Outputs the type, name, version, variant number and, depending on the SORT operand, the date or size of the selected member.

INFORMATION = *MINIMUM
Outputs only type, name and version of the selected member.

INFORMATION = *MAXIMUM
Outputs the complete information on the selected member.

INFORMATION = *SUMMARY
Outputs only the number of selected members per type.

INFORMATION = *DELTA-STRUCTURE
Outputs the relation “predecessor - successor” for delta members.
In addition to the member designation, this outputs the internal delta number (DELTA#, which reflects the chronological order) and the corresponding number of the base (BASE#). These internal delta numbers are unique within their tree and describe the concatenation of members in the tree (apart and distinct from the user’s own, external version designation). Tree output is always sorted according to DELTA#, i.e. the operand SORT (see below) has no effect within a tree. Different trees are separated from one another by a line.
In the case of full members, the DELTA# and BASE# output fields are empty.

INFORMATION = *PARAMETERS(...)
Additional outputs

GENERAL = *UNCHANGED / *NO / *YES
Defines whether, in addition to the type, name, version and variant number, a general information block is to be output comprising the storage format, status, member size and, if applicable, the character set and HOLDER of the selected member.

HISTORY = *UNCHANGED / *NO / *YES
Defines whether, in addition to the type, name, version and variant number, a HISTORY block is to be output comprising the user date/time, creation date/time, modification date/time and, if applicable, the access date/time of the selected member.

SECURITY = *UNCHANGED / *NO / *YES
Defines whether, in addition to the type, name, version and variant number, a SECURITY block for the selected member is to be output, if additional member protection has been granted for an access or borrowing right.

LAYOUT = *UNCHANGED / *VARIABLE / *FIXED
Specifies the format of the directory to be output.

LAYOUT = *VARIABLE
The number of print columns is determined by the longest member designation within the member type. With output to the screen and no special sorting, the layout is oriented to the longest member designation in the output buffer, where subsequent outputs within a member type only change if longer member designations occur.

LAYOUT = *FIXED
The directory is printed in a single column in a fixed format. Here, column is meant to signify the list of entries, one below the other, in the directory.

SORT = *UNCHANGED / *BY-NAME / *BY-VERSION / *BY-USER-DATE / *BY-CREATION-DATE / *BY-MODIFICATION-DATE / *BY-ACCESS-DATE / *BY-ELEMENT-SIZE / *BY-SECONDARY-NAME
Sort criterion for the directory entries of the selected members. Type is always used as the first sort criterion.

SORT = *BY-NAME
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, name and version.

SORT = *BY-VERSION
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, version and name.

SORT = *BY-USER-DATE
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, user date, name and version.

SORT = *BY-CREATION-DATE
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, creation date, name and version.

SORT = *BY-MODIFICATION-DATE
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, modification date, name and version.

SORT = *BY-ACCESS-DATE
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, access date, name and version.

SORT = *BY-ELEMENT-SIZE
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, member size, name and version.

SORT = *BY-SECONDARY-NAME
Sorts the directory entries of the selected members according to the following criteria in the order specified: type, secondary name, secondary attribute, name and version.

Note

For more information, see the SHOW-ELEMENT-ATTRIBUTES statement.

OUTPUT-FORM = *UNCHANGED / *STD / *CHARACTER / *HEXADECIMAL / *DUMP
Specifies the format for the output.

OUTPUT-FORM = *STD
The format is selected according to the member type. For text members this operand works in the same way as for OUTPUT-FORM = *CHARACTER.

OUTPUT-FORM = *CHARACTER
The output is in alphanumerical form.

OUTPUT-FORM = *HEXADECIMAL
The output is displayed in mixed alphanumerical/hexadecimal form, with the alphanumerical form above and the hexadecimal form below.

OUTPUT-FORM = *DUMP
The output is displayed in mixed alphanumerical/hexadecimal form, with the two forms displayed side by side
For member types S, P, D, J and M, this operand has an effect like *HEXADECIMAL.

DELETE-SOURCE = *UNCHANGED / *NO / *YES
Specifies whether the source file is to be kept (default *NO) or deleted (parameter *YES). This operand has no effect if the data is read from *OMF.

PROTECTION = *UNCHANGED / *STD / *BY-SOURCE
Member protection for a member or file protection for a file.

PROTECTION = *STD
In the case of member protection:
If the member already exists, its member protection remains unchanged. If the member does not yet exist and initial member protection has been defined for the relevant library or the member type, the member is given the protection specified here.
In the case of file protection:
The member protection in effect is not taken into account in the file protection given to the file being generated.

PROTECTION = *BY-SOURCE
Specifies that protection equivalent to the source's is to be applied. There are three possible cases:
A member being copied is given the same protection as the source member.
A member being added is given member protection corresponding to the file protection attributes of the source file's access protection mechanism.
A file being generated is assigned an access protection mechanism corresponding to the member protection in effect for the source member.

MAX-ERROR-WEIGHT = *UNCHANGED / *SERIOUS / *SIGNIFICANT / *RECOVERABLE
Specifies the errors which are to cause LMS to trigger the spin-off mechanism.

MAX-ERROR-WEIGHT = *SERIOUS
The spin-off mechanism is to be triggered in the event of serious errors, i.e. errors which make it pointless to continue processing the statement.

MAX-ERROR-WEIGHT = *SIGNIFICANT
The spin-off mechanism is to be triggered as with *SERIOUS, but also in the event of other errors (except when a member cannot be found or overwritten).

MAX-ERROR-WEIGHT = *RECOVERABLE
The spin-off mechanism is to be triggered in the event of any error.

EDT-MODE = *UNCHANGED / *COMPATIBLE / *UNICODE
Specifies the mode that EDT is to be called in.

EDT-MODE = *COMPATIBLE
EDT is called in compatibility mode.

EDT-MODE = *UNICODE
EDT is called in Unicode mode.

NEXT-ATTEMPT = *UNCHANGED / *NO / *YES(...)
Controls further attempts to open a source when a file, type, or member lock is encountered in a procedure or in batch mode.

NEXT-ATTEMPT = *NO
Specifies that no further attempts are to be made to open the source.

NEXT-ATTEMPT = *YES(...)
Specifies that further attempts are to be made to open the source.

NUMBER-OF-ATTEMPTS = *UNCHANGED / <integer 1..2147483647>
Number of further attempts to be made (default is 9).

PERIOD = *UNCHANGED / <integer 1..21599>
Wait time, in seconds, between attempts (default is 6).

COMPARE-PARAMETERS = *UNCHANGED / *PARAMETERS(...)
Specifies the comparison parameters, the type of comparison (formal or logical) and the scope and format of logging.

RECORD-PART = *UNCHANGED / *ALL / *PART(...)
Defines the record area to be compared.

RECORD-PART = *ALL
The entire record is compared.

RECORD-PART = *PART(...)
Area specification for the part of the record to be compared.

START = *UNCHANGED / <integer 1..32764>
Starting point of the area containing the part of the record to be compared. If no value is entered, the record is compared starting at the beginning.

LENGTH = *UNCHANGED / *REST / <integer 1..32764>
Length of the area in the record to be compared. If no value is entered, the record is compared starting at the beginning.

SPACES = *UNCHANGED / *STD / *IGNORED / *RELEVANT
Handling of space characters in the record.

SPACES = *STD
As *IGNORED for text members, otherwise as *RELEVANT.

SPACES = *IGNORED
Logical comparison. The comparison fields are compared one character at a time; spaces are ignored.

SPACES = *RELEVANT
Formal comparison. The comparison fields are first checked for matching length. If the lengths match, the fields are compared in their entirety. If the lengths differ, the records are logged as being non-matching.

INFORMATION = *UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM / *SUMMARY / *STATISTICS/ *NONE
Scope of logging.

INFORMATION = *MEDIUM
Standard comparison log. The comparison range of non-matching records is logged in its entirety. With matching records, only range specifications (record numbers) are logged. The comparison statistics are output.

INFORMATION = *MINIMUM
Minimum comparison log.
For matching and non-matching records, only range specifications (record numbers) are logged. The comparison statistics are output.

INFORMATION = *MAXIMUM
Detailed comparison log.
All records are logged.
The comparison statistics are output.

INFORMATION = *SUMMARY
No comparison log. Only the comparison statistics are output.

INFORMATION = *STATISTICS
No comparison log. The comparison statistics are output in compressed format.

INFORMATION = *NONE
No logging (no comparison log, no comparison statistics).
*NONE is expedient only when the SHOW-STATISTICS statement is used.

LAYOUT = *UNCHANGED / *COMPATIBLE / *COMPRESSED
Logging format.

LAYOUT = *COMPATIBLE
The comparison log is output in standard format. This format is compatible with earlier LMS versions.

LAYOUT = *COMPRESSED
The comparison log is output in a compressed format.

JOIN-ELEMENT-SETS = *UNCHANGED / *NO / *YES
Defines the member set to be compared.

JOIN-ELEMENT-SETS = *NO
Only the primary members and the secondary members determined through construction are used for the comparison.

JOIN-ELEMENT-SETS = *YES
All primary and secondary members are used for the comparison.

TEXT-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *FILE-ATTRIBUTES / *PARAMETERS(...)
Specifies the scope of information to be output for all members except members of types R, C and L. For PAM members, all entries except *FILE-ATTRIBUTES have an effect like *ALL.

TEXT-INFORMATION = *ALL
Outputs everything.

TEXT-INFORMATION = *STATISTICS
Outputs the number of records per record type and their total.

TEXT-INFORMATION = *FILE-ATTRIBUTES
Outputs only the stored file attributes.

TEXT-INFORMATION = *PARAMETERS(...)
Specifies the section of the member which is to be output.

INFORMATION = *UNCHANGED / *ALL / list-poss(2): *TEXT / *COMMENT
The member section to be output

INFORMATION = *ALL
Everything is output.

INFORMATION = *TEXT
Outputs the text itself, i.e. record type 1.

INFORMATION = *COMMENT
Outputs the separately stored comment, i.e. record type 2.

RECORD-RANGE = *UNCHANGED / *ALL / *RANGE(...)
The section of the member to be processed.

RECORD-RANGE = *ALL
All records are processed.

RECORD-RANGE = *RANGE(...)
Specifies the range of record numbers which is to be processed. The record numbers refer not to a record type, but to the section of the member designated by means of INFORMATION=. Within this section, the records are numbered consecutively from 1 through n.

FROM = *UNCHANGED / <integer 1..2147483647>
Specifies the first record number to indicate the beginning of the section. If nothing is specified, record number 1 is assumed.

TO = *UNCHANGED / *LAST / <integer 1..2147483647>
Specifies the last record number to indicate the end of the section. If nothing is specified, the last record number is used.

RECORD-PART = *UNCHANGED / *ALL / *PART(...)
Specifies the part of the record to be processed.

RECORD-PART = *ALL
Processes the entire record.

RECORD-PART = *PART(...)
Specifies the part of the record to be processed. If the default values are not changed, the entire record is processed.

START = *UNCHANGED / <integer 1..262144>
Specifies the first character in the record to indicate the beginning of the section. If nothing is specified, the first character is assumed to be the beginning.

LENGTH = *UNCHANGED / *REST / <integer 1..262144>
Specifies the length of the section. If nothing is specified, the rest of the record is taken as the length. LENGTH = 0 means that only the record headers are output

RECORD-NUMBER = *UNCHANGED / *BY-OUTPUT / *YES / *NO
Controls the output of the record numbers.

RECORD-NUMBER = *BY-OUTPUT
Record numbers will be output in all cases except when the output is directed to SYSOUT.

RECORD-NUMBER = *YES
Record numbers will also be output via SYSOUT.

RECORD-NUMBER = *NO
No record numbers are to be output.

MODULE-INFORMATION = *UNCHANGED /*ALL / *STATISTICS / *PARAMETERS(...)
Specifies the scope of information to be output for object modules (type-R members).

MODULE-INFORMATION = *ALL
Outputs everything.

MODULE-INFORMATION = *STATISTICS
Outputs the names, lengths and addresses of the CSECTS, as well as the overall length of the module.

MODULE-INFORMATION = *PARAMETERS(...)
Specifies whether all or only selected record types are to be output.

INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / list-poss(9): *ESD / *ISD / *LSD / *RLD / *REP / *INCLUDE / *DSDD / *REF / *END
The record types listed here may be selected.

INFORMATION = *TXT(...)
Selects text records.

CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
The text records can be limited to those from a CSECT.

ADDRESS = *UNCHANGED(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
Outputs TXTP records.

MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> / *RANGE(...)
Selects TXTP records using the specified identification.

MODIFICATION-ID = *RANGE(...)
Selects multiple TXTP records in a range.

FROM = *UNCHANGED / *LOWEST / <c-string 1..8 with-low>
Specifies the beginning of the range. If nothing is specified, the lowest identification for the TXTP records is used.

TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
Specifies the end of the range. If nothing is specified, the highest identification for the TXTP records is used.

PHASE-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
Specifies the scope of information to be output for phases (type-C members).

PHASE-INFORMATION = *ALL
Outputs everything.

PHASE-INFORMATION = *STATISTICS
Outputs the name, length and address of the segment, as well as its overall length.

PHASE-INFORMATION = *PARAMETERS(...)
Specifies whether all or only selected record types are to be output.

SEGMENT = *UNCHANGED / *ALL / *ROOT / <name 1..8>
The phase segment which is selected.

INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / list-poss(4): *ESD / *ISD / *LSD / *RLD
The record types listed here may be selected.

INFORMATION = *TXT(...)
Selects text records

ADDRESS = *UNCHANGED(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
Outputs TXTP records.

MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> / *RANGE(...)
Selects TXTP records using the specified identification.

MODIFICATION-ID = *RANGE(...)
Selects multiple TXTP records in a range.

FROM = *UNCHANGED / *LOWEST / <c-string 1..8 with-low>
Specifies the beginning of the range. If nothing is specified, the lowest identification for the TXTP records is used.

TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
Specifies the end of the range. If nothing is specified, the highest identification for the TXTP records is used.

LLM-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
Specifies the scope of information to be output for link and load modules (type-L members).

LLM-INFORMATION = *ALL
Outputs everything.

LLM-INFORMATION = *STATISTICS
Outputs general information about the link and load module (name, copyright, etc.).

LLM-INFORMATION = *PARAMETERS(...)
Specifies whether all or only selected record types are to be output.

LLM-PART = *UNCHANGED / *ALL / *SLICE(...) / *SUB-LLM(...)
Specifies the part of the LLM which is to be selected.

LLM-PART = *SLICE(...)
Specifies the slice to be output.

NAME = *UNCHANGED / <structured-name 1..32>
Name of the slice to be output.

LLM-PART = *SUB-LLM(...)
Specifies the SUB-LLM to be output.

PATH-NAME = *UNCHANGED / <c-string 1..255 with-low> / <text 1..255>
Specifies the SUB-LLM to be output by means of its path name.

INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF / list-poss(4): *RELOCATION / ***ESVD / *ESVR / *LRLD**
The record types listed here may be selected.

INFORMATION = *TXT(...)
Selects text records.

CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
The text records can be limited to those from a CSECT.

ADDRESS = *UNCHANGED(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
Outputs TXTP records.

CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
The TXTP records can be limited to those from a CSECT.

MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..12 with-low> / *RANGE(...)
Selects TXTP records using the specified identification.

MODIFICATION-ID = *RANGE(...)
Selects multiple TXTP records in a range.

FROM = *UNCHANGED / *LOWEST / <c-string 1..12 with-low>
Specifies the beginning of the range. If nothing is specified, the lowest identification for the TXTP records is used.

TO = *UNCHANGED / *HIGHEST / <c-string 1..12 with-low>
Specifies the end of the range. If nothing is specified, the highest identification for the TXTP records is used.

INFORMATION = *LOGICAL(...)
Outputs the logical structure of the LLM.

LEVEL = *UNCHANGED / *ALL / *NEXT
Specifies whether all or only the next substructure is to be output. The default is all.

INFORMATION = *PHYSICAL
Outputs the physical structure of the LLM.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Example

The file TEST1 is to be added as a type D member to library LIB3.
The default setting for the member type is changed to the desired value by means of the MODIFY-LMS-DEFAULTS statement. In this way it is no longer necessary to specify the member type for each of the subsequent ADD-ELEMENT statements. In order that LMS may report successful addition of the files, the MODIFY-LOGGING-PARAMETERS statement is used to set the scope of logging to the complete LMS log.

```
/START-LMS
//OPEN-LIBRARY LIB3,*UPDATE
//MODIFY-LMS-DEFAULTS TYPE=D
//MODIFY-LOGGING-PARAMETERS LOGGING=*MAXIMUM
//ADD-ELEMENT TEST1
INPUT  FILE
OUTPUT LIBRARY= :1OSQ:$USER.LIB3
         ADD :1OSQ:$USER.TEST1 AS (D)TEST1/@(0001)/2012-11-12
//END
```

The library LIBCCSN contains an element LONGR in EDF03IRV code with records longer than 256 bytes. In order to edit these long records with EDT, EDT must be called in Unicode-mode.
Therefore the default setting for the EDT operating mode (COMPATIBLE) is changed to the desired Unicode mode by means of the MODIFY-LMS-DEFAULTS statement.
Now the EDIT-ELEMENT statement calls EDT in Unicode mode, which allows to edit long records, and reads the LONGR element into work file 0.

```
//START-LMS
//MOD-LMS-DEF EDT-MODE=*UNICODE
//OPEN-LIB LIBCCSN, *UPD
//EDIT-ELEM (,LONGR,S)
//END
```

MODIFY-LOGGING-PARAMETERS

The MODIFY-LOGGING-PARAMETERS statement modifies the global settings for the editor, logging scope, output medium and logging format.

If one of these values is changed by the MODIFY-LOGGING-PARAMETERS statement, this new setting becomes the current setting. This remains valid for the LMS run (*UNCHANGED) until a new MODIFY-LOGGING-PARAMETERS statement for this value or RESET-LOGGING-PARAMETERS is issued.

At the beginning of the LMS run, the values immediately following *UNCHANGED are valid. The default or current values can be output with the appropriate SHOW statement..

MODIFY-**LOG**GING-**PAR**AMETERS

LOGGING = ***UNCHA**NGED / ***MIN**IMUM / ***MAX**IMUM

,**TEXT-OUTPUT** = **UNCHA**NGED / ***SYSOUT** / ***SYSLST**(...) / ***NONE** / ***EDT**(...) / ***LIB**RARY-**ELEM**ENT(...)

*SYSLST(...)

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

*EDT(...)

WRITE-**MODE** = ***UNCHA**NGED / ***EXTEND** / ***REPLACE**

*LIBRARY-ELEMENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

*LINK(...)

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

,**ELEM**ENT = <composed-name 1..64 with-under>(...)

<composed-name 1..64 with-under>(...)

VERSION = ***UP**PER-**LIM**IT / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

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

,**WR**ITE-**MODE** = ***UNCHA**NGED / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**

(part 1 of 2)

```
,OUTPUT-LAYOUT = *UNCHANGED / *PARAMETERS(...)
  *PARAMETERS(...)
    LINES-PER-PAGE = *UNCHANGED / <integer 1..9999>
    ,LINE-SIZE = *UNCHANGED / 132 / 80
    ,EXTRA-FORM-FEED = *UNCHANGED / *NO / *YES
    ,HEADER-LINES = *UNCHANGED / *YES / *NO
```

(part 2 of 2)

LOGGING = *UNCHANGED / *MINIMUM / *MAXIMUM
Defines the scope of LMS logging.

LOGGING = *MINIMUM
Only error messages and negative acknowledgments are output.

LOGGING = *MAXIMUM
A complete LMS log is output.

TEXT-OUTPUT = *UNCHANGED / *SYSOUT / *SYSLST(...) / *NONE / *EDT(...) / *LIBRARY-ELEMENT(...)
This parameter defines the output medium. If the medium is changed or if WRITE-MODE=*EXTEND is entered, page numbering always begins with 1.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Denotes the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *NONE
Except for error messages, output is suppressed.

TEXT-OUTPUT = *EDT(...)
The output is written to EDT work file 9.
If an error occurs during log output, LMS switches over to the default log stream (SYSOUT).

WRITE-MODE = *UNCHANGED / *EXTEND / *REPLACE
Write mode for the output with regard to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output is added to it. Otherwise, the output is written at the beginning of the work file.

WRITE-MODE =*REPLACE
Writes the output at the beginning of work file 9. Any data already present in the work file is overwritten.

TEXT-OUTPUT = *LIBRARY-ELEMENT(...)
The output is stored in a library member.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library in which the output is to be stored. Either the library set globally by means of OPEN-LIBRARY is used as standard, or the explicitly specified library or the library assigned via the link name is used.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = <composed-name 1..64 with-under>(...)
Specifies the member in which the output is to be stored.

VERSION = *UPPER-LIMIT / *HIGHEST-EXISTING / *INCREMENT / <composed-name 1..24 with-under>
Specifies the version that the member is to receive.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated (see also page 55).

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. For further information concerning specification of the base, see page 50.

TYPE = P / <alphanum-name 1..8>
Specifies the member type. By default the member in which the output is stored receives type P for print-edited files.

WRITE-MODE = *UNCHANGED / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise it will be created as a new member.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

OUTPUT-LAYOUT = *UNCHANGED / *PARAMETERS(...)
This parameter defines the LMS log format.

LINES-PER-PAGES = *UNCHANGED / <integer 1..9999>
This parameter defines the page length.
Default value: 64 lines

LINE-SIZE = *UNCHANGED / 132 / 80
This parameter defines the line length.

LINE-SIZE = 132
The line is to be 132 characters long.

LINE-SIZE = 80
The line is to be 80 characters long.

EXTRA-FORM-FEED = *UNCHANGED / *NO / *YES
This parameter controls an extra form feed.

EXTRA-FORM-FEED = *NO
A form feed will only occur when the page is full.

EXTRA-FORM-FEED = *YES
A form feed will occur either when the page is full or when a change of statement or member takes place.

HEADER-LINES = *UNCHANGED / *YES / *NO
This parameter controls the output of headers.

HEADER-LINES = *YES
Headers containing the library and member designations are output.

HEADER-LINES = *NO
No headers are output.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0151	Input or output medium set to standard
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0093	Protocol member already exists
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0510	Base not found
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

No access rights are necessary.

MODIFY-TYPE-ATTRIBUTES

This statement can be used to set or modify certain attributes for the specified type:

– the supertype
– the applicable version convention
– the storage form of the members
– additional checks performed when members are generated or overwritten
– the group of persons authorized to generate, delete and rename members
– the initial member protection

At the beginning of the LMS run, the values immediately following *UNCHANGED apply. The default or current values can be output with the appropriate SHOW statement. .

MODIFY-**TYPE-ATTR**IBUTES

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**TYPE** = ***LMS-DEF**AULT / <alphanum-name 1..8>

,**SUPER-TYPE** = ***UNCHA**NGED / ***NONE** / <alphanum-name 1..8>

,**CONVENT**ION = ***UNCHA**NGED / ***NONE** / ***STD-TREE** / ***STD-SEQU**ENCE(...) / ***MULTI-SEQU**ENCE(...)

***STD-SEQU**ENCE(...)

EXAMPLE = **001** / <composed-name 1..24 with-under>

***MULTI-SEQU**ENCE(...)

EXAMPLE = <composed-name 1..24 with-under>

,**STOR**AGE-**FORM** = ***UNCHA**NGED / ***NONE** / ***STD** / ***FULL** / ***DELTA**

,**WR**ITE-**CONTROL** = ***UNCHA**NGED / ***NONE** / ***DEACT**IVATE / ***ACTIV**ATE

,**ADMIN**ISTRATION = ***UNCHA**NGED / ***NONE** / ***BY-GUARD**(...) / ***PAR**AMETERS(...)

***BY-GUARD**(...)

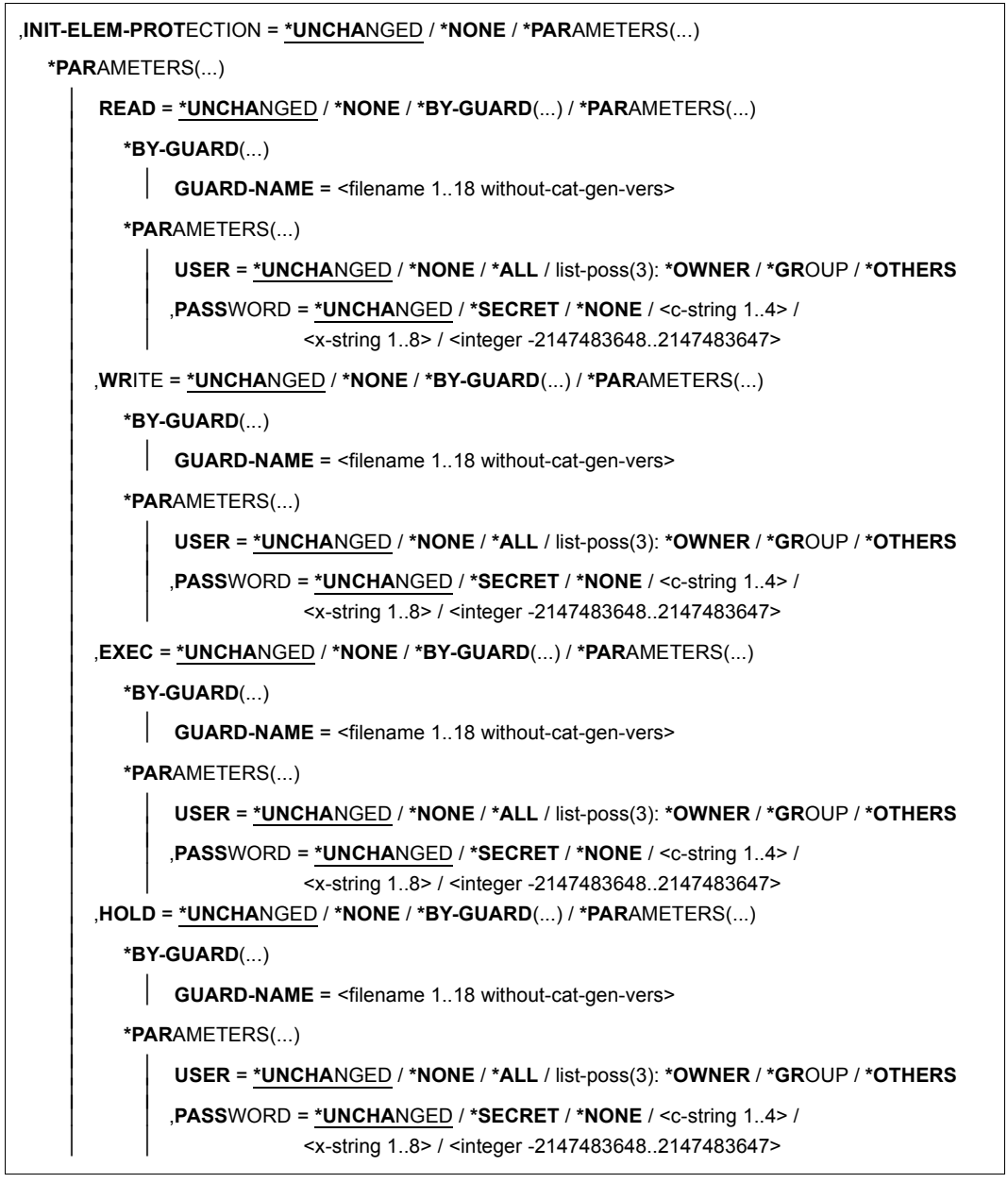
GUARD-NAME = <filename 1..18 without-cat-gen-vers>

***PAR**AMETERS(...)

USER = ***UNCHA**NGED / ***NONE** / ***ALL** / list-poss(3): ***OWNER** / ***GR**OUP / ***OTHERS**

,**PASS**WORD = ***UNCHA**NGED / ***SECRET** / ***NONE** / <c-string 1..4> /
<x-string 1..8> / <integer -2147483648..2147483647>

(part 1 of 2)


,INIT-ELEM-PROTECTION = *UNCHANGED / *NONE / *PARAMETERS(...)
*PARAMETERS(...)
READ = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
*PARAMETERS(...)
USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> /
<x-string 1..8> / <integer -2147483648..2147483647>
,WRITE = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
*PARAMETERS(...)
USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> /
<x-string 1..8> / <integer -2147483648..2147483647>
,EXEC = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
*PARAMETERS(...)
USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> /
<x-string 1..8> / <integer -2147483648..2147483647>
,HOLD = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..18 without-cat-gen-vers>
*PARAMETERS(...)
USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> /
<x-string 1..8> / <integer -2147483648..2147483647>

(part 2 of 2)

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library in which the type attributes are to be defined.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library in which the type attributes are to be defined.

LIBRARY = *LINK(..)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

TYPE = *LMS-DEFAULT / <alphanum-name 1..8>
Member type whose attributes are to be set or modified.

SUPER-TYPE = *UNCHANGED / *NONE / <alphanum-name 1..8>
Name of the superordinate type.

SUPER-TYPE = *NONE
No superordinate type exists.

SUPER-TYPE = <alphanum-name 1..8>
Name of the superordinate type. The resulting graph must contain no cycles (tree).
No SUPER-TYPE can be defined for standard types (those with one-character designations) and types with designations beginning with $ or SYS. The values R, C and L are not allowed as SUPER-TYPEs.

CONVENTION = *UNCHANGED / *NONE / *STD-TREE / *STD-SEQUENCE(...) / *MULTI-SEQUENCE(...)
Version convention which is henceforth to apply to the specified type.

CONVENTION = *NONE
The version convention is removed from the specified type. This can be done at any time.

CONVENTION = *STD-TREE
The version convention *STD-TREE applies to the type concerned. It is not permissible to specify *STD-TREE if a member exists under the specified type.

CONVENTION = *STD-SEQUENCE(...)
The version convention *STD-SEQUENCE applies to the type concerned. All members of the type have the same version format, as specified by means of an example.
It is not permissible to specify *STD-SEQUENCE if a member exists under the specified type.

EXAMPLE = 001 / <composed-name 1..24 with-under>
Example for the version format. The format comprises a leading part, possibly empty, and the maximum-length concluding digit group, which must not be empty. For all versions under the type concerned, both these parts must have the same length as the example. The concluding digit group is used for automatic version incrementation.
If no explicit specification is made here, the value “001” is used as the example for the version format.

CONVENTION = *MULTI-SEQUENCE(...)
The *MULTI-SEQUENCE version convention only applies to the affected type. All members of the type have the same version format. This is specified using an example. It is not permissible to specify *MULTI-SEQUENCE if a member exists under the specified type.

EXAMPLE = <composed-name 1..24 with-under>
Example for the version format. The format comprises a leading part, possibly empty, and the maximum-length concluding digit group, which must not be empty. For all versions under the type concerned, both these parts must have the same length as the example. The concluding digit group is used for automatic version incrementation.

STORAGE-FORM = *UNCHANGED / *NONE / *STD / *FULL / *DELTA
Permissible storage form for members of this type. Also, all members of the same type and name must have the same storage form.

STORAGE-FORM = *NONE
Both full storage and delta storage are permitted.

STORAGE-FORM = *STD
Both full storage and delta storage are permitted.

STORAGE-FORM = *FULL
Only full storage is permitted.

STORAGE-FORM = *DELTA
Only delta storage is permitted.

WRITE-CONTROL = *UNCHANGED / *NONE / *DEACTIVATE / *ACTIVATE
Attribute for controlling additional checks.

WRITE-CONTROL = *NONE
The WRITE-CONTROL setting for the library applies.

WRITE-CONTROL = *DEACTIVATE
No additional checks are performed when generating or overwriting versions.

WRITE-CONTROL = *ACTIVATE
A version can be written only

– if the USERID of the user wanting to write it is entered as the HOLDER in the relevant base version and

– if either a new version is generated or the base version is overwritten.

For the first version of a name, no base yet exists; it can be generated only by persons with ADMIN authorization. Whenever versions are generated or overwritten, LMS automatically adds a type-2 record documenting the HOLDER=author and the DATE and TIME of the operation. In addition, the STATE and HOLDER attributes and all rights are applied to the new version, provided the current statement does not require different values.

ADMINISTRATION = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Administer authorization. The circle of those with administer authorization for this type is specified explicitly. Only these persons may create, delete and rename members.

ADMINISTRATION = *NONE
No administer authorization is granted for this type. The setting for the library applies.

ADMINISTRATION = *BY-GUARD(...)
The administer authorization for this type is controlled by means of a guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Name of the guard.

ADMINISTRATION = *PARAMETERS(...)
Specifies the group of authorized users and perhaps a password for the administration of the specified type.

USER = *UNCHANGED / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
The circle of those with administer authorization is explicitly listed.

USER = *NONE
No one has administer authorization.

USER = *ALL
All users have administer authorization (full listing).

USER = *OWNER
Only the owner of the library file has administer authorization.

USER = *GROUP
Those belonging to the group of the owner of the library file have administer authorization.

USER = *OTHERS
All others have administer authorization.

PASSWORD = *UNCHANGED / *SECRET / *NONE / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
The circle of authorized persons is further restricted. In addition to the necessary access right, the correct password is required. Specifying 0 or X'00000000' does not cause the last value to be changed.
Specifying *SECRET or ^ allows the desired password to be entered invisibly. If the "secret" value is entered as a c-string, it must be enclosed in apostrophes. If it is entered as an x-string, it must similarly be enclosed in apostrophes and also be prefixed by an X.

INIT-ELEM-PROTECTION = *UNCHANGED / *NONE/ *PARAMETERS(...)
Specifies the initial member protection that is to be entered for the members newly created under the above type.

INIT-ELEM-PROTECTION = *NONE
No initial protection is defined for this member type.

INIT-ELEM-PROTECTION = *PARAMETERS(...)
Specifies the protection rights that the type specified above is to receive.

READ = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial read authorization is explicitly specified.

READ = *NONE
No further access restriction is provided.

READ = *BY-GUARD(...)
Specifies the read guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Name of the guard.

READ = *PARAMETERS(...)
Specifies the user circles for the read authorization.

USER = *UNCHANGED / *NONE / *ALL/ list-poss(3): *OWNER / *GROUP / *OTHERS
The circle of those with read authorization is explicitly listed.

USER = *NONE
None may access in the specified manner.

USER = *ALL
All may access in the specified manner (full listing).

USER = *OWNER
The owner of the library file may access.

USER = *GROUP
Those belonging to the group of the owner of the library file may access.

USER = *OTHERS
All others may access.

PASSWORD = *UNCHANGED /*SECRET / *NONE / <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
The circle of authorized persons is further restricted. In addition to the necessary access right, the correct password is required.
Specifying *SECRET or ^ allows the desired password to be entered invisibly. If the "secret" value is entered as a c-string, it must be enclosed in apostrophes. If it is entered as an x-string, it must similarly be enclosed in apostrophes and also be prefixed by an X.

WRITE = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial write authorization is explicitly defined.
The operands are analogous to those described for READ, see page 361.

EXEC = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial execute authorization is explicitly defined.
The operands are analogous to those described for READ.

HOLD = *UNCHANGED / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
An initial hold authorization is explicitly defined.
The operands are analogous to those described for READ.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0413	Type locked

Required access rights

Read and write authorization for LIBRARY

Only the owner of the library file can modify attributes of the type.

Note

The type attributes are retained even if all the members of a given type are deleted.

Example

Three changes are to be made regarding library X.1:

– Only specified types are to be permitted.

– The STD-SEQUENCE convention with version example V001 is to be set for all S-type members.

– The SUPER-TYPE ´S´ is to be assigned to the user-defined type USER1.

```
//MODIFY-LIBRARY-ATTRIBUTES LIBRARY=X.1,-                "library X"
//                        ADMIN=*PAR(USER=*NONE)         "type-specific setting"
//MODIFY-TYPE-ATTRIBUTES TYPE=S,-                        "standard type S"
//                        SUPER-TYPE=*NONE,-             "no supertype"
//                        CONVENTION=*STD-SEQUENCE -     "version convention"
//                                   (EXAMPLE=V001),-    "default version"
//                        ADMIN=*PAR(USER=*ALL)          "everyone may administer"
//MODIFY-TYPE-ATTRIBUTS TYPE=USER1,-                     "for user-defined type"
//                        SUPER-TYPE=S,-                 "suitable supertype"
//                        ADMIN=*PAR(USER=*UNCHANGED)    "selective setting"
//SHOW-TYPE-ATTRIBUTES TYPE=S
INPUT  LIBRARY= :1OSQ:$USER.X.1
TYPE       = S
SUPER-TYPE = *NONE
BASE-TYPE  = S
CONVENTION = *STD-SEQUENCE
EXAMPLE    = V001
INIT-ELEM-P= *NONE
ADMIN-PASS = *NONE         ADMIN-USER = *OWNER *GROUP *OTHERS
STORAGE    = *NONE         WR-CONTROL = *NONE
```

For another example dealing with member protection and showing how to set type attributes, see page 487.

OPEN-LIBRARY

OPEN-LIBRARY is used to define and open a global library. This is referenced in other statements by means of LIBRARY = *STD.

If two libraries are required in a statement, then the second library must be specified explicitly in the statement or via a link name.

Global libraries remain open until they are explicitly closed by means of the CLOSE-LIBRARY statement or until a new OPEN-LIBRARY statement is issued.

Global libraries are opened for reading as standard. If they are to be opened for reading and writing, the operand MODE=*UPDATE must be set.

If a new library is set up, it must be generated with MODE=*UPDATE.

```
OPEN-LIBRARY

 LIBRARY = <filename 1..54 without-vers> / *LINK(...)
    *LINK(...)
      | LINK-NAME = <structured-name 1..8>
,MODE = *READ / *UPDATE(...)
    *UPDATE(...)
      | STATE = *ANY / *OLD / *NEW
,DEFAULTS = *UNCHANGED /  *PARAMETERS (...)
    *PARAMETERS(...)
      | TYPE = *UNCHANGED / *NONE /   <alphanum-name 1..8>
,SNAPSET = *NONE / *LATEST / <name 1..1 with-low> / <integer -52..-1>
```

LIBRARY = <filename 1..54 without-vers>
The library with the name specified here is set up as a global library and opened.

LIBRARY = *LINK(...)
The library assigned via the link name is set up as a global library and opened.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

MODE = *READ / *UPDATE(...)
Library open mode.

MODE = *READ
The library is opened only for reading. It must already exist.

MODE = *UPDATE(...)
The library is opened for reading and writing.

STATE = *ANY / *OLD/ *NEW
Status of the library to be opened.

STATE = *ANY
The library may exist. If it does not exist, it will be created as a new library.

STATE = *OLD
The library must exist.

STATE = *NEW
The library must not exist. It will be created as a new library.

DEFAULTS = *UNCHANGED / *PARAMETERS(...)
Library-specific defaults for an LMS session.

TYPE = *UNCHANGED
No changes to the library-specific default type. By default, the library is not assigned a default type.

TYPE = *NONE
The library is not assigned a library-specific default type, or an assigned library-specific default type is canceled.

TYPE = <alphanum-name 1..8>
Name of the library-specific default type. This replaces the *LMS-DEFAULT type entry in all statements concerning this library.

SNAPSET = *NONE / *LATEST / <name 1..1 with-low> / <integer -52..-1>
The Operand allows to open libraries on a snapset.

The specification of a snapset is only allowed together with MODE=*READ (Default). Assignment of snapset libraries in LMS statements is only possible with LIBRARY=*STD. Other LIBRARY specification always refers to the original pubset. (see Example below)

To open a snapset library, a library with the same name must exist on the original pubset in order to evaluate some file attributes. Missing libraries can be restored as a whole by /RESTORE-FILE-FROM-SNAPSET.

Alternatively an empty library can be created by OPEN-LIBRARY *library*,U.

SNAPSET = *NONE
The library is not located on a snapset.

SNAPSET = *LATEST
The library is located on the latest created snapset.

*LATEST is equivalent to -1.

SNAPSET = <name 1..1 with-low>
The library is located on a snapset with snapset id a-z,A-Z.

Snapsets with capital letters A-Z are supported from BS2000/OSD-BC V8.0 on.

The snapset id of the library can be found out by the commands /LIST-FILE-FROM-SNAPSET and /SHOW-SNAPSET-CONFIGURATION.

SNAPSET = <integer -52..-1>
The library is located on a snapset in the chronological order of snapset creation with -1 for the newest snapset.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0211	Library already exists
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked

Required access rights

For MODE=*READ : Read authorization for LIBRARY

For MODE=*UPDATE : Read and write authorization for LIBRARY

Note

If the statement is aborted with LMS0024 and PLA0203, this could mean that an old library format is present. The way to check this and convert it into a PLAM library is described in the appendix in section “Migrating old library formats” on page 516.

Examples

– Opening an existing library LIB1:

```
//open-library library=lib1
```

– Opening an existing library via the link name:

```
/add-file-link link-name=glob-lib,file-name=lib1
//start-lms
.
.
//open-library library=*link(link-name=glob-lib)
```

– Creating a new library:

```
//open-library library=lib1new, mode=*update
```

– Opening a snapset library:
Library X is located on the original pubset and a snap of X on snapset a.

```
//open-lib x,snapset=a
```

The following statement copies the elements from snapset a to the original pubset.

```
//copy-elem (*std,*,*),(x)
```

By contrast the following statement only copies the elements within the original pubset.

```
//copy-elem (x,*,*),(y)
```

PROVIDE-ELEMENT

The PROVIDE-ELEMENT statement reserves members of a source library and makes copies of these members available in an output library. The source library is to be named explicitly. If WRITE-CONTROL is activated, the reserved members are protected against modification by anyone else. If a convention is also set, the entire version space above the specified version is reserved for the holder.

The source and target member base types may differ if text members are reserved.

```
PROVIDE-ELEMENT

ELEMENT = *LIBRARY-ELEMENT (...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
      *ALL(...)
        VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
                  <composed-name 1..24 with-under>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
      <composed-name 1..64 with-under with-wild(132)>(...)
        VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
                  <composed-name 1..24 with-under>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
    ,TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
```

(part 1 of 3)

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 2 of 3)

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**TO-E**LEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)

***BY-SOUR**CE(...)

VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..132 with-under with-wildcard-constr>(...)

VERSION = ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST-**EXIST**ING / ***INCR**EMENT / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / <alphanum-name 1..20 with-wild-constr>

,**USER-DATE** = ***BY-SOUR**CE / ***TODAY** / <date 8..10 with-compl>

,**STOR**AGE-**FORM** = ***LMS-DEF**AULT / ***STD** / ***FULL** / ***DELTA**

,**WR**ITE-**MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***ANY**

,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

(part 3 of 3)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation(s).

LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the members to be made available.

LIBRARY = <filename 1..54 without-vers>
Name of the library.

LIBRARY = *LINK(...)
The library assigned by means of the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
Names of the members to be made available.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the members to be made available.

VERSION = *HIGHEST-EXISTING
Makes available the member with the highest existing version with reference to BASE.

VERSION = *UPPER-LIMIT
Makes available the highest possible version X'FF' with the specified TYPE and name in the library.

VERSION = <composed-name 1..24 with-under>
A text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
The type of the members which are to be made available.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
Makes available members without regard to the date.

USER-DATE = *TODAY
Makes available only members with the current date.

USER-DATE = <date 8..10 with-compl>
Makes available only members with the specified date.

USER-DATE = *INTERVAL(...)
Makes available only members with a user date which falls within the specified interval.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see USER-DATE.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members which are to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are to be excluded.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are to be excluded. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies where and under what name the member is to be made available.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library into which the member is to be copied.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library into which the member is to be copied. If the library does not yet exist, it is created.

LIBRARY = *LINK(...)
The library assigned by means of the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *BY-SOURCE (...) /
<composed-name 1..132 with-under with-wildcard-constr>(...)
Name to be given to the target member.

ELEMENT = *BY-SOURCE(...)
The target member is given the same name as the source member.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version which the target member is to receive.

VERSION = *BY-SOURCE
The target member is given the same version as the source member.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..24 with-under>
The target member receives the version specified here.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member.

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Specifies the name which the target member is to receive. The name may also be entered with wildcards.

VERSION = *LMS-DEFAULT / *BY-SOURCE / *HIGHEST-EXISTING / *INCREMENT / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version which the target member is to receive. For a description of the operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member.

TYPE = *BY-SOURCE / *LMS-DEFAULT / <alphanum-name 1..20 with-wild-constr>
Type which the target member is to receive.

TYPE = *BY-SOURCE
The target member is given the same type designation as the source member.

USER-DATE = *BY-SOURCE / *TODAY / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *BY-SOURCE
The new member is given the same date as the source member.

USER-DATE = *TODAY
The current date is given.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

STORAGE-FORM = *LMS-DEFAULT / *STD / *FULL / *DELTA
Storage form for the member being copied. The storage form must not contradict the settings made by means of the MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES statements, and all members of a given type and name must have the same storage form.

STORAGE-FORM = *STD
The member is generated in accordance with the storage form required for the member scope. Contradictory requirements result in errors. If nothing is specified, full storage is selected.

STORAGE-FORM = *FULL
The new member is generated as a full member (if this is not possible, an error message is issued).

STORAGE-FORM = *DELTA
The new member is generated as a delta member (if this is not possible, an error message is issued). This entry is permissible for member types S, P, D, J, M, X and members types derived from them.

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *ANY
Controls overwriting.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member already exists and is replaced.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For ELEMENT:	Read authorization for LIBRARY
	Read and hold authorization for ELEMENT
For TO-ELEMENT:	Read and write authorization for LIBRARY

Administer authorization where the specified member designation is new. Otherwise, only write authorization for the member existing under the specified member designation (administer authorization no longer required).

If members are to be stored in delta form, the user must have read authorization for the member defined by BASE. If WRITE-CONTROL is active and a base version exists, the USERID of the user must be entered as the HOLDER of the member specified by BASE. Only if write authorization has been granted can a new version be generated or this base version overwritten. In this case, administer authorization is no longer required.

Notes

– When creating a member, be sure to take into account the convention applicable to the member type.
  Especially when the target type has the convention STD-TREE, problems can occur if the source side contains side branch versions whose main branch version is deleted. In this case, the affected side branches cannot be copied; LMS issues an error message.

– If WRITE-CONTROL is active in the output library, the access method adds a comment (record type 2) to the member which is to be written. The comment logs the HOLDER, DATE and TIME of the process. The record is written as the first record of the record type. Any comment records which already exist are copied after it. If, in addition, the member is written to the base of a different version (i.e. not the first version under a name), the member attributes STATE and HOLDER and all the rights of the base version are adopted for the new version. The CCSN is adopted from the source file. The USER-DATE is determined anew.

– The user is entered as the HOLDER of the source member. The source member has the status 'IN-HOLD', so it is no longer possible to issue a PROVIDE-ELEMENT statement for that member until it regains the FREE status. This can be implemented by RETURN-ELEMENT or MODIFY-ELEMENT-ATTRIBUTES.

REORGANIZE-LIBRARY

The REORGANIZE-LIBRARY statement reorganizes a library in such a way that as much as possible of the disk space not used for members is left for the library file, which is then released. This reduces the amount of disk space required for a library.

```
REORGANIZE-LIBRARY

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
  *LINK(...)
    | LINK-NAME = <structured-name 1..8>
```

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library which is to be reorganized.

LIBRARY = *STD
Reorganizes the library which was globally opened by means of OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Reorganizes the library with the name specified here.

LIBRARY = *LINK(...)
Reorganizes the library assigned by means of the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library, which was declared with a /ADD-FILE-LINK command.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked

Required access rights

Ownership, read and write authorization for LIBRARY.

Notes

– During a reorganization, no other access to the library is permitted (open mode: SHARUPD=NO). This is why a REORGANIZE-LIBRARY statement is rejected if the library is already open for a library application, e.g. through another task. If the library which is to be reorganized was opened earlier in the same LMS run, LMS implicitly closes the library before the reorganization, unless the LMS log is to be written to that library. In that case, users must themselves close the library by redirecting the log stream.

– The disk space which remains reserved, but unused after the reorganization may be as large as the largest member in the library. If this minimum is not reached, e.g. because following a system error blocks at the back may already be reserved but are not yet used, or to minimize the size of the library, use a buffer for copying (COPY-LIBRARY).

RESET-LMS-DEFAULTS

This statement resets the defaults that were set with MODIFY-LMS-DEFAULTS to the values that applied at the beginning of the LMS run.

RESET-LMS-DEFAULTS

The RESET-LMS-DEFAULTS statement has no operands.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

RESET-LOGGING-PARAMETERS

This statement resets the parameters that were set with MODIFY-LOGGING-PARAMETERS to the values that applied at the beginning of the LMS run.

RESET-LOGGING-PARAMETERS

The RESET-LOGGING-PARAMETERS statement has no operands.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

RESET-TYPE-ATTRIBUTES

This statement resets all attributes of the specified type to the default values of the MODIFY-TYPE-ATTRIBUTES statement.

```
RESET-TYPE-ATTRIBUTES

 LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
    *LINK(...)
       | LINK-NAME = <structured-name 1..8> / <filename 1..8>
,TYPE = <alphanum-name 1..8>
```

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library in which the type attributes are to be reset.

LIBRARY = *STD
The library opened by means of OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library in which the type attributes are to be reset.

LIBRARY = *LINK(..)
The library assigned by means of the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library which was declared with a /ADD-FILE-LINK command before LMS was invoked, and which must be known to LMS.

TYPE = <alphanum-name 1..8>
Type of the member whose attributes are to be reset.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted
	130	LMS0413	Type locked

RETURN-ELEMENT

Provided that the user has reserved the base specified for the target version in the output library, the RETURN-ELEMENT statement copies members of a source library into the output library, deletes the members from the source library and then releases the reservations in the output library.

For the first version of a member, no base yet exists. In this case, RETURN-ELEMENT requires administer authorization.

The source and target member base types may differ if text members are returned.

The access method adds a comment (record type 2) to the member which is to be written. The comment logs the HOLDER, DATE and TIME of the operation. This record is written as the first record of its record type and is followed by the comment record containing the text specified for COMMENT. Any comment records which already existed are copied after it. If, in addition, the member is written to the base of a different version (i.e. not the first version under a name), all the rights of the base version are applied to the new version. STATE is set to FREE for both versions. The CCSN is adopted from the source file. The USER-DATE is determined anew.

```
RETURN-ELEMENT

ELEMENT = *LIBRARY-ELEMENT (...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
      *ALL(...)
        VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
                  <composed-name 1..24 with-under>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
      <composed-name 1..64 with-under with-wild(132)>(...)
        VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
                  <composed-name 1..24 with-under>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
```

(part 1 of 3)

,**TYPE** = **<u>*LMS-DEF</u>**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>
,**USER-DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
*INTERVAL(...)
FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>
,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>
,**CRE**ATION-**DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
*INTERVAL(...)
FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>
,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>
,**MOD**IFICATION-**DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
*INTERVAL(...)
FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>
,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>
,**EXCEPT-ELEM**ENT = **<u>*NONE</u>** / ***ELEM**ENT(...)
*ELEMENT(...)
ELEMENT = **<u>*ANY</u>** (...) / <composed-name 1..64 with-under with-wild(132)>(...)
*ANY(...)
VERSION = **<u>*ANY</u>** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>
,**BASE** = **<u>*STD</u>** / <composed-name 1..24 with-under with-wild>
<composed-name 1..64 with-under with-wild(132)>(...)
VERSION = **<u>*ANY</u>** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under>
,**BASE** = **<u>*STD</u>** / <composed-name 1..24 with-under with-wild>
,**TYPE** = **<u>*ANY</u>** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>
,**USER-DATE** = **<u>*ANY</u>** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
*INTERVAL(...)
FROM = **<u>1900-01-01</u>** / <date 8..10 with-compl>
,**TO** = **<u>*TODAY</u>** / <date 8..10 with-compl>

(part 2 of 3)

,**CRE**ATION**-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION**-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**TO-E**LEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

LIBRARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***BY-SOUR**CE (...) / <composed-name 1..132 with-under with-wild-constr>(...)

***BY-SOUR**CE(...)

VERSION = ***INCR**EMENT / ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST**-EXIST**ING / ***UP**PER**-LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..132 with-under with-wildcard-constr>(...)

VERSION = ***INCR**EMENT / ***LMS-DEF**AULT / ***BY-SOUR**CE / ***HIGH**EST**-EXIST**ING / ***UP**PER**-LIM**IT / <composed-name 1..24 with-under>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***BY-SOUR**CE / ***LMS-DEF**AULT / <alphanum-name 1..20 with-wild-constr>

,**USER-DATE** = ***BY-SOUR**CE / ***TODAY** / <date 8..10 with-compl>

,**STOR**AGE**-FORM** = ***LMS-DEF**AULT / ***STD** / ***FULL** / ***DELTA**

,**COMMENT** = ***BY-SOUR**CE / <c-string 1..72 with-low>

,**WR**ITE**-MODE** = ***LMS-DEF**AULT / ***CREATE** / ***REPLACE** / ***ANY**

,**DIALOG-CONTR**OL = ***LMS-DEF**AULT / ***NO** / ***Y**ES / ***ERROR**

(part 3 of 3)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation(s).

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the relevant members.

LIBRARY = <filename 1..54 without-vers>
Name of the library.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *LINK(...)
The library assigned by means of the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL / <composed-name 1..64 with-under with-wild(132)>(...)
Names of the members to be returned.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>
Version of the members to be returned.

VERSION = *HIGHEST-EXISTING
Returns the member with the highest existing version with reference to BASE.

VERSION = *UPPER-LIMIT
Returns the highest possible version X'FF' with the specified TYPE and name in the library.

VERSION = <composed-name 1..24 with-under>
A text specified here is interpreted as the version designation.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
The type of the members which are to be returned.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
Returns members without regard to this date.

USER-DATE = *TODAY
Returns only members with the current date.

USER-DATE = <date 8..10 with-compl>
Returns only members with the specified date.

USER-DATE = *INTERVAL(...)
Returns only members with a user date which falls within the specified interval.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members which are to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are to be excluded.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are to be excluded. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.

For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies where and under what name the member is to be copied.

LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Specifies the library into which the member is to be copied.

LIBRARY = <filename 1..54 without-vers>
Name of the library into which the member is to be copied. If the library does not yet exist, it is created.

LIBRARY = *LINK(...)
The library assigned by means of the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wild-constr>(...)
Name to be given to the target member.

ELEMENT = *BY-SOURCE(...)
The target member is given the same name as the source member.

VERSION = *INCREMENT / *LMS-DEFAULT / *BY-SOURCE /
***HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>**
Version which the target member is to receive.

VERSION = *INCREMENT
Depending on the convention applicable for the type, this generates a new, higher version among existing members having the same type and name; otherwise a default version is generated.

VERSION = *BY-SOURCE
The target member is given the same version as the source member.

VERSION = *HIGHEST-EXISTING
Depending on the convention applicable for the type, this overwrites the highest existing version with reference to BASE among the members of the same type and name; otherwise a default version is generated.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..24 with-under>
The target member receives the version specified here.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member. If BASE=*STD applies, the version held by the user is the base version (in the event of ambiguity, an error message is issued).

ELEMENT = <composed-name 1..132 with-under with-wild-constr>(...)
Specifies the name which the target member is to receive. The name may also be entered with wildcards.

VERSION = *INCREMENT / *LMS-DEFAULT / *BY-SOURCE /
***HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under>**
Version which the target member is to receive. For a description of the operands, see above.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Defines the base for the target member.

TYPE = *BY-SOURCE / *LMS-DEFAULT /
<alphanum-name 1..20 with-wildcard-constr>
Type which the target member is to receive.

TYPE = *BY-SOURCE
The target member is given the same type designation as the source member.

USER-DATE = *BY-SOURCE / *TODAY / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *BY-SOURCE
The new member is given the same date as the source member.

USER-DATE = *TODAY
The current date is given.

USER-DATE = <date 8..10 with-compl>
The date must be entered in the form [YY]YY-MM-DD.

STORAGE-FORM = *LMS-DEFAULT / *STD /*FULL / *DELTA
Storage form for the member being generated. The storage form must not contradict the settings made by means of the MODIFY-TYPE-ATTRIBUTES or MODIFY-LIBRARY-ATTRIBUTES statements, and all members of a given type and name must have the same storage form.

STORAGE-FORM = *STD
The member is generated in accordance with the storage form required for the member scope. Contradictory requirements result in errors. If nothing is specified, full storage is selected.

STORAGE-FORM = *FULL
The new member is generated as a full member (if this is not possible, an error message is issued).

STORAGE-FORM = *DELTA
The new member is generated as a delta member (if this is not possible, an error message is issued). This entry is permissible for member types S, P, D, J, M, X and members types derived from them.

COMMENT = *BY-SOURCE / <c-string1..72 with-low>
For the specification of a comment text.

COMMENT = *BY-SOURCE
Adopts the comment text from the source member.

COMMENT = <c-string 1..72 with-low>
Comment text which is to be inserted in the target member. The comment text of the source is also retained.

WRITE-MODE = *LMS-DEFAULT / *CREATE / *REPLACE / *ANY
Controls overwriting.

WRITE-MODE = *CREATE
The new member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The new member already exists and is replaced.

WRITE-MODE = *ANY
The new member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *LMS-DEFAULT / *NO / *YES / *ERROR
This operand determines whether or not a dialog is to be conducted with the user during execution of a statement.

For more detailed information on dialog control, see the MODIFY-LMS-DEFAULTS statement.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0020	Target member or target file does not exist
	64	LMS0213	Name exists as delta member
	64	LMS0214	Name exists as full member
	64	LMS0302	Member not found
	64	LMS0509	Target member or target file already exists
	64	LMS0510	Base not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0223	Only leaves of a delta tree can be overwritten
	64	PLA0224	Storage form not allowed
	64	PLA0229	No access right for the member
	64	PLA0233	Borrow status prevents member access
	64	PLA0475	Function violates version automation
	64	PLA0476	Version does not match applicable convention
	64	PLA0478	Increase causes version overflow
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For ELEMENT: Read authorization for LIBRARY and ELEMENT

For TO-ELEMENT: Read and write authorization for LIBRARY

Administer authorization where the specified member designation is new. Otherwise, only write authorization for the member existing under the specified member designation (administer authorization no longer required).

If members are to be stored in delta form, the user must have read authorization for the member defined by BASE.

If WRITE-CONTROL is active and a base version exists, the USERID of the user must be entered as the HOLDER of the member specified by BASE. Only if write authorization has been granted can a new version be generated or this base version overwritten. In this case, administer authorization is no longer required.

Note

When creating a member, be sure to take into account the convention applicable to the member type. Especially when the target type has the convention STD-TREE, problems can occur if the source side contains side branch versions whose main branch version is deleted. In this case, the affected side branches cannot be copied; LMS issues an error message.

SHOW-ELEMENT

SHOW-ELEMENT displays the contents of a specified member, depending on its type. The contents of text-oriented members, modules, phases and link and load modules (LLM) can be output. The representation format of the output is controlled by the OUTPUT-FORM operand. The meaning of the attributes with modules and link and load modules is explained in [4].

The statement is permissible for all member types. User-defined types are handled according to their respective base type. If the base type is unknown to LMS, only the TEXT-INFORMATION and OUTPUT-FORM operands are effective.

```
SHOW-ELEMENT

ELEMENT = *LIBRARY-ELEMENT (...)
  *LIBRARY-ELEMENT(...)
    LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
      *LINK(...)
        LINK-NAME = <structured-name 1..8>
    ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
      *ALL(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
      <composed-name 1..64 with-under with-wild(132)>(...)
        VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
                  <composed-name 1..24 with-under with-wild(52)>
        ,BASE = *STD / <composed-name 1..24 with-under with-wild>
    ,TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
    ,USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
      *INTERVAL(...)
        FROM = 1900-01-01 / <date 8..10 with-compl>
        ,TO = *TODAY / <date 8..10 with-compl>
```

(part 1 of 5)

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**EXCEPT-ELEM**ENT = ***NONE** / ***ELEM**ENT(...)

***ELEM**ENT(...)

ELEMENT = ***ANY** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ANY**(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ANY** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***ANY** / ***LMS-DEF**AULT / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 2 of 5)

,**TEXT-INF**ORMATION = ***LMS-DEF**AULT / ***ALL** / ***STATIS**TICS / ***F**ILE-**ATTR**IBUTES / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

INFORMATION = ***LMS-DEF**AULT / ***ALL** / list-poss(2): ***TEXT** / ***COMMENT**

,**REC**ORD-**RANGE** = ***LMS-DEF**AULT / ***ALL** / ***RANGE**(...)

***RANGE**(...)

FROM = ***LMS-DEF**AULT / <integer 1..2147483647>

,**TO** = ***LMS-DEF**AULT / ***LAST** / <integer 1..2147483647>

,**REC**ORD-**PART** = ***LMS-DEF**AULT / ***ALL** / ***PART**(...)

***PART**(...)

START = ***LMS-DEF**AULT / <integer 1..262144>

,**LEN**GTH = ***LMS-DEF**AULT / ***REST** / <integer 0..262144>

,**REC**ORD-**NUM**BER = ***LMS-DEF**AULT / ***BY-OUT**PUT / ***Y**ES / ***NO**

,**MOD**ULE-**INF**ORMATION = ***LMS-DEF**AULT / ***ALL** / ***STATIS**TICS / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

INFORMATION = ***LMS-DEF**AULT / ***ALL** / ***TXT**(...) / ***TXTP**(...) /
list-poss(9): ***ESD** / ***ISD** / ***LSD** / ***RLD** / ***REP** / ***INCL**UDE / ***DSDD** / ***REF** / ***END**

***TXT**(...)

CSECT-NAME = ***LMS-DEF**AULT / ***ALL** / <c-string 1..32 with-low> / <text 1..32>

,**ADDR**ESS = ***LMS-DEF**AULT (...) / <x-string 1..8>(...)

***LMS-DEF**AULT(...)

BASE-ADDRESS = ***LMS-DEF**AULT / <x-string 1..8>

<x-string 1..8>(...)

BASE-ADDRESS = ***LMS-DEF**AULT / <x-string 1..8>

,**LEN**GTH = ***LMS-DEF**AULT / ***REST** / <integer 1..2147483647> / <x-string 1..8>

***TXTP**(...)

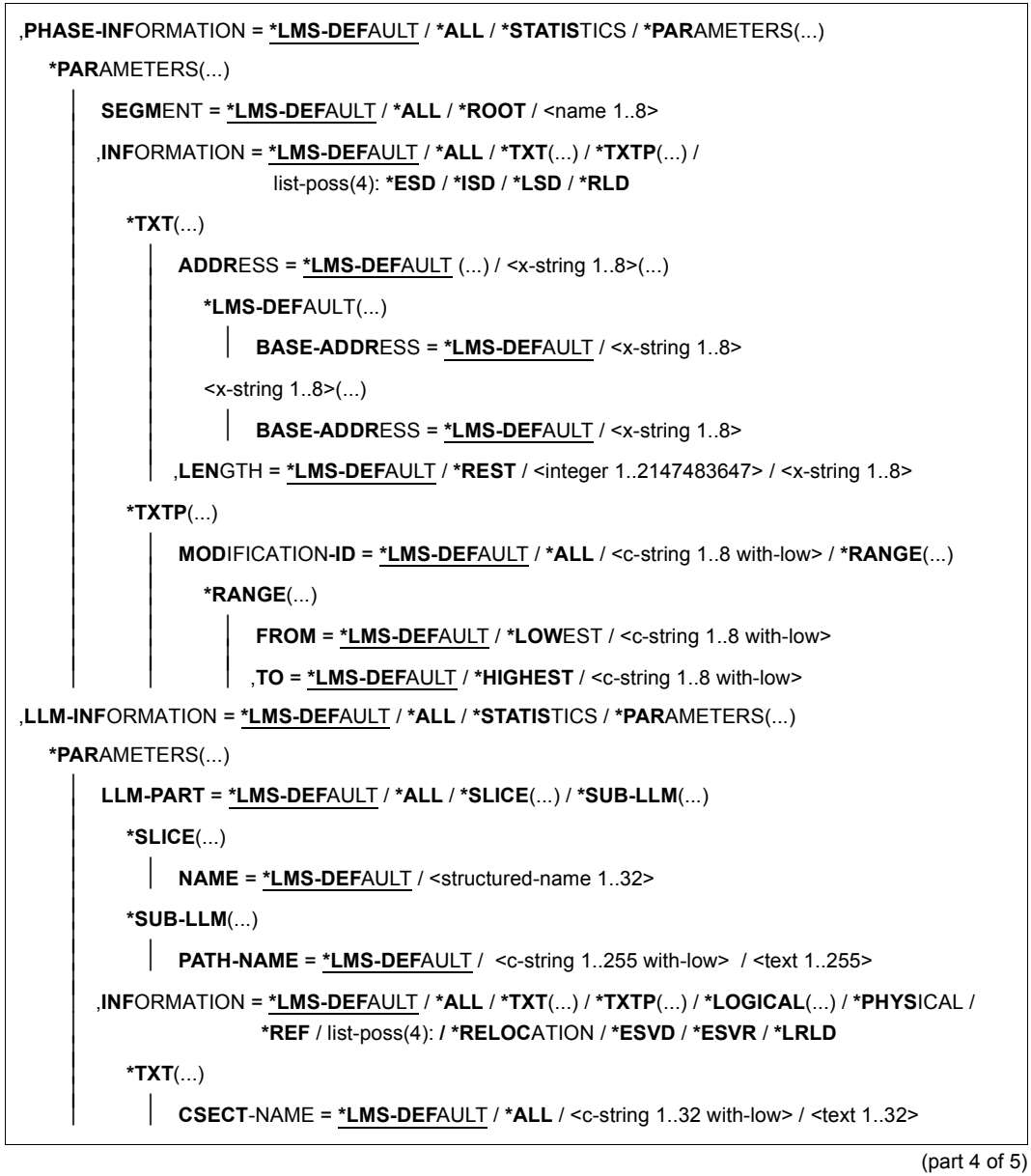
MODIFICATION-**ID** = ***LMS-DEF**AULT / ***ALL** / <c-string 1..8 with-low> / ***RANGE**(...)

***RANGE**(...)

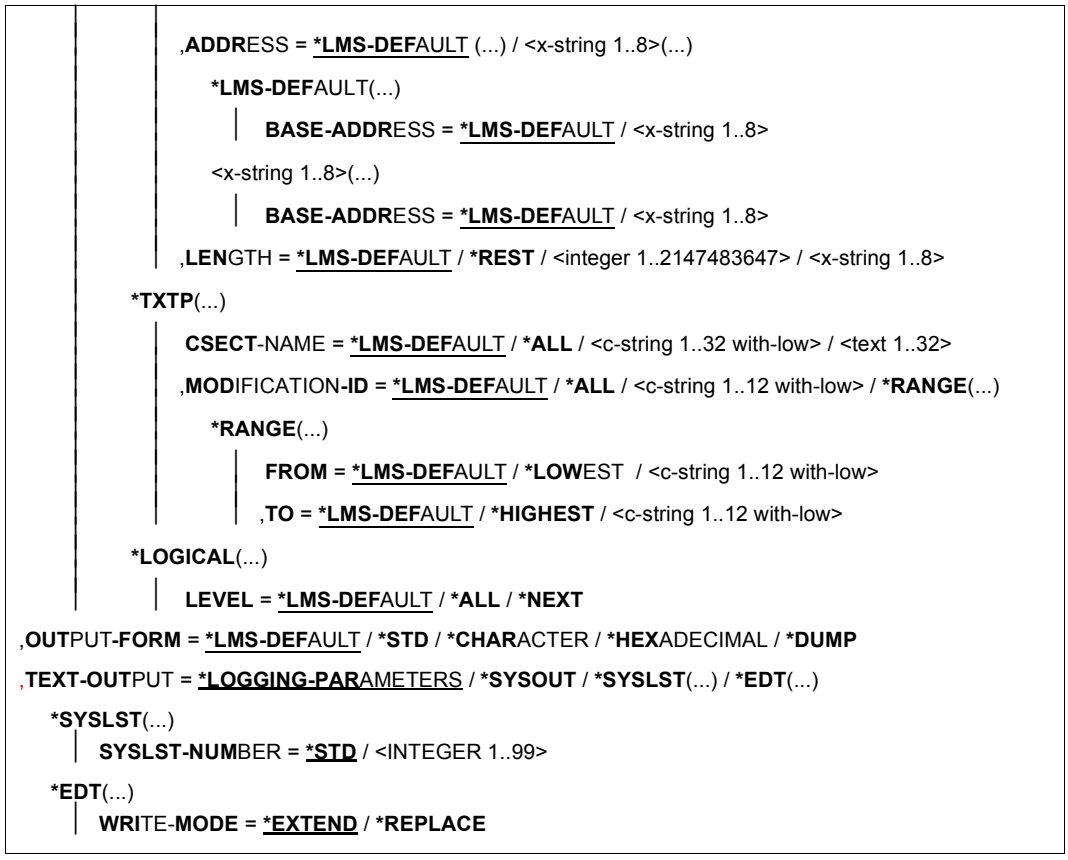
FROM = ***LMS-DEF**AULT / ***LOW**EST / <c-string 1..8 with-low>

,**TO** = ***LMS-DEF**AULT / ***HIGHEST** / <c-string 1..8 with-low>

(part 3 of 5)


,**PHASE-INF**ORMATION = ***LMS-DEF**AULT / ***ALL** / ***STATIS**TICS / ***PAR**AMETERS(...)
***PAR**AMETERS(...)
SEGMENT = ***LMS-DEF**AULT / ***ALL** / ***ROOT** / <name 1..8>
,**INF**ORMATION = ***LMS-DEF**AULT / ***ALL** / ***TXT**(...) / ***TXTP**(...) /
list-poss(4): ***ESD** / ***ISD** / ***LSD** / ***RLD**
***TXT**(...)
ADDRESS = ***LMS-DEF**AULT (...) / <x-string 1..8>(...)
***LMS-DEF**AULT(...)
BASE-ADDRESS = ***LMS-DEF**AULT / <x-string 1..8>
<x-string 1..8>(...)
BASE-ADDRESS = ***LMS-DEF**AULT / <x-string 1..8>
,**LEN**GTH = ***LMS-DEF**AULT / ***REST** / <integer 1..2147483647> / <x-string 1..8>
***TXTP**(...)
MODIFICATION-**ID** = ***LMS-DEF**AULT / ***ALL** / <c-string 1..8 with-low> / ***RANGE**(...)
***RANGE**(...)
FROM = ***LMS-DEF**AULT / ***LOW**EST / <c-string 1..8 with-low>
,**TO** = ***LMS-DEF**AULT / ***HIGHEST** / <c-string 1..8 with-low>
,**LLM-INF**ORMATION = ***LMS-DEF**AULT / ***ALL** / ***STATIS**TICS / ***PAR**AMETERS(...)
***PAR**AMETERS(...)
LLM-PART = ***LMS-DEF**AULT / ***ALL** / ***SLICE**(...) / ***SUB-LLM**(...)
***SLICE**(...)
NAME = ***LMS-DEF**AULT / <structured-name 1..32>
***SUB-LLM**(...)
PATH-NAME = ***LMS-DEF**AULT / <c-string 1..255 with-low> / <text 1..255>
,**INF**ORMATION = ***LMS-DEF**AULT / ***ALL** / ***TXT**(...) / ***TXTP**(...) / ***LOGICAL**(...) / ***PHYS**ICAL /
***REF** / list-poss(4): **/ *RELOC**ATION / ***ESVD** / ***ESVR** / ***LRLD**
***TXT**(...)
CSECT-NAME = ***LMS-DEF**AULT / ***ALL** / <c-string 1..32 with-low> / <text 1..32>

(part 4 of 5)



(part 5 of 5)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

ELEMENT = *ALL / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be displayed.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member to be output.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version with reference to BASE is output.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is displayed.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be displayed.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be output.

USER-DATE = *ANY / *TODAY / <date8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be output has any date.

USER-DATE = *TODAY
The member with the current date is displayed.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is displayed.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are displayed.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see USER-DATE.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see USER-DATE.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded from correction.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members that are not to be displayed. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see the *LIBRARY-ELEMENT operand of this statement.

TEXT-INFORMATION = *LMS-DEFAULT / *ALL / *STATISTICS / *FILE-ATTRIBUTES / *PARAMETERS(...)
Defines the information scope for all members except member types R, C and L. For PAM members, all specifications apart from *FILE-ATTRIBUTES have the same effect as *ALL.

TEXT-INFORMATION = *ALL
Everything is output.

TEXT-INFORMATION = *STATISTICS
The number of records per record type and the total number of records are output. For each record type, the total of all record lengths (without record length fields) is output, as is the total record length across all record types.

TEXT-INFORMATION = *FILE-ATTRIBUTES
Only the stored file attributes are output.

TEXT-INFORMATION = *PARAMETERS(...)
Defines a member extract to be output.

INFORMATION = *LMS-DEFAULT / *ALL / list-poss(2): *TEXT / *COMMENT
The member section to be output.

INFORMATION = *ALL
Everything is output.

INFORMATION = *TEXT
Outputs the text itself, i.e. record type 1.

INFORMATION = *COMMENT
Outputs the separately stored comment, i.e. record type 2.

RECORD-RANGE = *LMS-DEFAULT / *ALL / *RANGE(...)
The section of the member to be processed.

RECORD-RANGE = *ALL
All records are processed.

RECORD-RANGE = *RANGE(...)
Specifies the range of record numbers which is to be processed. The record numbers refer not to a record type, but to the section of the member designated by means of INFORMATION=. Within this section, the records are numbered consecutively from 1 through n.

FROM = *LMS-DEFAULT / <integer 1..2147483647>
Beginning of the range, specified by the first record number. Record number 1 is the default value.

TO = *LMS-DEFAULT / *LAST / <integer 1..2147483647>
End of the range, specified by the last record number. The last record number is used as the default value.

RECORD-PART = *LMS-DEFAULT / *ALL / *PART(...)
Specifies the part of the record to be processed.

RECORD-PART = *ALL
Processes the entire record.

RECORD-PART = *PART(...)
Specifies the part of the record to be processed. If the default values are not changed, the entire record is processed.

START = *LMS-DEFAULT / <integer 1..262144>
Beginning of the record part, specified by the first character in the record. The first character is used as the default value.

LENGTH = *LMS-DEFAULT / *REST / <integer 0..262144>
Length of the record part. The remainder of the record is used as the default value.

RECORD-NUMBER = *LMS-DEFAULT / *BY-OUTPUT / *YES / *NO
Specifies output of the record numbers.

RECORD-NUMBER = *BY-OUTPUT
Only if the output is directed to SYSOUT will no record numbers be output. With any other output medium, the record numbers are included in the output.

RECORD-NUMBER = *YES
The record numbers are also output to SYSOUT.

RECORD-NUMBER = *NO
No record numbers are included in the output.

MODULE-INFORMATION = *LMS-DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for object modules (R-type members).

MODULE-INFORMATION = *ALL
Everything is output.

MODULE-INFORMATION = *STATISTICS
The name, length and address of the CSECTS and also the overall length of the module are output.

MODULE-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

INFORMATION = *LMS-DEFAULT / *ALL / *TXT(...) / *TXTP(...) / list-poss(9): *ESD / *ISD / *LSD / *RLD / *REP / *INCLUDE / *DSDD / *REF / *END
The record types listed here can be selected.

INFORMATION = *TXT(...)
Text records are selected.

CSECT-NAME = *LMS-DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
The text records can be restricted to one CSECT.

ADDRESS = *LMS-DEFAULT / <x-string 1..8>(...)
Start address of the text. The default setting is X‘00000000‘.

BASE-ADDRESS = *LMS-DEFAULT / <x-string 1..8>
The base address specified here is added to the start address. The default setting is X‘00000000‘.

LENGTH = *LMS-DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
TXTP records are output.

MODIFICATION-ID = *LMS-DEFAULT / *ALL / <c-string 1..8 with-low> / *RANGE(...)
Those TXTP records with the specified identification are selected.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *LMS-DEFAULT / *LOWEST / <c-string 1..8 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *LMS-DEFAULT / *HIGHEST / <c-string 1..8 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

PHASE-INFORMATION = *LMS-DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for phases (C-type members).

PHASE-INFORMATION = *ALL
Everything is output.

PHASE-INFORMATION = *STATISTICS
The name, length and address of the segment and also the overall length of the segment are output.

PHASE-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

SEGMENT = *LMS-DEFAULT / *ALL / *ROOT / <name 1..8>
Phase segment that is selected.

INFORMATION = *LMS-DEFAULT / *ALL / *TXT(...) / *TXTP(...) / list-poss(4): *ESD / *ISD / *LSD / *RLD
The record types listed here can be selected.

INFORMATION = *TXT(...)
Text records are selected.

ADDRESS = *LMS-DEFAULT(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *LMS-DEFAULT / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *LMS-DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
TXTP records are output.

MODIFICATION-ID = *LMS-DEFAULT / *ALL / <c-string 1..8 with-low> / *RANGE(...)
Those TXTP records with the specified identification are selected.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *LMS-DEFAULT / *LOWEST / <c-string 1..8 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *LMS-DEFAULT / *HIGHEST / <c-string 1..8 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

LLM-INFORMATION = *LMS-DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for link and load modules (L-type members).

LLM-INFORMATION = *ALL
Everything is output.

LLM-INFORMATION = *STATISTICS
General information on the link and load module (name, copyright, ...) is output.

LLM-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

LLM-PART = *LMS-DEFAULT / *ALL / *SLICE(...) / *SUB-LLM(...)
Specifies the LLM part to be selected. By default the entire LLM is selected.

LLM-PART = *SLICE(...)
Specifies the slice to be output.

NAME = *LMS-DEFAULT / <structured-name 1..32>
Name of the slice to be output.

LLM-PART = *SUB-LLM(...)
Specifies the sub-LLM to be output.

PATH-NAME = *LMS-DEFAULT / <c-string 1..255 with-low> / <text 1..255>
The sub-LLM to be output is determined by way of its path name.

INFORMATION = *LMS-DEFAULT / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF / list-poss(4): *RELOCATION/ *ESVD / *ESVR / *LRLD
The record types listed here can be selected.

INFORMATION = *TXT(...)
Text records are selected.

CSECT-NAME = *LMS-DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
The text records can be restricted to one CSECT.

ADDRESS = *LMS-DEFAULT(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *LMS-DEFAULT / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *LMS-DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
TXTP records are output.

CSECT-NAME = *LMS-DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
The TXTP records can be restricted to one CSECT.

MODIFICATION-ID = *LMS-DEFAULT / *ALL / <c-string 1..12 with-low> / *RANGE(...)
Those TXTP records with the specified identification are selected.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *LMS-DEFAULT / *LOWEST / <c-string 1..12 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *LMS-DEFAULT / *HIGHEST / <c-string 1..12 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

INFORMATION = *LOGICAL(...)
Outputs the logical structure of the LLM.

LEVEL = *LMS-DEFAULT / *ALL / *NEXT
Outputs all substructures by default; otherwise, only the next substructure.

INFORMATION = *PHYSICAL
Outputs the physical structure of the LLM.

OUTPUT-FORM = *LMS-DEFAULT / *STD / *CHARACTER / *HEXADECIMAL / *DUMP
Specifies the form of representation for the output.

OUTPUT-FORM = *STD
The form of representation is selected dependent on the member type. For text members this operand works in the same way as for OUTPUT-FORM = *CHARACTER.

OUTPUT-FORM = *CHARACTER
The output is in alphanumeric form.

OUTPUT-FORM = *HEXADECIMAL
The output is in alphanumeric and hexadecimal form, one above the other.

OUTPUT-FORM = *DUMP
The output is in alphanumeric and hexadecimal form, side by side. For member types S, P, D, J and M, this operand has the same effect as OUTPUT-FORM=*HEXADECIMAL.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *SYSOUT / *SYSLST(...) / *EDT(...)
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified with //MODIFY-LOGGING-PARAMETERS, TEXT-OUTPUT=.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
Output is to the work file 9 of EDT. If an error occurs during log output, then the system switches to the default output stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output will be added to this data. If there is no data in the file, the output will be written at the beginning of the file.

WRITE-MODE = *REPLACE
The output will be written at the beginning of work file 9. Any data that is already in the file will be replaced.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0084	VTSUCB macro error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	64	PLA0229	No access right for the member
	130	LMS0041	System address space exhausted
	130	LMS0411	Library locked
	130	LMS0412	Member locked
	130	LMS0413	Type locked

Required access rights

For LIBRARY-ELEMENT: read authorization for LIBRARY and ELEMENT

If more than one member is affected by the statement, members without read authorization are excluded from the statement.

Example

The member LETTER.A, which contains the text 'Dear ...', is to be output.

```
//show-element (element=letter.a,type=d)
INPUT  LIBRARY= :1OSQ:$USER.LIB.SHOW
INPUT  ELEMENT= (D)LETTER.A/@(0001)/2011-04-09
 DEAR ...


NUMBER OF PROCESSED RECORDS IS       1
```

SHOW-ELEMENT-ATTRIBUTES

SHOW-ELEMENT-ATTRIBUTES outputs the directory entries of the specified members or of the entire library.

The entries are output on the medium specified by the MODIFY-LOGGING-PARAMETERS statement. With the STRUCTURE-OUTPUT operand, the output can also be placed in a structured S variable (list).

The directory is always output sorted by type. The remainder of the sort sequence is determined by the SORT operand. The default sort sequence is type, name and version.

The INFORMATION and LAYOUT operands are used to specify the scope and format of the directory output. By default, the type, name, version, variant number and date are output.

With the aid of the SECONDARY-NAME and SECONDARY-ATTRIBUTE operands, the directory can be limited to the members containing a certain reference entry.

Note

In order to obtain the entire contents of a library (all members with all versions), it is sufficient to specify only SHOW-ELEMENT-ATTRIBUTES without any operands, provided no specific member type or version was set using MODIFY-LMS-DEFAULTS.

SHOW-ELEMENT-**ATTR**IBUTES

ELEMENT = ***LIB**RARY-**ELEM**ENT (...)

***LIB**RARY-**ELEM**ENT(...)

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

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

,**ELEM**ENT = ***ALL** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

***ALL**(...)

VERSION = ***ALL** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

<composed-name 1..64 with-under with-wild(132)>(...)

VERSION = ***ALL** / ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24 with-under with-wild(52)>

,**BASE** = ***STD** / <composed-name 1..24 with-under with-wild>

,**TYPE** = ***LMS-DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

(part 1 of 5)

,**ACCESS-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **1900-01-01** / <date 8..10 with-compl>

,**TO** = ***TODAY** / <date 8..10 with-compl>

,**USER-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **00:00:00** / <time 1..8>

,**TO** = **23:59:59** / <time 1..8>

,**CRE**ATION**-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **00:00:00** / <time 1..8>

,**TO** = **23:59:59** / <time 1..8>

,**MOD**IFICATION**-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **00:00:00** / <time 1..8>

,**TO** = **23:59:59** / <time 1..8>

,**ACCESS-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)

***INTERVAL**(...)

FROM = **00:00:00** / <time 1..8>

,**TO** = **23:59:59** / <time 1..8>

,**CODED-CHAR**ACTER**-SET** = ***ANY** / ***NONE** / <name 1..8 with-wild(20)>

,**STATE** = ***ANY** / ***FREE** / ***IN-HOLD**(...)

***IN-HOLD**(...)

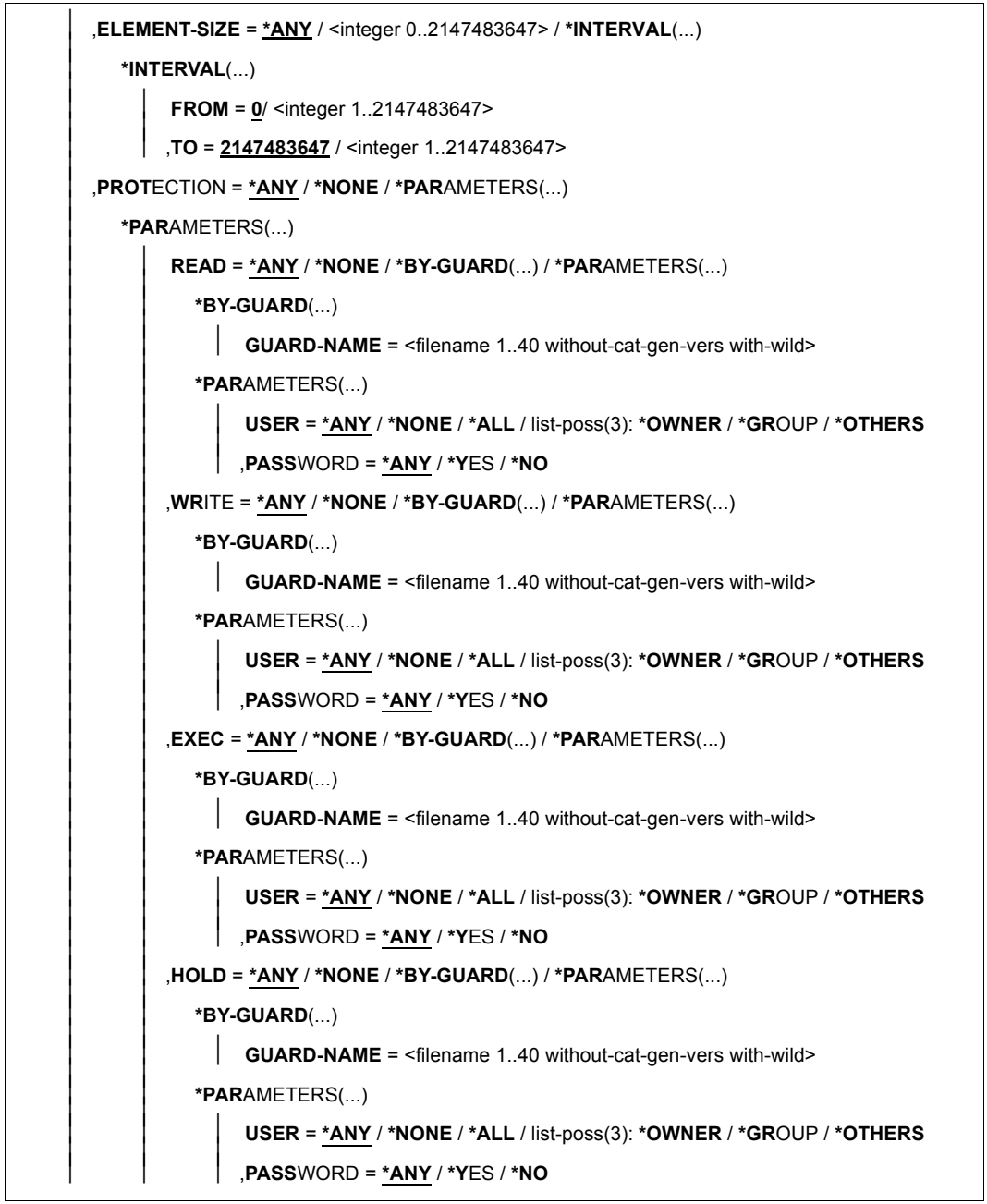
HOLDER = ***ANY** / <name 1..8 with-wild(20)>

,**STOR**AGE**-FORM** = ***ANY** / ***FULL** / ***DELTA**

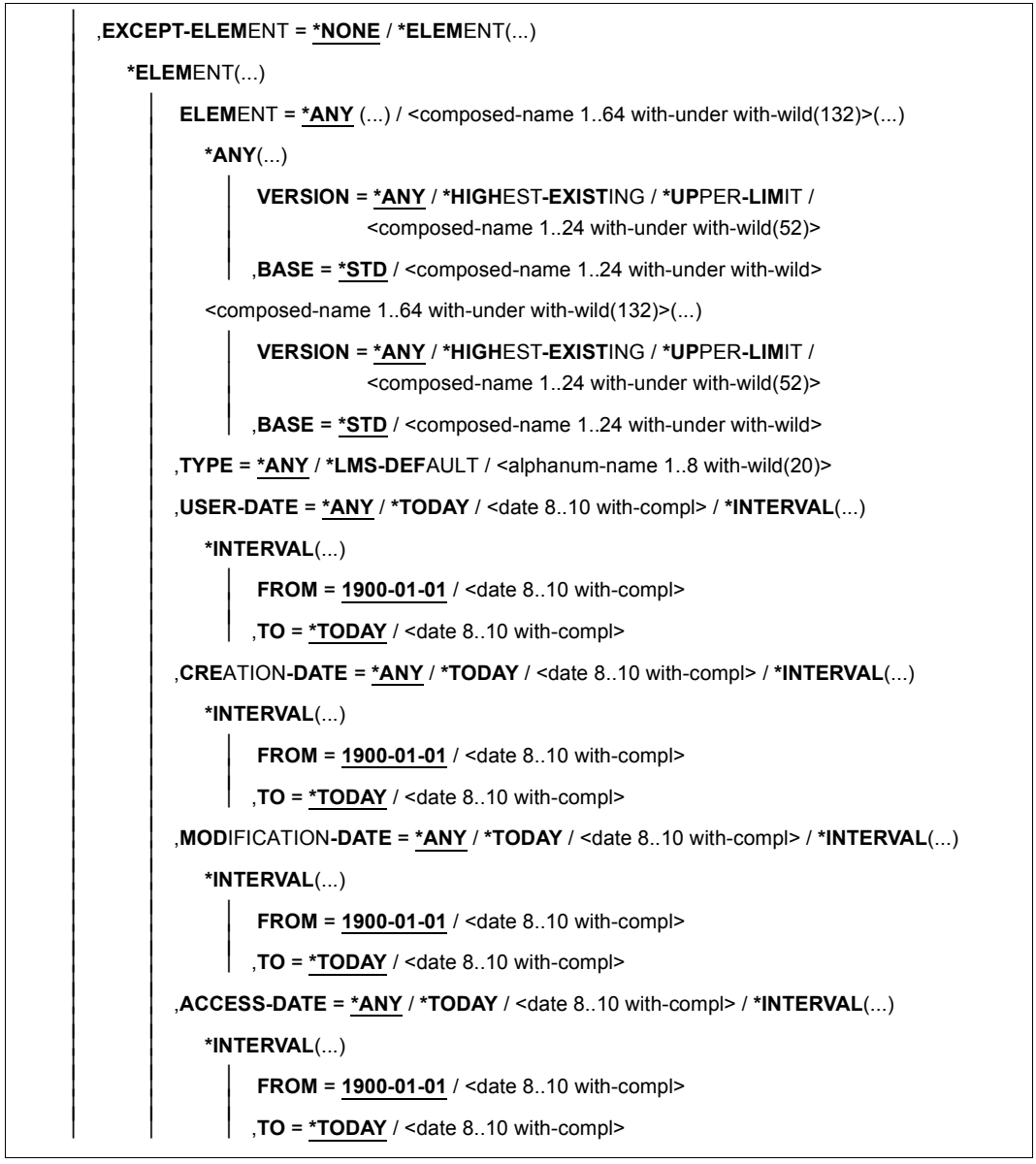
,**SEC**ONDARY**-NAME** = ***ANY** / <alphanum-name 1..32 with-wild(68)>

,**SEC**ONDARY**-ATTR**IBUTE = ***ANY** / ***CSECT** / ***ENTRY**

(part 2 of 5)


,ELEMENT-SIZE = *ANY / <integer 0..2147483647> / *INTERVAL(...)
*INTERVAL(...)
FROM = 0/ <integer 1..2147483647>
,TO = 2147483647 / <integer 1..2147483647>
,PROTECTION = *ANY / *NONE / *PARAMETERS(...)
*PARAMETERS(...)
READ = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
*PARAMETERS(...)
USER = *ANY / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *ANY / *YES / *NO
,WRITE = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
*PARAMETERS(...)
USER = *ANY / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *ANY / *YES / *NO
,EXEC = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
*PARAMETERS(...)
USER = *ANY / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *ANY / *YES / *NO
,HOLD = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
*BY-GUARD(...)
GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
*PARAMETERS(...)
USER = *ANY / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
,PASSWORD = *ANY / *YES / *NO

(part 3 of 5)


,EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
*ELEMENT(...)
ELEMENT = *ANY (...) / <composed-name 1..64 with-under with-wild(132)>(...)
*ANY(...)
VERSION = *ANY / *HIGHEST-EXISTING / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
,BASE = *STD / <composed-name 1..24 with-under with-wild>
<composed-name 1..64 with-under with-wild(132)>(...)
VERSION = *ANY / *HIGHEST-EXISTING / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
,BASE = *STD / <composed-name 1..24 with-under with-wild>
,TYPE = *ANY / *LMS-DEFAULT / <alphanum-name 1..8 with-wild(20)>
,USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>
,CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>
,MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>
,ACCESS-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
*INTERVAL(...)
FROM = 1900-01-01 / <date 8..10 with-compl>
,TO = *TODAY / <date 8..10 with-compl>

(part 4 of 5)

,**USER-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **00:00:00** / <time 1..8>
,**TO** = **23:59:59** / <time 1..8>
,**CRE**ATION-**TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **00:00:00** / <time 1..8>
,**TO** = **23:59:59** / <time 1..8>
,**MOD**IFICATION-**TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **00:00:00** / <time 1..8>
,**TO** = **23:59:59** / <time 1..8>
,**ACCESS-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
***INTERVAL**(...)
FROM = **00:00:00** / <time 1..8>
,**TO** = **23:59:59** / <time 1..8>

,**INF**ORMATION = ***LMS-DEF**AULT / ***MED**IUM / ***MIN**IMUM / ***MAX**IMUM / ***SUMM**ARY / ***DELTA-STRUCT**URE / ***PARA**METERS(...)

***PARA**METERS(...)
GENERAL = ***LMS-DEF**AULT / ***NO** / ***YES**
,HISTORY = ***LMS-DEF**AULT / ***NO** / ***YES**
,SECURITY = ***LMS-DEF**AULT / ***NO** / ***YES**

,**LAYOUT** = ***LMS-DEF**AULT / ***VAR**IABLE / ***FIXED**

,**SORT** = ***LMS-DEF**AULT / ***BY-NAME** / ***BY-VERS**ION / ***BY-USER-DATE** / ***BY-CREATION-DATE** / ***BY-MODIFICATION-DATE** / ***BY-ACCESS-DATE** / ***BY-ELEMENT-SIZE** / ***BY-SEC**ONDARY-**NAME**

,**TEXT-OUT**PUT = ***LOGGING-PAR**AMETERS / ***NONE** / ***SYSOUT** / ***SYSLST**(...) / ***EDT**(...)

***SYSLST**(...)
SYSLST-NUMBER = ***STD** / <INTEGER 1..99>

***EDT**(...)
WRITE-**MODE** = ***EXTEND** / ***REPLACE**

,**STRUC**TURE-**OUTPUT** = ***SYSINF** / ***NONE** / <composed-name 1..255>(...)

<composed-name 1..255>(...)
WRITE-**MODE** = ***REPLACE** / ***EXTEND**

(part 5 of 5)

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation(s).

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library for which the directory is to be output.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Outputs the directory of the library specified here.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library, which was defined by means of a /ADD-FILE-LINK command.

ELEMENT = *ALL / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member for which the library entry is to be output.
If the default value "*ALL" is entered, LMS outputs the library entries for all of the members with the corresponding version and type.

VERSION = *ALL / *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(52)>
Version of the member.

VERSION = *ALL
Outputs the library entries of all members selected above, regardless of their respective versions.

VERSION = *HIGHEST-EXISTING
The library entries of all members selected above are output with the highest existing version.

VERSION = *UPPER-LIMIT
The library entries of all members selected above are output with the version X'FF'.

VERSION = <composed-name 1..24 with-under with-wild(52)>
The library entries of all members selected above are output with the version specified here.

BASE = *STD / <composed-name 1..24 with-under with-wild>
Prefix for the version selection. In conjunction with VERSION=***HIGHEST-EXISTING**, it is then possible to use a certain prefix to reference the highest existing version. BASE=*STD has the same effect as BASE=*.
The operand will be ignored, if anything other than *ANY is specified in the SECONDARY-NAME or the SECONDARY-ATTRIBUTE operands.

TYPE = *LMS-DEFAULTS / *ALL / <alphanum-name 1..8 with-wild(20)
Type of the member which is to be output.
If the LMS default value for TYPE is *UNDEFINED, then *LMS-DEFAULT has the same effect as *ALL

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member has any date.

USER-DATE = *TODAY
The library entries of all members with the current date are output.

USER-DATE = <date 8..10 with-compl>
The library entries of all members with the date specified here in the form [YY]YY-MM-DD are output.

USER-DATE = *INTERVAL(...)
The library entries of all members lying in the specified interval are output.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see USER-DATE.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see USER-DATE.

ACCESS-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date on which the member was last accessed. For a description of the operands, see USER-DATE.

USER-TIME = *ANY / <time 1..8> / *INTERVAL(...)
Time given by the user.

USER-TIME = *ANY
The library entries of all members are output, regardless of the time.

USER-TIME = <time 1..8>
The library entries of all members with the time specified in the form HH:MM:SS are output.

USER-TIME = *INTERVAL(...)
The library entries of all members lying in the specified interval are output.

FROM = 00:00:00 / <time 1..8>
Beginning of interval.

TO = 23:59:59 / <time 1..8>
End of interval.

CREATION-TIME = *ANY / <time1..8> / *INTERVAL(...)
Creation time of the member. For a description of the operands, see USER-TIME, page 413.

MODIFICATION-TIME = *ANY / <time 1..8> / *INTERVAL(...)
Time of the last modification to the member. For a description of the operands, see USER-TIME, page 413.

ACCESS-TIME = *ANY / <time 1..8> / *INTERVAL(...)
Time at which the member was last accessed. For a description of the operands, see USER-TIME, page 413.

CODED-CHARACTER-SET = *ANY / *NONE / <name 1..8 with-wild(20)>
Character set assigned to the member.

CODED-CHARACTER-SET = *ANY
Selects members without regard to their assigned character set.

CODED-CHARACTER-SET = *NONE
Selects members which have not been assigned a character set.

CODED-CHARACTER-SET = <name 1..8 with-wild(20)>
Selects the members to which the specified character set has been assigned.

STATE = *ANY / *FREE / *IN-HOLD(...)
State assigned to the member.

STATE = *ANY
Selects members without regard to their respective STATEs.

STATE = *FREE
Selects only members with STATE=FREE

STATE = *IN-HOLD(...)
Selects only members with STATE=IN-HOLD.

HOLDER = *ANY / <name 1..8 with-wild(20)>
HOLDER assigned to the member.

HOLDER = *ANY
Selects members without regard to their respective HOLDERs.

HOLDER = <name 1..8 with-wild(20)>
Selects only members which are assigned a HOLDER matching the pattern.

STORAGE-FORM = *ANY / *FULL / *DELTA
Storage form for the member to be displayed.

STORAGE-FORM = *ANY
LMS selects the member on the basis of the storage form. The member may be stored as a non-delta or delta member.

STORAGE-FORM = *FULL
The member must be stored as a non-delta member.

STORAGE-FORM = *DELTA
The member must be stored as a delta member.

SECONDARY-NAME = *ANY / <alphanum-name 1..32 with-wild(68)>
Secondary name. If anything other than *ANY is specified here, the selection is made via the secondary directory of the library.
If wildcards are specified, only the first 32 characters of the secondary name are used to determine the selection.

SECONDARY-ATTRIBUTE = *ANY / *CSECT / *ENTRY
Secondary attribute. If anything other than *ANY is specified here, the selection is made via the secondary directory of the library.

ELEMENT-SIZE = *ANY / <integer 0..2147483647> / *INTERVAL(...)
Specifies the size according to which the members are to be selected.

ELEMENT-SIZE = *ANY
The member size is not used as a selection criterion.

ELEMENT-SIZE = <integer 0..2147483647>
Members are selected whose memory allocation corresponds to the specified number of PAM pages (2-K unit).

ELEMENT-SIZE = *INTERVAL(..)
Members are selected whose memory allocation lies within the specified range.

FROM = 0 / <integer 1..2147483647>
Members are selected whose memory allocation corresponds to at least the specified number of PAM pages (2-K unit).

TO = 2147483647 / <integer 1..2147483647>
Members are selected whose memory allocation corresponds to no more than the specified number of PAM pages (2-K unit).

PROTECTION = *ANY / *NONE / *PARAMETERS(...)
Member protection for the selected members.

PROTECTION = *ANY
The members are selected, regardless of the member protection.

PROTECTION = *NONE
The members have no additional member protection.

PROTECTION = *PARAMETERS(...)
Specifies the protection with which the members to be selected are provided.

READ = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Read protection setting assigned to the member.

READ = *ANY
Selects members without regard to their read protection settings.

READ = *NONE
Selects only members with no read protection.

READ = *BY-GUARD(...)
Selects only members which have GUARD read protection.

GUARD-NAME = <filename 1..40 without-cat-gen-vers with-wild>
Selects only members which have read protection by a GUARD-NAME matching the pattern.

READ = *PARAMETERS(...)
Selects only members which have read protection by BACL and/or password.

USER = *ANY / *NONE / *ALL / list-poss(3): *OWNER / *GROUP / *OTHERS
The read-authorized user group which is assigned to the member.

USER = *ANY
Selects members without regard to their respective read-authorized user groups.

USER = *NONE
Selects only members for which no read authorization has been granted.

USER = *OWNER
Selects only members for which the owner of the library file has read authorization.

USER = *GROUP
Selects only members for which the library file owner’s group has read authorization.

USER = *OTHERS
Selects only members for which read authorization for OTHERS has been granted.

PASSWORD = *ANY / *YES / *NO
Read password assigned to the member.

PASSWORD = *ANY
Selects members without regard to their respective passwords.

PASSWORD = YES
Selects only members which are protected by a read password.

PASSWORD = *NO
Selects only members which are not protected by a read password.

WRITE = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Write authorization. Selects only members for which this authorization has been granted in the specified manner.
The operands are analogous to those described above for READ.

EXEC = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Execute authorization. Selects only members for which this authorization has been granted in the specified manner.
The operands are analogous to those described above for READ.

HOLD = *ANY / *NONE / *BY-GUARD(...) / *PARAMETERS(...)
Hold authorization. Selects only members for which this authorization has been granted in the specified manner.
The operands are analogous to those described above for READ.

EXCEPT-ELEMENT = *NONE / *ELEMENT(...)
Specifies the members to be excluded from the above selection.

EXCEPT-ELEMENT = *NONE
No members are excluded.

EXCEPT-ELEMENT = *ELEMENT(...)
Specifies the members to be excluded from selection. A member is excluded when all the fields of the EXCEPT-ELEMENT structure that are not set to *ANY identify the member as a hit. If all the fields of the EXCEPT-ELEMENT structure are set to *ANY, then all members will be excluded.
For a description of the operands, see *LIBRARY-ELEMENT on page 412.

INFORMATION = *LMS-DEFAULT / *MEDIUM / *MINIMUM / *MAXIMUM / *SUMMARY / *DELTA-STRUCTURE / *PARAMETERS(...)
This parameter defines the scope of directory information to be output. It also specifies the scope of the structure output (see parameter dependencies on page 422ff.

INFORMATION = *MEDIUM
Outputs the type, name, version, variant number and, depending on the SORT operand, the date or size of the selected member.

INFORMATION = *MINIMUM
Outputs only the type, name and version of the selected member.

INFORMATION = *MAXIMUM
All the information of the selected member is output.

INFORMATION = *SUMMARY
Outputs only the number of selected members per type.

INFORMATION = *DELTA-STRUCTURE
Only the relationship “predecessor - successor” is output for delta members. As well as the member designation the internal delta number (DELTA#, reflects the chronological order) and the associated base number (BASE#) are output. These internal delta numbers are unique within a tree, they define the chaining of the members in the tree (independent of the external user-specific version designation). The output of a tree is always sorted by DELTA#, i.e. the SORT operand is not effective within a tree. Different trees are separated from each other by means of a continuous line.
The output fields DELTA# and BASE# for non-delta members are empty.

INFORMATION = *PARAMETERS(...)
Additional outputs

GENERAL = *LMS-DEFAULT / *NO / *YES
Defines whether, in addition to the type, name, version and variant number, a general information block is to be output comprising the storage format, status, member size and, if applicable, the character set and HOLDER of the selected member.

HISTORY = *LMS-DEFAULT / *NO / *YES
Defines whether, in addition to the type, name, version and variant number, a HISTORY block is to be output comprising the user date/time, creation date/time, modification date/time and, if applicable, the access date/time of the selected member.

SECURITY = *LMS-DEFAULT / *NO / *YES
Defines whether, in addition to the type, name, version and variant number, a SECURITY block for the selected member is to be output, if additional member protection has been granted for an access or borrowing right.

LAYOUT = *LMS-DEFAULT / *VARIABLE / *FIXED
This parameter defines the format of the directory to be output.

LAYOUT = *VARIABLE
The number of print columns is determined by the longest member designation within a member type. With output to the screen and no special sorting, the layout is oriented to the longest member designation in the output buffer, where subsequent outputs within a member type only change if longer member designations occur.

LAYOUT = *FIXED
The directory is printed in a single column in fixed format. Single column means that the entries in the directory appear one beneath the other.

SORT = *LMS-DEFAULT / *BY-NAME / *BY-VERSION / *BY-USER-DATE / *BY-CREATION-DATE / *BY-MODIFICATION-DATE / *BY-ACCESS-DATE / *BY-ELEMENT-SIZE / *BY-SECONDARY-NAME
Sort criterion for the directory entries of the selected members. The type is always used as the first sort criterion.

SORT = *BY-NAME
The directory entries of the selected members are sorted on the basis of the following sequence: type, name and version.

SORT = *BY-VERSION
The directory entries of the selected members are sorted on the basis of the following sequence: type, version and name.

SORT = *BY-USER-DATE
The directory entries of the selected members are sorted on the basis of the following sequence: type, user date, name and version.

SORT = *BY-CREATION-DATE
The directory entries of the selected members are sorted on the basis of the following sequence: type, creation date, name and version.

SORT = *BY-MODIFICATION-DATE
The directory entries of the selected members are sorted on the basis of the following sequence: type, modification date, name and version.

SORT = *BY-ACCESS-DATE
The directory entries of the selected members are sorted on the basis of the following sequence: type, access date, name and version.

SORT = *BY-ELEMENT-SIZE
The directory entries of the selected members are sorted on the basis of the following sequence: type, member size, name and version.

SORT = *BY-SECONDARY-NAME
The directory entries of the selected members are sorted on the basis of the following sequence: type, secondary name, secondary attribute, name and version.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified with MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=.

TEXT-OUTPUT = *NONE
The log output is suppressed, apart from error messages.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
Output is to the work file 9 of EDT. If an error occurs during log output, then the system switches to the default output stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output will be added to this data. If there is no data in the file, the output will be written at the beginning of the file.

WRITE-MODE = *REPLACE
The output will be written at the beginning of work file 9. Any data that is already in the file will be replaced.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Structured output.

STRUCTURE-OUTPUT = *SYSINF
The structured output is placed in the SYSINF stream assigned by means of ASSIGN-STREAM (see the [12]).

STRUCTURE-OUTPUT = *NONE
There is no structured output.

STRUCTURE-OUTPUT = <composed-name 1..255>(...)
Specifies the S variable in which the structured output is to be placed. This variable must be declared as a dynamic list variable.
(Command: `DECLARE-VARIABLE NAME=...(TYPE=*STRUCTURE),MULTIPLE-ELEMENTS=*LIST`)

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the structured output is to be appended to any existing data in the list variable or it to be written over it.

WRITE-MODE = *REPLACE
Overwrites any existing data in the list variable.

WRITE-MODE = *EXTEND
Appends the new list members to the existing list, if any.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0302	Member not found
	64	LMS0303	Member outside reference condition range
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Notes

– If the SECONDARY-NAME or SECONDARY-ATTRIBUTE operand is specified and if the value of this operand is other than *ANY, then a header will also be included in the directory output, providing information on the secondary name and the secondary attribute. The secondary names, however, are not displayed at maximum length.

– In the case of delta members, the value output for ELEM-SIZE or the selection of the member size is based on the memory allocation for the whole delta tree. Other measurements of the size of individual member versions can be displayed by means of SHOW-ELEMENT ...,TEXT-INFO=*STATISTICS.

– If a library list is specified, INFORMATION=*DELTA-STRUCTURE is not permitted. For SECONDARY-NAME and SECONDARY-ATTRIBUTES, only the value *ANY is allowed.

Required access rights

Read authorization for LIBRARY.

Parameter dependencies

The following dependencies exist between the INFORMATION, SORT and LAYOUT operands:

– The current date or the member size only influences the sorting if it, too, is to be included in the directory output. In this case, the INFORMATION operand must have a setting other than *MINIMUM.

– The LAYOUT operand is only effective if INFORMATION = *MEDIUM/*MINIMUM is specified. With all other settings, the directory is always output in fixed format.

– If INFORMATION = *MAXIMUM is specified, the information for each member will be too long for a single line. The information will then be output in a format independent of layout control.

– Output of the directory is accelerated if the INFORMATION operand is set to *MINIMUM and the selection is restricted to the ELEMENT, VERSION and TYPE operands. All other operands should be set to the value *ANY.

– The SORT and INFORMATION operands (except for INFORMATION=*SUMMARY) influence the sort sequence of the directory. The following table shows these dependencies:

	INFORMATION	
SORT	***MINIMUM**	***MEDIUM / *MAXIMUM / *PAR**
*BY-NAME	1. Type 2. Name 3. Version	1. Type 2. Name 3. Version
*BY-VERSION	1. Type 2. Version 3. Name	1. Type 2. Version 3. Name
*BY-USER-DATE	as SORT = BY-NAME	1. Type 2. User date 3. Name 4. Version
*BY-CREATION-DATE	as SORT = BY-NAME	1. Type 2. Creation date 3. Name 4. Version
*BY-MODIFICATION-DATE	as SORT = BY-NAME	1. Type 2. Modification date 3. Name 4. Version

SORT	INFORMATION	
	***MINIMUM**	***MEDIUM / *MAXIMUM / *PAR**
*BY-ACCESS-DATE	as SORT = BY-NAME	1. Type 2. Access date 3. Name 4. Version
*BY-ELEMENT-SIZE	as SORT = BY-NAME	1. Type 2. Member size 3. Name 4. Version
*BY-SECONDARY-NAME - with reference [1])	1. Type 2. Secondary name 3. Secondary attribute 4. Name 5. Version	1. Type 2. Secondary name 3. Secondary attribute 4. Name 5. Version
- without reference [2])	1. Type 2. Name 3. Version	1. Type 2. Name 3. Version

[1]) with reference means that in member selection either the secondary name or the secondary attribute was specified with a value other than the default ANY:
SECONDARY-NAME = <alphanum-name...>
and/or SECONDARY-ATTRIBUTE = *CSECT or *ENTRY

[2]) without reference means that in member selection neither a secondary name nor a secondary attribute was specified.

When generating structured output (STRUCTURE-OUTPUT =<composed-name 1..255>), bear in mind the SORT and INFORMATION operands. The following dependencies exist:

– The sort order is identical to that used for text output, and the INFORMATION operand has the same effect on the sort order as it does in text output.

– One list member is generated for each directory entry to be output.

– The INFORMATION operand influences the scope of information output in the S variable.

– The individual variable members are described in chapter “Format of LMS output in S variables” on page 445.

– If INFORMATION=*DELTA-STRUCTURE is specified, no structured output is generated.

Example

Outputting the directory for the library USER.BSPLIB. The library contains precisely one member which is displayed with all its attributes.

```
//SHOW-ELEMENT-ATTRIBUTES -
 (LIBRARY=USER.BSPLIB,ELEMENT=*(VERSION=*)),INFORMATION=*MAXIMUM
INPUT  LIBRARY= :N:$USER.USER.BSPLIB
TYPE       = D
NAME       = TEST
VERSION    = @                                          VARIANT   = 0001
--------------------------------GENERAL-----------------------------------
ELEM-SIZE  = 12
STORAGEW   = *FULL
STATE      = *IN-HOLD     HOLDER     = MUBF
-------------------------------HISTORY------------------------------------
USER-DATE  = 2012-08-12   CRE-DATE   = 2012-09-12   MOD-DATE  = 2012-08-12
USER-TIME  =   11:55:36   CRE-TIME   =   11:55:36   MOD-TIME  =   11:55:36
                                                    ACC-DATE  = 2012-08-12
                                                    ACC-TIME  =   11:55:36
-------------------------------SECURITY-----------------------------------
READ-PASS  = *NONE        READ-USER  = *OWNER *GROUP   -
WR-PASS    = *NONE        WR-PASS    = *OWNER    -     -
```

SHOW-LIBRARY-ATTRIBUTES

This statement displays all attributes set for the library. These are as follows:

– the attributes set by means of MODIFY-LIBRARY-ATTRIBUTES
– library size in 2-K units
– number of 2-K units available (can be removed by copying)
– library format (NK2/NK4)
– UPAM protection (Y/N)

```
SHOW-LIBRARY-ATTRIBUTES

 LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
    *LINK(...)
      | LINK-NAME = <structured-name 1..8>
,TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
    *SYSLST(...)
      | SYSLST-NUMBER = *STD / <INTEGER 1..99>
    *EDT(...)
      | WRITE-MODE = *EXTEND / *REPLACE
,STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
    <composed-name 1..255>(...)
      | WRITE-MODE = *REPLACE / *EXTEND
```

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
The library whose attributes are to be displayed.

LIBRARY = *STD
The global library opened by OPEN-LIBRARY is displayed.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose status is to be displayed.

LIBRARY = *LINK(...)
The status of the library assigned via a link name is displayed.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified with MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=.

TEXT-OUTPUT = *NONE
The log output is suppressed, apart from error messages.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
Output is to the work file 9 of EDT. If an error occurs during log output, then the system switches to the default output stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output will be added to this data. If there is no data in the file, the output will be written at the beginning of the file.

WRITE-MODE = *REPLACE
The output will be written at the beginning of work file 9. Any data that is already in the file will be replaced.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Structured output.

STRUCTURE-OUTPUT = *SYSINF
The structured output is placed in the SYSINF stream assigned by means of ASSIGN-STREAM (see the “SDF-P“ manual [12]).

STRUCTURE-OUTPUT = *NONE
There is no structured output.

STRUCTURE-OUTPUT =<composed-name 1..255>(...)
Specifies the S variable in which the structured output is to be placed. This variable must be declared as a dynamic list variable.
(Command: `DECLARE-VARIABLE NAME =...(TYPE=*STRUCTURE), MULTIPLE-ELEMENTS=*LIST`)

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the structured output is to be appended to any existing data in the list variable or it to be written over it.

WRITE-MODE = *REPLACE
Overwrites any existing data in the list variable.

WRITE-MODE = *EXTEND
Appends the new list members to the existing list, if any.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Note

The individual variable members are described in chapter “Format of LMS output in S variables” on page 445.

Required access rights

Read authorization for LIBRARY.

Example

```
//SHOW-LIBRARY-ATTRIBUTES LIB=BSPLIB
INPUT  LIBRARY= :1OSQ:$USER.BSPLIB
READ-PASS  = *NONE       READ-USER  = *OWNER   -     -
WR-PASS    = *YES        WR-USER    = *OWNER   -     -
EXEC-PASS  = *NONE       EXEC-USER  = *OWNER   -     -
HOLD-PASS  = *NONE       HOLD-USER  = *OWNER   -     -
ADMIN-PASS = *NONE       ADMIN-USER = *OWNER   -     -
FILE-SIZE  = 291         FREE-SIZE  = 62       FORMAT = NK2  UPAM-PROT = N
ACCESS-DATE= *KEEP       WR-CONTRPL = *NONE    STORAGE= *STD
```

SHOW-LIBRARY-STATUS

This statement displays the status of the libraries used.

LMS outputs the following information on execution of the statement:

– name of the library/libraries
– status of the library/libraries (opened or closed)
– assigned link name, if any
– assigned library-specific default type, if any

SHOW-LIBRARY-**STA**TUS
LIBRARY = ***ALL** / ***STD** / <filename 1..54 without-vers> / ***LINK**(...) ***LINK**(...) **LINK**-NAME = <structured-name 1..8>

LIBRARY = *ALL / *STD / <filename 1..54 without-vers> / *LINK(...)
The library whose status is to be displayed.

LIBRARY = *ALL
All libraries used are displayed.

LIBRARY = *STD
The global library opened by OPEN-LIBRARY is displayed.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose status is to be displayed.

LIBRARY = *LINK(...)
The status of the library assigned via a link name is displayed.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Example

Five different libraries were used during the LMS run. One library was addressed via the link name LIB1:

```
//SHOW-LIBRARY-STATUS
STATUS FILENAME                                          MODE    LINK  DEF-TYPE
OPEN   :N:$USER.LMSPL.LIB                                UPDATE
CLOSED :N:$USER.MODUL.LIB                                        LIB1
CLOSED :N:$USER.MACRO.LIB                                              M
CLOSED :N:$USER.QUELL.LIB
CLOSED :N:$USER.TEST.LIB
```

SHOW-LMS-DEFAULTS

This statement outputs the current values of the LMS defaults. These can be modified by means of the MODIFY-LMS-DEFAULTS statement.

SHOW-LMS-DEFAULTS
DEFAULTS = **<u>*STD</u>** / ***ALL** / list-poss(2000): ***ELEM**ENT-**ATTR**IBUTES / ***F**ILE-**ATTR**IBUTES / ***DESTROY-DATA** / ***WR**ITE-**MODE** / ***DIALOG-CONTR**OL / ***INF**ORMATION / ***LAYOUT** / ***SORT** / ***OUT**PUT-**FORM** / ***DEL**ETE-**SOUR**CE / ***PROT**ECTION / ***MAX-ERROR-WEIGHT** / ***EDT-MODE** / ***NEXT-ATTEM**PT / ***COMP**ARE-**PAR**AMETERS / ***TEXT-INF**ORMATION / ***MOD**ULE-**INF**ORMATION / ***PHASE-INF**ORMATION / ***LLM-INF**ORMATION

DEFAULTS = <u>*STD</u>
Outputs the default values for the following with their current settings: ELEMENT-ATTRIBUTES, FILE-ATTRIBUTES, DESTROY-DATA, WRITE-MODE, DIALOG-CONTROL, INFORMATION, LAYOUT, SORT, OUTPUT-FORM, DELETE-SOURCE and PROTECTION.

DEFAULTS = <u>*ALL</u>
Outputs all defaults with their current values.

DEFAULTS = *ELEMENT-ATTRIBUTES
Outputs the current values for member type, source and target version, storage form and the file attributes.

DEFAULTS = *FILE-ATTRIBUTES
The current value for the file access method is output.

DEFAULTS = *DESTROY-DATA
Whether or not data is to be overwritten is output.

DEFAULTS = *WRITE-MODE
The current value for the write mode is output.

DEFAULTS = *DIALOG-CONTROL
The current value for the dialog control is output.

DEFAULTS = *INFORMATION
Outputs the current setting for the scope of directory information to be output.

DEFAULTS = *LAYOUT
Outputs the current setting for the layout of the directory to be output.

DEFAULTS = *SORT
Outputs the current setting for the sort criterion of the directory to be output.

DEFAULTS = *OUTPUT-FORM
Outputs the current setting for the output form.

DEFAULTS = *DELETE-SOURCE
Outputs the default setting for source file deletion, i.e. whether the source file is to be deleted or kept.

DEFAULTS = *PROTECTION
Outputs the default setting for the adoption of protection attributes.

DEFAULTS = *MAX-ERROR-WAIT
Outputs the default setting for spin-off control.

DEFAULTS = *EDT-MODE
Outputs the default setting for the mode that EDT is to be called in.

DEFAULTS = *NEXT-ATTEMPT
Outputs the default setting for the control of attempts to open files, etc.

DEFAULTS = *COMPARE-PARAMETERS
Outputs the default setting for the comparison parameters.

DEFAULTS = *TEXT-INFORMATION
Outputs the default setting for the scope of information for textual members.

DEFAULTS = *MODULE-INFORMATION
Outputs the default setting for the scope of information for object modules.

DEFAULTS = *PHASE-INFORMATION
Outputs the default setting for the scope of information for phases.

DEFAULTS = *LLM-INFORMATION
Outputs the default setting for the scope of information for link and load modules.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Required access rights

No access rights are necessary.

Example

```
//SHOW-LMS-DEFAULTS  *ALL
ELEMENT-ATTRIBUTES
     TYPE               = *NONE
     ELEMENT-VERSION    = *ALL
     TO-ELEM-VERSION    = *BY-SOURCE
     STORAGE-FORM       = *STD
     SOURCE-ATTRIBUTES  = *STD
FILE-ATTRIBUTES
     ACCESS-METHOD      = *ISAM
DESTROY-DATA            = *NO
WRITE-MODE              = *CREATE
DIALOG-CONTROL          = *NO
INFORMATION             = *MEDIUM
LAYOUT                  = *VARIABLE
SORT                    = *BY-NAME
OUTPUT-FORM             = *STD
DELETE-SOURCE           = *NO
PROTECTION              = *STD
MAX-ERROR-WEIGHT        = *SERIOUS
EDT-MODE                = *COMPATIBLE
NEXT-ATTEMPT            = *NO
COMPARE-PARAMETERS
     RECORD-PART        = *ALL
     SPACES             = *STD
     INFORMATION        = *MEDIUM
     LAYOUT             = *COMPATIBLE
     JOIN-ELEMENT-SETS  = *NO
TEXT-INFORMATION        = *ALL
MODULE-INFORMATION      = *ALL
PHASE-INFORMATION       = *ALL
LLM-INFORMATION         = *ALL
```

All LMS default values are output. These values are applicable immediately after the start of LMS.

SHOW-LOGGING-PARAMETERS

This statement outputs the global LMS option values currently in force. These values are modified by means of the MODIFY-LOGGING-PARAMETERS statement.

If this statement is specified without operands, the presettings for all of the parameters are output (see example).

SHOW-LOGGING-PARAMETERS
PARAMETERS = ***ALL** / list-poss(2000): ***LOG**GING / ***TEXT-OUTPUT** / ***OUTPUT-LAYOUT**

PARAMETERS = *ALL / list-poss(2000): *LOGGING / *TEXT-OUTPUT / *OUTPUT-LAYOUT
Specifies the parameters.

PARAMETERS = *ALL
The current settings of all the parameters are output.

PARAMETERS = *LOGGING
The setting for the scope of LMS logging is output.

PARAMETERS = *TEXT-OUTPUT
The output medium setting is output.

PARAMETERS = *OUTPUT-LAYOUT
The parameter settings for the LMS log format are output.

Required access rights

No access rights are necessary.

Example

```
//SHOW-LOGGING-PARAMETERS
LOGGING                 = *MINIMUM
TEXT-OUTPUT             = *SYSOUT
OUTPUT-LAYOUT
     LINES-PER-PAGE     = 60
     LINE-SIZE          = 132
     EXTRA-FORM-FEED    = *NO
     HEADER-LINES       = *YES
```

All global LMS options are output. These values are applicable immediately after the start of LMS.

SHOW-STATISTICS

Comparison statistics are generated when comparing members with COMPARE-ELEMENT. COMPARE-ELEMENT stores these comparison statistics in an internal memory area C0. After execution of the COMPARE-ELEMENT statement, C0 is added to area C1. C0 is reinitialized before COMPARE-ELEMENT is executed again.

The SHOW-STATISTICS statement outputs these comparison statistics. To permit this, the number of the area must be specified.

```
SHOW-STATISTICS

 NUMBER = *C0 / *C1

,HEADER-LINE = *NONE / <c-string 1..132>

,TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)

    *SYSLST(...)
        | SYSLST-NUMBER = *STD / <INTEGER 1..99>

    *EDT(...)
        | WRITE-MODE = *EXTEND / *REPLACE

,STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)

    <composed-name 1..255>(...)
        | WRITE-MODE = *REPLACE / *EXTEND
```

NUMBER = *C0 / *C1
Number of the area to be output:

*C0: area containing the statistics for the current comparison.

*C1: area containing the overall statistics for all the comparisons performed thus far in this LMS run.

HEADER-LINE = *NONE / <c-string 1..132>
It is possible to control whether or not a user-specific header (<c-string>) is output. By default no user-specific header is output.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified with MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=.

TEXT-OUTPUT = *NONE
The log output is suppressed, apart from error messages.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
Output is to the work file 9 of EDT. If an error occurs during log output, then the system switches to the default output stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output will be added to this data. If there is no data in the file, the output will be written at the beginning of the file.

WRITE-MODE = *REPLACE
The output will be written at the beginning of work file 9. Any data that is already in the file will be replaced.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Structured output.

STRUCTURE-OUTPUT = *SYSINF
The structured output is placed in the SYSINF stream assigned by means of ASSIGN-STREAM (see the [12]).

STRUCTURE-OUTPUT = *NONE
There is no structured output.

STRUCTURE-OUTPUT = <composed-name 1..255>(...)
Specifies the S variable in which the structured output is to be placed. This variable must be declared as a dynamic list variable.
(Command: `DECLARE-VARIABLE NAME =...(TYPE=*STRUCTURE), MULTIPLE-ELEMENTS=*LIST`)

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the structured output is to be appended to any existing data in the list variable or it to be written over it.

WRITE-MODE = *REPLACE
Overwrites any existing data in the list variable.

WRITE-MODE = *EXTEND
Appends the new list members to the existing list, if any.

The individual variable members are described in chapter “Format of LMS output in S variables” on page 445.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
2	0	LMS0313	Overflow in statistic counter
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Required access rights

No access rights are necessary.

Notes

The maximum value for element-count fields is 99,999. For line-count fields, it is 999,999,999.

On SYSOUT, LMS shows up to 8 digits if the value is less than 100 millions, otherwise it shows the value in exponential notation, e.g. 12345E+4. The exact values can be obtained from the structured output into an S variable or from text output to SYSLST, EDT or a library member.

If the 9-digit limit overflows, the message LMS0313 will be shown and the affected counters will continue counting modulo 10^9.

Example

The S variable C1 was declared as follows:

```
/DECL-VARIABLE C1(TYPE=*STRUCTURE),MULTIPLE-ELEMENTS=*LIST
```

Two comparisons have been performed. Memory C0 contains the result of the last comparison, memory C1 contains the totals for the comparisons performed thus far.

```
//SHOW-STATISTICS NUMBER=*C0
AREA C0
           PRIM.   PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.  SEC
STATISTIC  ELEM.   LINES    LINES    LINES    LINES    LINES   LINES ELEM.
S (SAME)      0       0        -        0        -        -       0     0
C (CHANGED)   1       5        5        0        9       14       9     1
I (INSERTED)  0       0        0        -        -        0       -     -
D (DELETED)   -       -        -        -        0        0       0     0
-------------------------------------------------------------------------
   TOTAL      1       5        5        0        9       14       9     1
//SHOW-STATISTICS NUMBER=*C1, STRUCTURE-OUTPUT=C1, TEXT-OUTPUT=*NONE
```

Structure C1 can be viewed with `/SHOW-VARIABLE C1#1`:

```
C1(*LIST).SAME.NUM-OF-PRIMARY = 5
C1(*LIST).SAME.NUM-OF-SECONDARY = 5
C1(*LIST).SAME.LINE.PRIMARY = 37
C1(*LIST).SAME.LINE.INS = 0
C1(*LIST).SAME.LINE.SAME = 37
C1(*LIST).SAME.LINE.DEL = 0
C1(*LIST).SAME.LINE.SECONDARY = 37
C1(*LIST).CHA.NUM-OF-PRIMARY = 0
C1(*LIST).CHA. .....
           .....
C1(*LIST).DEL.LINE.SECONDARY = 0
C1(*LIST).TOTAL.NUM-OF.PRIMARY = 7
C1(*LIST).TOTAL.NUM-OF.SECONDARY = 5
C1(*LIST).TOTAL.LINE.PRIMARY = 47
C1(*LIST).TOTAL.LINE.INS = 10
C1(*LIST).TOTAL.LINE.SAME = 37
C1(*LIST).TOTAL.LINE.DEL = 0
C1(*LIST).TOTAL.LINE.SECONDARY = 37
```

SHOW-TYPE-ATTRIBUTES

This statement outputs all the attributes set for a given member type.

```
SHOW-TYPE-ATTRIBUTES

 LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
    *LINK(...)
      | LINK-NAME = <structured-name 1..8>
,TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
,TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
    *SYSLST(...)
      | SYSLST-NUMBER = *STD / <INTEGER 1..99>
    *EDT(...)
      | WRITE-MODE = *EXTEND / *REPLACE
,STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
    <composed-name 1..255>(...)
      | WRITE-MODE = *REPLACE / *EXTEND
```

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the type whose attributes are to be displayed.

LIBRARY = *STD
The library opened globally is used.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose type attributes are to be displayed.

LIBRARY = *LINK(...)
The type attributes of a library assigned via a link name are displayed.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of a /ADD-FILE-LINK command prior to calling LMS.

TYPE = *LMS-DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Member type whose attributes are to be output. If the LMS default value is set to *UNDEFINED, *LMS-DEFAULT has the same effect as *ALL.

TEXT-OUTPUT = *LOGGING-PARAMETERS / *NONE / *SYSOUT / *SYSLST(...) / *EDT(...)
Controls the log output.

TEXT-OUTPUT = *LOGGING-PARAMETERS
The log is output to the output medium specified with MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=.

TEXT-OUTPUT = *NONE
The log output is suppressed, apart from error messages.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Determines the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *EDT(...)
Output is to the work file 9 of EDT. If an error occurs during log output, then the system switches to the default output stream (SYSOUT).

WRITE-MODE = *EXTEND / *REPLACE
Write mode of the output in relation to the contents of work file 9.

WRITE-MODE = *EXTEND
If data exists in work file 9, the output will be added to this data. If there is no data in the file, the output will be written at the beginning of the file.

WRITE-MODE = *REPLACE
The output will be written at the beginning of work file 9. Any data that is already in the file will be replaced.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Structured output.

STRUCTURE-OUTPUT = *SYSINF
The structured output is placed in the SYSINF stream assigned by means of ASSIGN-STREAM (see the “SDF-P“ manual [12]).

STRUCTURE-OUTPUT = *NONE
There is no structured output.

STRUCTURE-OUTPUT = <composed-name 1..255>(...)
Specifies the S variable in which the structured output is to be placed. This variable must be declared as a dynamic list variable.
(Command: `DECLARE-VARIABLE NAME =...(TYPE=*STRUCTURE), MULTIPLE-ELEMENTS=*LIST`)

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the structured output is to be appended to any existing data in the list variable or it to be written over it.

WRITE-MODE = *REPLACE
Overwrites any existing data in the list variable.

WRITE-MODE = *EXTEND
Appends the new list members to the existing list, if any.

The individual variable members are described in chapter “Format of LMS output in S variables” on page 445).

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS0304	Type not found
	64	LMS1003	Error during wildcard processing with at least one member or file
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Required access rights

Read authorization for LIBRARY.

Notes

– If a certain type is specified, the values explicitly set using MODIFY-TYPE-ATTR, if any, will be output. Otherwise, the statement will output the implicit default values.

– If TYPE=* is specified, the statement outputs the attributes for all explicitly declared types and for types which are embodied by existing members.

Example

The type attributes for member type S are set and then displayed. The library used is the globally set library.

```
//MODIFY-TYPE-ATTRIBUTES TYPE=S,CONVENTION=*STD-SEQUENCE(EXAMPLE=V001)
//SHOW-TYPE-ATTRIBUTES TYPE=S
INPUT  LIBRARY= :1OSQ:$USER.LIB.MODTYATT
TYPE       = S
SUPER-TYPE = *NONE
CONVENTION = *STD-SEQUENCE
EXAMPLE    = V001
INIT-ELEM-P= *NONE
ADMINISTRAT= *NONE
STORAGE    = *STD          WR-CONTROL = *NONE
```

SHOW-USER-EXITS

This statement displays the active user exits.

SHOW-USER-EXITS

The SHOW-USER-EXITS statement has no operands.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Example

```
//SHOW-USER-EXITS
FUNCTION  ENTRY    LIBRARY                                          INT
SHOW      USELST   TEST.LIB                                         V1
```

WRITE-COMMENT

This statement is used to write comments to the output medium defined by the statement //MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=..., unlike the WRITE-TEXT statement, which always directs output to SYSOUT or SYSLST.

WRITE-**COMMENT**
COMMENT = **''** / <c-string 1..1024 with-low>

COMMENT = '' / <c-string 1..1024 with-low>
Comment text. If nothing is specified, a blank is used as the default value, i.e. a blank line is generated.

Statement return code

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	LMS0129	Statement aborted by user
	1	CMD0230	Syntax error
	32	LMS1002	Internal error
	64	LMS1004	Other error
	130	LMS0041	System address space exhausted

Example

The text enclosed in apostrophes, which may also extend over multiple lines, is interpreted as a comment.

```
//write-comment 'This is the last LMS statement to be described.-
//               A new chapter begins on the next page.'
```

8 Format of LMS output in S variables

If the structure members of a variable are invalid due to underlying conditions (see below), the members are created and filled with an empty string or 0.

In the tables below, the columns have the following meanings: **T** stands for data type (and the entries **S** and **I** for string and integer, respectively), **W** stands for information selection (only with the format for //SHOW-ELEMENT-ATTRIBUTES; see page 447) and **Con.** stands for condition.

8.1 COMPARE-ELEMENT statement

Output information	Name of S variable	T	Contents	Con.
Library name (first member)	varname#.PRIMARY.LIB	S	<filename 1..54>	
Member name (first member)	varname#.PRIMARY.ELEM	S	<comp-name 1..64>	
Member version (first member)	varname#.PRIMARY.VERSION	S	<comp-name 1..24> / ´*UP-LIM´	
Member type (first member)	varname#.PRIMARY.TYPE	S	<alphanum 1..8>	
Library name (second member)	varname#.SECONDARY.LIB	S	<comp-name 1..64>	
Member name (second member)	varname#.SECONDARY.ELEM	S	<comp-name 1..64>	
Member version (second member)	varname#.SECONDARY.VERSION	S	<comp-name 1..24> / ´*UP-LIM´	
Member type (second member)	varname#.SECONDARY.TYPE	S	<alphanum 1..8>	
Result of comparison	varname#.COMPARE-RESULT	S	´*SAME´ / ´*CHA´ / ´*DEL´ / ´*INS´	
Number of records in first member	varname#.LINE.PRIMARY	I	<integer>	
Number of inserted records	varname#.LINE.INS	I	<integer>	
Number of same records	varname#.LINE.SAME	I	<integer>	
Number of deleted records	varname#.LINE.DEL	I	<integer>	
Number of records in second member	varname#.LINE.SECONDARY	I	<integer>	

8.2 FIND-ELEMENT statement

Output information	Name of S variable	T	Contents	Con.
Library name	varname#.LIB	S	<filename 1..54>	
Member name	varname#.ELEM	S	<comp-name 1..64>	
Member version	varname#.VERSION	S	<comp-name 1..24> / ´*UP-LIM´	
Member type	varname#.TYPE	S	<alphanum 1..8>	
Number of records	varname#.LINE.ALL	I	<integer>	
Number of hit records	varname#.LINE.MATCH	I	<integer>	

8.3 SHOW-ELEMENT-ATTRIBUTES statement

Information selection	Selection#
INFORMATION = *MINIMUM / *MEDIUM	1
INFORMATION = *MAXIMUM	2
INFORMATION = *PARAMETERS (GENERAL=*YES)	3
INFORMATION = *PARAMETERS (HISTORY=*YES)	4
INFORMATION = *PARAMETERS (SECURITY=*YES)	5
INFORMATION = *SUMMARY	6

Output information	Name of S variable	T	Contents	W	Con.
Library name	varname#.LIB	S	<filename 1..54>	1-6	
Member name	varname#.ELEM	S	<comp-name 1..64>	1-5	
Member version	varname#.VERSION	S	<comp-name 1..24> / ´*UP-LIM´	1-5	
Member type	varname#.TYPE	S	<alphanum 1..8>	1-6	
Secondary name	varname#.SECONDARY-NAME	S	<alphanum 1..32>	1,2	*1
Secondary attribute	varname#.SECONDARY-ATTR	S	´*CSECT´ / ´*ENTRY´	1,2	*1
User date	varname#.USER-DATE	S	<yyyy-mm-dd>	2,4	
User time	varname#.USER-TIME	S	<hh:mm:ss>	2,4	
Date of member creation	varname#.CRE-DATE	S	<yyyy-mm-dd>	2,4	
Time of member creation	varname#.CRE-TIME	S	<hh:mm:ss>	2,4	
Date of last modification to member	varname#.MOD-DATE	S	<yyyy-mm-dd>	2,4	
Time of last modification to member	varname#.MOD-TIME	S	<hh:mm:ss>	2,4	
Date of last access to member	varname#.ACCESS-DATE	S	´*NONE´ / <yyyy-mm-dd>	2,4	
Time of last access to member	varname#.ACCESS-TIME	S	´*NONE´ / <hh:mm:ss>	2,4	
Storage form (full / delta storage)	varname#.STOR-FORM	S	´*FULL´ / ´*DELTA´	2,3	
Character set assigned to the member	varname#.CODED-CHAR-SET	S	´´ / <name 1..8>	2,3	
Status of the member	varname#.STA	S	´*FREE´ / ´*IN-HOLD´	2,3	
Holder assigned to the member	varname#.HOLDER	S	´´ / <name 1..8>	2,3	
Indicator, of whether member protection has been defined	varname#.PROT-DEFI	S	´*NONE´ / ´*PAR´	2,5	
Indicator for read protection	varname#.READ.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	2,5	*2
Name of read guard	varname#.READ.GUARD-NAME	S	<filename 1..18>	2,5	*3

Output information	Name of S variable	T	Contents	W	Con.
Indicator for read authorization (BACL)	varname#.READ.USER-DEFI	S	´*NONE´ / ´*LIST´	2,5	*4
Indicator for read authorization of owner	varname#.READ.OWNER	S	´´ / ´*OWNER´	2,5	*5
Indicator for read authorization of owner's group	varname#.READ.GROUP	S	´´ / ´*GROUP´	2,5	*5
Indicator for read authorization for OTHERS	varname#.READ.OTHERS	S	´´ / ´*OTHERS´	2,5	*5
Indicator of whether a read password has been defined for the member	varname#.READ.PASS	S	´*NO´ / ´*YES´	2,5	*4
Indicator for write protection	varname#.WRITE.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	2,5	*2
Name of write guard	varname#.WRITE.GUARD-NAME	S	<filename 1..18>	2,5	*3
Indicator for write authorization (BACL)	varname#.WRITE.USER-DEFI	S	´*NONE´ / ´*LIST´	2,5	*4
Indicator for write authorization of owner	varname#.WRITE.OWNER	S	´´ / ´*OWNER´	2,5	*5
Indicator for write authorization of owner's group	varname#.WRITE.GROUP	S	´´ / ´*GROUP´	2,5	*5
Indicator for write authorization for OTHERS	varname#.WRITE.OTHERS	S	´´ / ´*OTHERS´	2,5	*5
Indicator of whether a write password has been defined for the member	varname#.WRITE.PASS	S	´*NO´ / ´*YES´	2,5	*4
Indicator for execute protection	varname#.EXEC.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	2,5	*2
Name of execute guard	varname#.EXEC.GUARD-NAME	S	<filename 1..18>	2,5	*3
Indicator for execute authorization (BACL)	varname#.EXEC.USER-DEFI	S	´*NONE´ / ´*LIST´	2,5	*4
Indicator for execute authorization of owner	varname#.EXEC.OWNER	S	´´ / ´*OWNER´	2,5	*5
Indicator for execute authorization of owner's group	varname#.EXEC.GROUP	S	´´ / ´*GROUP´	2,5	*5
Indicator for execute authorization for OTHERS	varname#.EXEC.OTHERS	S	´´ / ´*OTHERS´	2,5	*5
Indicator of whether an execute password has been defined for the member	varname#.EXEC.PASS	S	´*NO´ / ´*YES´	2,5	*4
Indicator for hold protection	varname#.HOLD.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	2,5	*2
Name of hold guard	varname#.HOLD.GUARD-NAME	S	<filename 1..18>	2,5	*3
Indicator for hold authorization (BACL)	varname#.HOLD.USER-DEFI	S	´*NONE´ / ´*LIST´	2,5	*4
Indicator for hold authorization of owner	varname#.HOLD.OWNER	S	´´ / ´*OWNER´	2,5	*5
Indicator for hold authorization of owner's group	varname#.HOLD.GROUP	S	´´ / ´*GROUP´	2,5	*5

Output information	Name of S variable	T	Contents	W	Con.
Indicator for hold authorization for OTHERS	varname#.HOLD.OTHERS	S	´´ / ´*OTHERS´	2,5	*5
Indicator of whether an hold password has been defined for the member	varname#.HOLD.PASS	S	´*NO´ / ´*YES´	2,5	*4
Member size in 2K pages	varname#.ELEM-SIZE	I	<integer>	2,3	
Number of members for member type	varname#.NUM-OF-ELEM	I	<integer>	6	

*1 : SORT = *BY-SECONDARY-NAME and SECONDARY-NAME ≠ *ANY
*2 : varname#.PROT-DEFI = ´*PAR´
*3 : varname#.PROT-DEFI = ´*PAR´ and varname#.xxxx.DEFI = ´*BY-GUARD´
*4 : varname#.PROT-DEFI = ´*PAR´ and varname#.xxxx.DEFI = ´*PAR´
*5 : *4 and varname#.xxxx.USER-DEFI = ´*LIST´

8.4 SHOW-LIBRARY-ATTRIBUTES statement

Output information	Name of S variable	T	Contents	Con.
Library name	varname#.LIB	S	<filename 1..54>	
Permissible storage form	varname#.STOR-FORM	S	´*STD´ / ´*NONE´ / ´*FULL´ / ´*DELTA´	
Control of additional checks	varname#.WRITE-CONTR	S	´*NONE´ / ´*ACTIVATE´ ´*DEACTIVATE´	
Indicator for recording of access dates	varname#.ACCESS-DATE	S	´*NONE´ / ´*KEEP´	
Indicator for administration protection	varname#.ADM.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	
Name of administration guard	varname#.ADM.GUARD-NAME	S	<filename 1..18>	*3
Indicator for administer authorization (BACL)	varname#.ADM.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for administer authorization of owner	varname#.ADM.OWNER	S	´´ / ´*OWNER´	*5
Indicator for administer authorization of owner's group	varname#.ADM.GROUP	S	´´ / ´*GROUP´	*5
Indicator for administer authorization for OTHERS	varname#.ADM.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether an administration password has been defined for the member	varname#.ADM.PASS	S	´*NO´ / ´*YES´	*4
Indicator of whether member protection has been defined	varname#.PROT-DEFI	S	´*NONE´ / ´*PAR´	
Indicator for read protection	varname#.PROT.READ-DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of read guard	varname#.PROT.READ.GUARD-NAME	S	<filename 1..18>	*3
Indicator for read authorization (BACL)	varname#.PROT.READ.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for read authorization of owner	varname#.PROT.READ.OWNER	S	´´ / ´*OWNER´	*5
Indicator for read authorization of owner's group	varname#.PROT.READ.GROUP	S	´´ / ´*GROUP´	*5
Indicator for read authorization for OTHERS	varname#.PROT.READ.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether a read password has been defined for the member	varname#.PROT.READ.PASS	S	´*NO´ / ´*YES´	*4
Indicator for write protection	varname#.PROT.WRITE.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of write guard	varname#.PROT.WRITE.GUARD-NAME	S	<filename 1..18>	*3
Indicator for write authorization (BACL)	varname#.PROT.WRITE.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for write authorization of owner	varname#.PROT.WRITE.OWNER	S	´´ / ´*OWNER´	*5
Indicator for write authorization of owner's group	varname#.PROT.WRITE.GROUP	S	´´ / ´*GROUP´	*5
Indicator for write authorization for OTHERS	varname#.PROT.WRITE.OTHERS	S	´´ / ´*OTHERS´	*5

Output information	Name of S variable	T	Contents	Con.
Indicator of whether a write password has been defined for the member	varname#.PROT.WRITE.PASS	S	´*NO´ / ´*YES´	*4
Indicator for execute protection	varname#.PROT.EXEC.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of execute guard	varname#.PROT.EXEC.GUARD-NAME	S	<filename 1..18>	*3
Indicator for execute authorization (BACL)	varname#.PROT.EXEC.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for execute authorization of owner	varname#.PROT.EXEC.OWNER	S	´´ / ´*OWNER´	*5
Indicator for execute authorization of owner's group	varname#.PROT.EXEC.GROUP	S	´´ / ´*GROUP´	*5
Indicator for execute authorization for OTHERS	varname#.PROT.EXEC.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether an execute password has been defined for the member	varname#.PROT.EXEC.PASS	S	´*NO´ / ´*YES´	*4
Indicator for hold protection	varname#.PROT.HOLD.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of hold guard	varname#.PROT.HOLD.GUARD-NAME	S	<filename 1..18>	*3
Indicator for hold authorization (BACL)	varname#.PROT.HOLD.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for hold authorization of owner	varname#.PROT.HOLD.OWNER	S	´´ / ´*OWNER´	*5
Indicator for hold authorization of owner's group	varname#.PROT.HOLD.GROUP	S	´´ / ´*GROUP´	*5
Indicator for hold authorization for OTHERS	varname#.PROT.HOLD.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether a hold password has been defined for the member	varname#.PROT.HOLD.PASS	S	´*NO´ / ´*YES´	*4
Format of the library	varname#.LIB-FORM	S	´*NK2´ / ´*NK4´	
Indicator of whether the library is (UPAM) protected	varname#.PROT-LIB	S	´*YES´ / ´*NO´	
Library size in 2-K pages	varname#.F-SIZE	I	<integer>	
Number of available 2-K pages	varname#.FREE-SIZE	I	<integer>	

*2 : varname#.PROT-DEFI = ´*PAR´
*3 : varname#.PROT-DEFI = ´*PAR´ and varname#.xxxx.DEFI = ´*BY-GUARD´
*4 : varname#.PROT-DEFI = ´*PAR´ and varname#.xxxx.DEFI = ´*PAR´
*5 : *4 and varname#.xxxx.USER-DEFI = ´*LIST´

8.5 SHOW-STATISTICS statement

Output information	Name of S variable	T	Contents	Con.
Number of primary members (result: “same”)	varname#.SAME.NUM-OF-PRIMARY	I	<integer>	
Number of secondary members (result: “same”)	varname#.SAME.NUM-OF-SECONDARY	I	<integer>	
Number of primary member records (result: “same”)	varname#.SAME.LINE.PRIMARY	I	<integer>	
	varname#.SAME.LINE.INS	I	0	
Number of same records (result: “same”)	varname#.SAME.LINE.SAME	I	<integer>	
	varname#.SAME.LINE.DEL	I	0	
Number of secondary member records (result: “same”)	varname#.SAME.LINE. SECONDARY	I	<integer>	
Number of primary members (result: “changed”)	varname#.CHA.NUM-OF-PRIMARY	I	<integer>	
Number of secondary members (result: “changed”)	varname#.CHA.NUM-OF-SECONDARY	I	<integer>	
Number of primary member records (result: “changed”)	varname#.CHA.LINE.PRIMARY	I	<integer>	
Number of inserted records (result: “changed”)	varname#.CHA.LINE.INS	I	<integer>	
Number of same records (result: “changed”)	varname#.CHA.LINE.SAME	I	<integer>	
Number of deleted records (result: “changed”)	varname#.CHA.LINE.DEL	I	<integer>	
Number of secondary member records (result: “changed”)	varname#.CHA.LINE. SECONDARY	I	<integer>	
Number of primary members (result: “inserted”)	varname#.INS.NUM-OF-PRIMARY	I	<integer>	
	varname#.INS.NUM-OF-SECONDARY	I	0	
Number of primary member records (result: “inserted”)	varname#.INS.LINE.PRIMARY	I	<integer>	
Number of inserted records (result: “inserted”)	varname#.INS.LINE.INS	I	<integer>	
	varname#.INS.LINE.SAME	I	0	
	varname#.INS.LINE.DEL	I	0	
Number of secondary member records (result: “inserted”)	varname#.INS.LINE.SECONDARY	I	0	
	varname#.DEL.NUM-OF-PRIMARY	I	0	

Output information	Name of S variable	T	Contents	Con.
Number of secondary members (result: “deleted”)	varname#.DEL.NUM-OF-SECONDARY	I	<integer>	
	varname#.DEL.LINE.PRIMARY	I	0	
	varname#.DEL.LINE.INS	I	0	
	varname#.DEL.LINE.SAME	I	0	
Number of deleted records (result: “deleted”)	varname#.DEL.LINE.DEL	I	<integer>	
Number of secondary member records (result: “deleted”)	varname#.DEL.LINE.SECONDARY	I	<integer>	
Number of primary members (total)	varname#.TOTAL.NUM-OF-PRIMARY	I	<integer>	
Number of secondary members (total)	varname#.TOTAL.NUM-OF-SECONDARY	I	<integer>	
Number of primary member records (total)	varname#.TOTAL.LINE.PRIMARY	I	<integer>	
Number of inserted records (total)	varname#.TOTAL.LINE.INS	I	<integer>	
Number of same records (total)	varname#.TOTAL.LINE.SAME	I	<integer>	
Number of deleted records (total)	varname#.TOTAL.LINE.DEL	I	<integer>	
Number of secondary member records (total)	varname#.TOTAL.LINE.SECONDARY	I	<integer>	

Fields with a value of 0 are created in order to keep the variable substructures uniform.

8.6 SHOW-TYPE-ATTRIBUTES statement

Output information	Name of S variable	T	Contents	Con.
Library name	varname#.LIB	S	<filename 1..54>	
Member type	varname#.TYPE	S	<alphanum 1..8>	
Name of supertype	varname#.SUPER-TYPE	S	´*NONE´ / <alphanum 1..8>	
Name of base type	varname#.BASE-TYPE	S	<alphanum 1..8>	
Current version convention	varname#.CONVENTION	S	´*NONE´ / ´*STD-TREE´ / ´*STD-SEQ´ / '*MUL-SEQ'	
Current version example (for *STD-SEQ and *MUL-SEQ)	varname#.CONVENTION-EXAMPLE	S	´´ / <comp-name 1..24>	
Permissible storage form	varname#.STOR-FORM	S	´*STD´ / ´*NONE´ / ´*FULL´ / ´*DELTA´	
Control of additional checks	varname#.WRITE-CONTR	S	´*NONE´ / ´*ACTIVATE´ / ´*DEACTIVATE´	
Indicator for administration protection	varname#.ADM.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	
Name of administration guard	varname#.ADM.GUARD-NAME	S	<filename 1..18>	*3
Indicator for administer authorization (BACL)	varname#.ADM.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for administer authorization of owner	varname#.ADM.OWNER	S	´´ / ´*OWNER´	*5
Indicator for administer authorization of owner's group	varname#.ADM.GROUP	S	´´ / ´*GROUP´	*5
Indicator for administer authorization for OTHERS	varname#.ADM.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether an administration password has been defined for the member	varname#.ADM.PASS	S	´*NO´ / ´*YES´	*4
Indicator of whether member protection has been defined	varname#.PROT-DEFI	S	´*NONE´ / ´*PAR´	
Indicator for read protection	varname#.PROT.READ-DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of read guard	varname#.PROT.READ.GUARD-NAME	S	<filename 1..18>	*3
Indicator for read authorization (BACL)	varname#.PROT.READ.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for read authorization of owner	varname#.PROT.READ.OWNER	S	´´ / ´*OWNER´	*5
Indicator for read authorization of owner's group	varname#.PROT.READ.GROUP	S	´´ / ´*GROUP´	*5
Indicator for read authorization for OTHERS	varname#.PROT.READ.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether a read password has been defined for the member	varname#.PROT.READ.PASS	S	´*NO´ / ´*YES´	*4

Output information	Name of S variable	T	Contents	Con.
Indicator for write protection	varname#.PROT.WRITE.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of write guard	varname#.PROT.WRITE.GUARD-NAME	S	<filename 1..18>	*3
Indicator for write authorization (BACL)	varname#.PROT.WRITE.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for write authorization of owner	varname#.PROT.WRITE.OWNER	S	´´ / ´*OWNER´	*5
Indicator for write authorization of owner's group	varname#.PROT.WRITE.GROUP	S	´´ / ´*GROUP´	*5
Indicator for write authorization for OTHERS	varname#.PROT.WRITE.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether a write password has been defined for the member	varname#.PROT.WRITE.PASS	S	´*NO´ / ´*YES´	*4
Indicator for execute protection	varname#.PROT.EXEC.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of execute guard	varname#.PROT.EXEC.GUARD-NAME	S	<filename 1..18>	*3
Indicator for execute authorization (BACL)	varname#.PROT.EXEC.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for execute authorization of owner	varname#.PROT.EXEC.OWNER	S	´´ / ´*OWNER´	*5
Indicator for execute authorization of owner's group	varname#.PROT.EXEC.GROUP	S	´´ / ´*GROUP´	*5
Indicator for execute authorization for OTHERS	varname#.PROT.EXEC.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether an execute password has been defined for the member	varname#.PROT.EXEC.PASS	S	´*NO´ / ´*YES´	*4
Indicator for hold protection	varname#.PROT.HOLD.DEFI	S	´*NONE´ / ´*BY-GUARD´ / ´*PAR´	*2
Name of hold guard	varname#.PROT.HOLD.GUARD-NAME	S	<filename 1..18>	*3
Indicator for hold authorization (BACL)	varname#.PROT.HOLD.USER-DEFI	S	´*NONE´ / ´*LIST´	*4
Indicator for hold authorization of owner	varname#.PROT.HOLD.OWNER	S	´´ / ´*OWNER´	*5
Indicator for hold authorization of owner's group	varname#.PROT.HOLD.GROUP	S	´´ / ´*GROUP´	*5
Indicator for hold authorization for OTHERS	varname#.PROT.HOLD.OTHERS	S	´´ / ´*OTHERS´	*5
Indicator of whether a hold password has been defined for the member	varname#.PROT.HOLD.PASS	S	´*NO´ / ´*YES´	*4

*2 : varname#.PROT-DEFI = ´*PAR´
*3 : varname#.PROT-DEFI = ´*PAR´ and varname#.xxxx.DEFI = ´*BY-GUARD´
*4 : varname#.PROT-DEFI = ´*PAR´ and varname#.xxxx.DEFI = ´*PAR´
*5 : *4 and varname#.xxxx.USER-DEFI = ´*LIST´

9 Examples

The following chapter uses execution examples to illustrate certain typical LMS applications.

In the examples, user inputs are indicated by means of lowercase letters and bold print.

9.1 Adding, correcting and assembling library source programs

A source program is added as an S-type member to a library and then assembled. Since errors were found during assembly, the member is corrected with EDT and subsequently assembled again. The module from the EAM area is added to the same library as an R-type member.

```
/start-lms ──────────────────────────────────────────────────────────────────── (1)
//modify-logging-parameters logging=*maximum ──────────────────────────────────── (2)
//modify-lms-defaults version=*increment ──────────────────────────────────────── (3)
//open-library library=bsp1.bib,mode=*update ──────────────────────────────────── (4)
LIBRARY IS CLEARED AND PREPARED ─────────────────────────────────────────────── (5)
```

(1) LMS is called.

(2) In addition to error messages, positive acknowledgments are also logged.

(3) At every write operation, the target version is incremented by 1.

(4) Library BSP1.BIB is to be created as a new library and assigned as an I/O library.

(5) Library BSP1.BIB has been created.

```
//add-element from-file=quell.erfass,to-elem=(elem=erfass,type=s) ——————————  (6)
INPUT  FILE
OUTPUT LIBRARY= :1OSN:$USER.BSP1.BIB
         ADD :1OSN:$USER.QUELL.ERFASS AS (S)ERFASS/001(0001)/2013-03-01 ———————  (7)
//show-element-attributes ——————————————————————————————————————————————  (8)
INPUT  LIBRARY= :1OSN:$USER.BSP1.BIB
TYP NAME   VER (VAR#) DATE
(S) ERFASS 001 (0001) 2013-03-01
     1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS ———————————————————————————  (9)
//show-element (element=erfass, type=s) ———————————————————————————————— (10)
INPUT  LIBRARY= :1OSN:$USER.BSP1.BIB
INPUT  ELEMENT= (S)ERFASS/001(0001)/2013-03-01
         TITLE 'ERFASSEN VON DATEN'
         PRINT NOGEN
ERFAS    START
         BALR  5,0
         USING *,5
         OPEN  DATEI,OUTPUT
LESEN    RDATA SATZ,ENDPGM
         PUT   DATEI,SATZ
         B     LESEN
ENDPGM   TERM
*
DATEI    FXB   FCBTYPE=SAM, LINK=DATEN
SATZ     DS    CL84
         END
NUMBER OF PROCESSED RECORDS IS      14 ———————————————————————————————————— (11)
//end ——————————————————————————————————————————————————————————————————— (12)
```

(6) File QUELL.ERFASS is added to the library as an S-type member having the name ERFASS.

(7) Positive acknowledgment: member ERFASS, with the version designation 001 and variant number 0001, is written to the library.

(8) The directory of library BSP1.BIB is to be listed.

(9) Directory entry of library BSP1.BIB.

(10) Member ERFASS is to be listed.

(11) Contents of member ERFASS.

(12) LMS is terminated.

```
/start-assembh ──────────────────────────────────────────────────────────── (13)
//compile source=*library-element(lib=bsp1.bib,element=erfass) ──────────── (14)
%  ASS6011 ASSEMBLY TIME: 820 MSEC ──────────────────────────────────────── (15)
%  ASS6018 1 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: SIGNIFICANT ERROR
%  ASS6006 LISTING GENERATOR TIME: 439 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-lms ──────────────────────────────────────────────────────────────── (16)
//modify-logging-parameters logging=*maximum ──────────────────────────────── (17)
//modify-lms-defaults version=*increment ──────────────────────────────────── (18)
//open-library library=bsp1.bib,mode=*update ──────────────────────────────── (19)
//edit-element (element=erfass,type=s) ──────────────────────────────────── (20)
```

(13) The assembler is called.

(14) The source program in member ERFASS of library BSP1.BIB is to be assembled.

(15) The program contains errors.

(16) LMS is called again.

(17) In addition to error messages, positive acknowledgments are also logged.

(18) The target version is incremented by 1 during each write operation.

(19) Library BSP1.BIB is opened for writing and reading.

(20) Member ERFASS is to be processed with EDT.

```
 0.10          TITLE 'ERFASSEN VON DATEN'
 0.20          PRINT NOGEN
 0.30 ERFAS    START
 0.40          BALR 5,0
 0.50          USING *,5
 0.60          OPEN  DATEI,OUTPUT
 0.70 LESEN    RDATA SATZ,ENDPGM
 0.80          PUT   DATEI,SATZ
 0.90          B     LESEN
 1.00 ENDPGM   TERM
 1.10 *
x1.20 DATEI    FcB   FCBTYPE=SAM, LINK=DATEN
 1.30 SATZ           DS    CL84                                          (21)
 1.40 END
 2.40
 3.40
 4.40
 5.40
 6.40
 7.40
 8.40
 9.40
                     OUTPUT ELEMENT= (S)ERFASS/@(0002)/2013-03-01

halt                                                      0000.10:001(0)
```

```
%  LMS0420 EDITED ELEMENT (S)ERFASS/002(0002)/2013-03-01 TO BE ADDED? REPLY
(Y=YES; N=NO OR R=RETURN TO EDITOR)
y ──────────────────────────────────────────────────────────────────────── (22)
INPUT  LIBRARY= :1OSN:$USER.BSP1.BIB
OUTPUT LIBRARY= :1OSN:$USER.BSP1.BIB
INPUT  ELEMENT= (S)ERFASS/001(0001)/2013-03-01
OUTPUT ELEMENT= (S)ERFASS/002(0002)/2013-03-01
       CORRECT (S)ERFASS/001(0001)/2013-03-01 AS (S)ERFASS/002(0001)/2013-03-01
──────────────────────────────────────────────────────────────────────── (23)
//end
```

(21) The error is corrected:

- Make line 1.20 overwriteable by entering “x” in the statement column.
- Change “FcB” in line 1.20 to “FCB” and terminate EDT by entering HALT in the statement line.

(22) “Y” causes the corrected member to be added with version 002 to the output library. The old version (version 001) is retained.

(23) Positive acknowledgment: input member ERFASS has been corrected. The output member is given the same name and the version 002.

```
/delete-system-file system-file=*omf
/start-assembh
//compile source=*library(lib=bsp1.bib,element=erfass)
%  ASS6011 ASSEMBLY TIME: 1037 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 406 MSEC
//end
%  ASS6012 END OF ASSEMBH ------------------------------------------------------ (24)
/start-lms ------------------------------------------------------------------- (25)
//modify-logging-parameters logging=*maximum ---------------------------------- (26)
//open-library library=BSP1.BIB,mode=*update ---------------------------------- (27)
//add-element from-file=*omf,to-elem=(type=r) --------------------------------- (28)
INPUT  OMF
OUTPUT LIBRARY= :1OSN:$USER.BSP1.BIB
         ADD ERFAS AS (R)ERFAS/@(0001)/2013-03-01
//end ------------------------------------------------------------------------- (29)
```

(24) The assembly run has been executed successfully.

(25) LMS is called.

(26) In addition to error messages, positive acknowledgments are also logged.

(27) Library BSP1.BIB is again opened for reading and writing.

(28) Module ERFAS is taken from the EAM area and incorporated as member ERFAS.

(29) LMS is terminated.

9.2 Copying members

Members from various libraries are copied to another library.

```
/add-file-link link-name=lib1,file-name=modul.lib ————————————————————— (1)
/start-lms ——————————————————————————————————————————————————————————— (2)
//modify-logging-parameters logging=*maximum ————————————————————————— (3)
//open-library library=bsp2.bib,mode=*update ————————————————————————— (4)
LIBRARY IS CLEARED AND PREPARED
//show-element-attributes (library=*link(link-name=lib1),type=r) ———————— (5)
INPUT  LIBRARY= :1OSN:$USER.MODUL.LIB,LINK=LIB1
TYP NAME   VER (VAR#) DATE
(R) MODERF @   (0001) 2013-03-01
       1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//copy-element elem=(library=*link(link-name=lib1),elem=moderf,type=r), -
//            to-element=(library=*std,element=mod.erf) ————————————————— (6)
INPUT  LIBRARY= :1OSN:$USER.MODUL.LIB,LINK=LIB1
OUTPUT LIBRARY= :1OSN:$USER.BSP2.BIB
         COPY (R)MODERF/@(0001)/2011-02-19 AS (R)MOD.ERF/@(0001)/2011-02-19
```

(1) This input enables the library MODUL.LIB to be assigned via the link name LIB1 during the LMS run.

(2) LMS is called.

(3) In addition to error messages, positive acknowledgments are also logged.

(4) The new library BSP2.BIB is created as a global library and opened for writing and reading.

(5) The directory of library MODUL.LIB, that is specified via link name LIB1, is to be listed.

(6) Module MODERF from library MODUL.LIB is copied to library BSP2.BIB under the member name MOD.ERF.

```
//show-element-attributes (library=macro.lib,type=m) ——————————————————  (7)
INPUT  LIBRARY= :1OSN:$USER.MACRO.LIB
TYP NAME VER (VAR#) DATE         NAME VER (VAR#) DATE
(M) MAC1 @   (0001) 2011-02-19 MAC2 @   (0001) 2011-02-19
       2 (M)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//copy-element elem=(library=macro.lib,elem=mac*,type=m), -
//           to-element=(library=*std,element=mu*) ——————————————————————  (8)
INPUT  LIBRARY= :1OSN:$USER.MACRO.LIB
OUTPUT LIBRARY= :1OSN:$USER.BSP2.BIB
         COPY (M)MAC1/@(0001)/2011-02-19 AS (M)MU1/@(0001)/2011-02-19
         COPY (M)MAC2/@(0001)/2011-02-19 AS (M)MU2/@(0001)/2011-02-19 ————————  (9)
//show-element-attributes (library=quell.lib,type=s) ——————————————————— (10)
INPUT  LIBRARY= :1OSN:$USER.QUELL.LIB
TYP NAME    VER (VAR#) DATE         NAME    VER (VAR#) DATE
(S) EDTB    @   (0001) 2011-02-19   PROT    @   (0001) 2011-02-19
(S) SEINAUS @   (0001) 2011-02-19   SERFAS  @   (0001) 2011-02-19
       4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//copy-element elem=(library=quell.lib,elem=*,type=s, -
//                  except-element=(element=edtb)) ——————————————————————— (11)
INPUT  LIBRARY= :1OSN:$USER.QUELL.LIB
OUTPUT LIBRARY= :1OSN:$USER.BSP2.BIB
         COPY (S)PROT/@(0001)/2011-02-19 AS (S)PROT/@(0001)/2011-02-19
         COPY (S)SEINAUS/@(0001)/2011-02-19 AS (S)SEINAUS/@(0001)/2011-02-19
         COPY (S)SERFAS/@(0001)/2011-02-19 AS (S)SERFAS/@(0001)/2011-02-19
```

(7) The directory of library MACRO.LIB, that is assigned as a local library, is to be listed.

(8) Those members of library MACRO.LIB whose member designations begin with “MAC” are to be copied to library BSP2.BIB. The new member designations begin with “MU”. The member designations of the input members are transferred, starting from the third position.

(9) Positive acknowledgment: the selected members, with the new member designations, are copied.

(10) The directory of library QUELL.LIB, that is assigned as a local library, is to be listed.

(11) All members from library QUELL.LIB, with the exception of member EDTB, are copied to the output library.

```
//show-element-attributes (library=test.lib) ------------------------------ (12)
INPUT  LIBRARY= :1OSN:$USER.TEST.LIB
TYP NAME   VER (VAR#) DATE
(S) SERFAS @   (0001) 2011-02-19
       1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//copy-element elem=(library=test.lib,elem=*,type=*), -
//             to-elem=(library=*std,element=*(version=007)) ---------------------- (13)
INPUT  LIBRARY= :1OSN:$USER.TEST.LIB
OUTPUT LIBRARY= :1OSN:$USER.BSP2.BIB
         COPY (S)SERFAS/@(0001)/2011-02-19 AS (S)SERFAS/007(0001)/2011-02-19
```

(12) The directory of program library TEST.LIB, that is assigned as a local library, is to be listed.

(13) All members of program library TEST.LIB are copied to the output library and stored under the same name and version number 007.

```
//show-element-attributes ------------------------------------------------- (14)
INPUT  LIBRARY= :1OSN:$USER.BSP2.BIB
TYP NAME VER (VAR#) DATE         NAME VER (VAR#) DATE
(M) MU1  @   (0001) 2011-02-19   MU2  @   (0001) 2011-02-19
      2 (M)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME    VER (VAR#) DATE
(R) MOD.ERF @   (0001) 2011-02-19
      1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME    VER (VAR#) DATE         NAME    VER (VAR#) DATE
(S) PROT    @   (0001) 2011-02-19   SEINAUS @   (0001) 2011-02-19
(S) SERFAS  007 (0001) 2011-02-19   SERFAS  @   (0001) 2011-02-19
      4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
------------------------------------------------------------------------------
      7 ELEMENT(S) IN THIS TABLE OF CONTENTS
//show-library-status ----------------------------------------------------- (15)
STATUS FILENAME                                          MODE    LINK      DEF-TYPE
CLOSED :1OSN:$USER.TEST.LIB
CLOSED :1OSN:$USER.QUELL.LIB
CLOSED :1OSN:$USER.MACRO.LIB
CLOSED :1OSN:$USER.MODUL.LIB                                     LIB1
OPEN   :1OSN:$USER.BSP2.BIB                             UPDATE
//close-library ----------------------------------------------------------- (16)
//show-library-status ----------------------------------------------------- (17)
STATUS FILENAME                                          MODE    LINK      DEF-TYPE
CLOSED :1OSN:$USER.TEST.LIB
CLOSED :1OSN:$USER.QUELL.LIB
CLOSED :1OSN:$USER.MACRO.LIB
CLOSED :1OSN:$USER.MODUL.LIB                                     LIB1
CLOSED :1OSN:$USER.BSP2.BIB
//end --------------------------------------------------------------------- (18)
```

(14) The directory of the current input library BSP2.BIB is to be listed with all members. There are two members with member name SERFAS, but with different version numbers.

(15) The status of the libraries used during the LMS run is queried.

(16) Library BSP2.BIB is closed.

(17) The status of the libraries used during the LMS run is queried.

(18) The LMS run is terminated.

9.3 Comparing members

Member ERFASS (listed in the example on page 458) and member EINAUS are compared. A comparison log is generated.

```
/start-lms ──────────────────────────────────────────────────────────────── (1)
//modify-logging-parameters logging=*maximum ──────────────────────────────── (2)
//open-library library=bsp1.bib,mode=*update ──────────────────────────────── (3)
//add-element from-file=quell.einaus, to-elem=(elem=einaus,type=s) ──────────── (4)
INPUT  FILE
OUTPUT LIBRARY= :1OSN:$USER.BSP1.BIB
         ADD :1OSN:$USER.QUELL.EINAUS AS (S)EINAUS/@(0001)/2013-03-01
```

(1) LMS is called.

(2) In addition to error messages, positive acknowledgments are also logged.

(3) Library BSP1.BIB is opened for reading and writing.

(4) File QUELL.EINAUS is added to the library as S-type member EINAUS.

```
//show-element (element=einaus,type=s) ──────────────────────────────────────── (5)
INPUT  LIBRARY= :1OSN:$USER.BSP1.BIB
INPUT  ELEMENT= (S)ES/@(0001)/2013-03-01
         TITLE 'ERFASSEN VON DATEN'
         PRINT NOGEN
ERFAS    START
         BALR  5,0
         USING *,5
         OPEN  DATEI,OUTPUT
LESEN    RDATA SATZ,ENDPGM
         CLC   TEXT(4),=C'/EOF'
         BE    ENDPGM
         MVC   ATEXT,TEXT
         LH    9,SL
         AH    9,=H'1'
         STH   9,ASL
         WROUT ASATZ,ENDPGM
         PUT   DATEI,SATZ
         B     LESEN
ENDPGM   TERM
*
DATEI    FCB   FCBTYPE=SAM,LINK=DATEN
         DS    0H
SATZ     DS    CL84
SL       DS    CL2
         DS    CL2
TEXT     DS    CL80
ASATZ    DS    0CL85
ASL      DS    CL2
         DC    X'000001'
ATEXT    DS    CL80
         END
NUMBER OF PROCESSED RECORDS IS      29
```

(5) Member EINAUS is listed.

```
//show-element (element=erfass,type=s) ------------------------------------ (6)
INPUT  LIBRARY= :1OSQ:$USER.BSP1.BIB
INPUT  ELEMENT= (S)ERFASS/002(0001)/2013-03-01
         TITLE 'ERFASSEN VON DATEN'
         PRINT NOGEN
ERFAS    START
         BALR 5,0
         USING *,5
         OPEN  DATEI,OUTPUT
LESEN    RDATA SATZ,ENDPGM
         PUT   DATEI,SATZ
         B     LESEN
ENDPGM   TERM
*
DATEI    FCB   FCBTYPE=SAM, LINK=DATEN
SATZ     DS    CL84
         END
NUMBER OF PROCESSED RECORDS IS      14

//compare-element primary-elem=(elem=einaus,type=s), -
//                secondary-elem=(elem=erfass), -
//                compare-parameters = *par(information=*maximum) ------------- (7)
FUNCTION          = C O M P A R E
PAR         COMPARE= 00001/32764/L/MAX
PRIMARY     LIBRARY= :1OSN:$USER.BSP1.BIB
PRIMARY     ELEMENT= (S)EINAUS/@(0001)/2013-03-01
SECONDARY   LIBRARY= :1OSN:$USER.BSP1.BIB
SECONDARY   ELEMENT= (S)ERFASS/002(0001)/2013-03-01 ---------------------------- (8)
------------------------------------------------------------------------------

SAME FROM       #1 TO       #7 AS FROM       #1 TO       #7 ------------------- (9)
      #1 >         TITLE 'ERFASSEN VON DATEN'<
      #2 >         PRINT NOGEN<
      #3 >ERFAS    START<
      #4 >         BALR  5,0<
      #5 >         USING *,5<
      #6 >         OPEN  DATEI,OUTPUT
      #7 >LESEN    RDATA SATZ,ENDPGM<
```

(6) The member ERFASS is listed.

(7) Members EINAUS and ERFASS are compared. The comparison log is to be output in its full scope.

(8) Start of comparison log:
The log includes the values set for COMPARE, the names of the primary and secondary libraries, and the names of the primary and secondary members.

(9) The records with record IDs #1 through #7 are identical in both members.

```
INS. FROM      #8 TO     #14 ------------------------------------------------- (10)
      #8 >         CLC   TEXT(4),=C'/EOF'<
      #9 >         BE    ENDPGM<
     #10 >         MVC   ATEXT,TEXT<
     #11 >         LH    9,SL<
     #12 >         AH    9,=H'1'<
     #13 >         STH   9,ASL<
     #14 >         WROUT ASATZ,ENDPGM< ---------------------------------------- (11)
-------------------------------------------------------------------------------
SAME FROM     #15 TO     #19 AS FROM      #8 TO     #12 ---------------------- (12)
     #15 >         PUT   DATEI,SATZ <
     #16 >         B     LESEN<
     #17 >ENDPGM   TERM<
     #18 >*<
     #19 >DATEI    FCB   FCBTYPE=SAM, LINK=DATEN< ----------------------------- (13)
-------------------------------------------------------------------------------
INS.          #20 ------------------------------------------------------------- (14)
     #20 >         DS    0H< ------------------------------------------------- (15)
-------------------------------------------------------------------------------
```

(08) –(21) Comparison log

(10) Records #8 through #14 are present in the primary member only and are represented as INS(erted).

(11) Output of inserted records.

(12) Records #15 through #19 of the primary member are identical to records #8 through #12 of the secondary member.

(13) Output of identical records.

(14) Record #20 is present in the primary member only and is represented as INS(erted).

(15) Output of inserted record.

```
SAME          #21            AS          #13 ———————————————————————————— (16)
     #21 >SATZ          DS    CL84< ———————————————————————————————————— (17)
————————————————————————————————————————————————————————————————————————————

INS. FROM     #22 TO     #28 ———————————————————————————————————————————— (18)
     #22 >SL       DS    CL2<
     #23 >         DS    CL2<
     #24 >TEXT     DS    CL80<
     #25 >ASATZ    DS    OCL85<
     #26 >ASL      DS    CL2<
     #27 >         DC    X'000001'<
     #28 >ATEXT    DS    CL80< ———————————————————————————————————————— (19)
————————————————————————————————————————————————————————————————————————————

SAME          #29            AS          #14 ———————————————————————————— (20)
     #29 >              END< ———————————————————————————————————————— (21)
————————————————————————————————————————————————————————————————————————————

PRIMARY    ELEMENT= (S)EINAUS/@(0001)/2013-03-01
SECONDARY  ELEMENT= (S)ERFASS/002(0001)/2013-03-01
RESULT: C    PRIMARY=     29 INSERTED=     15 (    3)  DELETED=      0 (    0)
           SECONDARY=     14     SAME=     14 (    4) ———————————————————— (22)
//end ———————————————————————————————————————————————————————————————— (23)
```

(16) Record #21 of the primary member is identical to record #13 of the secondary member.

(17) Output of identical record.

(18) Records #20 through #28 are present in the primary member only and are represented as INS(erted).

(19) Output of inserted records.

(20) Record #29 of the primary member is identical to record #14 of the secondary member.

(21) Output of identical record.

(22) Result of the comparison; output of the number of records of the primary and secondary members, and of the number of inserted, identical and deleted records.

The numbers in parentheses indicate how many continuous sections (consisting of consecutive records) have been inserted, identified as identical, or deleted.

(23) LMS is terminated.

9.4 Processing delta members

```
/start-lms ──────────────────────────────────────────────────────────────────
//modify-logging-parameters logging=*maximum                                  (1)
//open-library library=bsp4.bib,mode=*update ──────────────────────────────   (2)
LIBRARY IS CLEARED AND PREPARED
//add-element from-file=workelem,to-elem=(elem=delta(v=v00),type=s, -
//            storage-form=*delta) ────────────────────────────────────────   (3)
INPUT  FILE
OUTPUT LIBRARY= :1OSN:$USER.BSP4.BIB
         ADD :1OSN:$USER.WORKELEM AS (S)DELTA/V00(0001)/2013-03-01
             , FIRST DELTA VERSION
//edit-element (,element=delta(version=v00),type=s), -
//             to-elem=(,elem=delta(v=v01)) ──────────────────────────────────   (4)
```

```
 0.10 MINI    START
 0.20         BALR  3,0
 0.30         USING *,3
 0.40         OPEN  SAMFCB,OUTPUT
 0.50         PUT   SAMFCB,EINGABE
 0.60         TERM
 0.70 SAMFCB  FCB   FCBTYPE=SAM,LINK=MINI
 0.80 EINGABE DC    C´DA IST ER JA´
 0.90         END
 1.90
 2.90
 3.90
 4.90
 5.90
 6.90
 7.90
 8.90
 9.90
10.90
11.90
12.90
13.90
                    OUTPUT ELEMENT= (S)DELTA/V01(0002)/2011-02-27

halt                                                    0000.10:001(0)
```

(1) LMS is called.

(2) The new library BSP4.BIB is created.

(3) File WORKELEM is added to the library as a new S-type delta member, DELTA/V00.

(4) Delta member DELTA/V00 is to be processed with EDT and the result added as new member DELTA/V01 to DELTA/V00.

```
%  LMS0420 EDITED ELEMENT (S)DELTA/V01(0002)/2013-03-01 TO BE ADDED OR RETURN TO EDITOR ?
REPLY (Y=YES; N=NO OR R=RETURN)?
```

```
y
INPUT  LIBRARY= :1OSN:$USER.BSP4.BIB
OUTPUT LIBRARY= :1OSN:$USER.BSP4.BIB
INPUT  ELEMENT= (S)DELTA/V00(0001)/2013-03-01
OUTPUT ELEMENT= (S)DELTA/V01(0002)/2013-03-01
         CORRECT (S)DELTA/V00(0001)/2013-03-01 AS (S)DELTA/V01(0002)/2011-10-01
                 ON BASE (S)DELTA/V00(0002)/2013-03-01
//copy-element (library=lib.arbeit,elem=input,type=s), -
//             to-elem=(elem=delta(version=v02,base=v00)) ------------------------  (5)
INPUT  LIBRARY= :1OSN:$USER.LIB.ARBEIT
OUTPUT LIBRARY= :1OSN:$USER.BSP4.BIB
         COPY (S)INPUT/@(0001)/2011-02-19 AS (S)DELTA/V02(0003)/2011-02-19
              ON BASE (S)DELTA/V00(0003)/2011-02-19
//show-element-attributes information=*delta-structure ------------------------  (6)
INPUT  LIBRARY= :1OSN:$USER.BSP4.BIB
TYP        NAME                        VERSION    (VAR#) DATE       DLT#  BASE#
(S       ) DELTA . . . . . . . . . . . V00  . . . (0000)            00001 00000
(S       ) DELTA . . . . . . . . . . . V01  . . . (0000)            00002 00001
(S       ) DELTA . . . . . . . . . . . V02  . . . (0000)            00003 00001
------------------------------------------------------------------------------

       3 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
```

(5) The non-delta member INPUT is copied from library LIB.ARBEIT to the global library as S-type delta member DELTA/V02 to the base V00.

(6) The directory of library BSP4.BIB is to be listed as a complete delta tree.

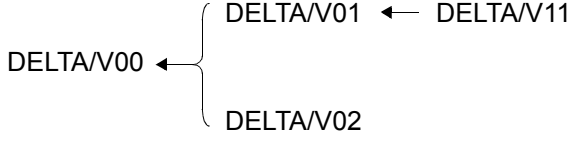
```
//add-element workelem,(,delta(v11,v01),s,,delta) ————————————————————————— (7)
INPUT  FILE
OUTPUT LIBRARY= :1OSN:$USER.BSP4.BIB
         ADD :1OSN:$USER.WORKELEM AS (S)DELTA/V11(0004)/2013-03-01 ON BASE
             (S)DELTA/V01(0004)/2011-10-01
//show-element-attributes information=*delta-structure ————————————————————— (8)
INPUT  LIBRARY= :1OSN:$USER.BSP4.BIB
TYP        NAME                        VERSION    (VAR#) DATE       DLT#  BASE#
(S       ) DELTA . . . . . . . . . . . V00  . . . (0000)            00001 00000
(S       ) DELTA . . . . . . . . . . . V01  . . . (0000)            00002 00001
(S       ) DELTA . . . . . . . . . . . V02  . . . (0000)            00003 00001
(S       ) DELTA . . . . . . . . . . . V11  . . . (0000)            00004 00002
————————————————————————————————————————————————————————————————————————————

      4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//end ————————————————————————————————————————————————————————————————————— (9)
```

(7) File WORKELEM is added as type-S delta member DELTA/V11 to delta member DELTA/V01. This statement is an example of an input using only positional operands.

(8) The directory of library BSP4.BIB as a delta tree is to be listed.

(9) LMS is terminated.

The delta members now have the following structure:



9.5 Modifying an object module

The member USELST is added to the library as an object module and modified there by means of a substatement.

```
/start-assembh ——————————————————————————————————————————————————— (1)
//compile source=*library-element(lib=bsp5.bib,element=uselst), -
//module-lib=bsp5.bib(elemnt=uselst) ————————————————————————————————— (2)
%  ASS6011 ASSEMBLY TIME: 240 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 453 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-lms ——————————————————————————————————————————————————————— (3)
//modify-logging-parameters logging=*maximum ————————————————————————— (4)
//open-library library=bsp5.bib,mode=*update ————————————————————————— (5)
//show-element-attributes (type=r) ——————————————————————————————————— (6)
INPUT  LIBRARY= :1OSN:$USER.BSP5.BIB
TYP NAME   VER (VAR#) DATE
(R) USELST @   (0001) 2013-03-01
       1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
```

(1) The assembler is called to generate an object module.

(2) The source program USELST is to be assembled. On error-free assembly, the object module thus generated is written to library BSP5.BIB. The object module automatically receives member type R.

(3) LMS is called.

(4) In addition to error messages, positive acknowledgments are also logged.

(5) Library BSP5.BIB is assigned. It must be opened for reading and writing, otherwise modification of the object module is impossible.

(6) Display directory for all R-type members. The library now contains object module USELST generated from the assembler run.

```
//modify-element (element=uselst,type=r) ————————————————————————————————— (7)
//add-text-modification address=x´c0´,new-contents=´aa´(old=´ER´) ———————————— (8)
//end-modify ————————————————————————————————————————————————————————————— (9)
INPUT  LIBRARY= :1OSN:$USER.BSP5.BIB
OUTPUT LIBRARY= :1OSN:$USER.BSP5.BIB
INPUT  ELEMENT= (R)USELST/@(0001)/2013-03-01
OUTPUT ELEMENT= (R)USELST/@(0002)/2013-03-01
TEXT-ADR:            000000C0
TEXT BEFORE CHANGE:  E R
                     C5D9
TEXT AFTER  CHANGE:  a a
                     8181 ——————————————————————————————————————————————— (10)
        CORRECT (R)USELST/@(0001)/2013-03-01 AS (R)USELST/@(0002)/2013-03-01
                , OUTPUT REPLACED ——————————————————————————————————————— (11)
//show-element-attributes (type=r) ——————————————————————————————————————— (12)
INPUT  LIBRARY= :1OSN:$USER.BSP5.BIB
TYP NAME   VER (VAR#) DATE
(R) USELST @   (0002) 2013-03-01
      1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//end ———————————————————————————————————————————————————————————————————— (13)
```

(7) Module USELST is to be modified.

(8) This substatement replaces the text ’ER’ with ’aa’ at address c0.

(9) Substatement input is terminated.

(10) The text at address 0000C0 is replaced.

(11) Message indicating that the change has been made and the original member is replaced by the modified member.

(12) Display directory for all R-type members. The library now contains the modified object module.

(13) LMS is terminated.

9.6 Generating SAM/ISAM files

EDT is used to create a SAM file. This file is added as a member and stored in the form of two different types of file:

– as a SAM file, on the basis of the file attributes stored in the member

– as an ISAM file using ISAM keys created by default.

```
/start-lms ──────────────────────────────────────────────────────────── (1)
//modify-logging-parameters logging=*maximum ─────────────────────────── (2)
//open-library library=bsp6.bib,mode=*update ─────────────────────────── (3)
LIBRARY IS CLEARED AND PREPARED
//modify-lms-defaults type=d ──────────────────────────────────────────── (4)
```

(1) LMS is called.

(2) In addition to error messages, positive acknowledgments are also logged.

(3) Library BSP6.BIB is opened.

(4) The default value for the member type will be set.

```
//call-edt ────────────────────────────────────────────────────────────── (5)
```

```
 1.00 BACH          SEBASTIAN      MUENCHEN       AUF DER HOEHE 7     AB 3
 2.00 BERGMANN      NORBERT        MUENCHEN       TORWEG 10           AB 5
 3.00 FINK          SUSANNE        NUERNBERG      RINGSTR. 23         AB 1
 4.00 MEYER         FRANZ          NUERNBERG      WASSERMUNGENWEG     AB 1
 5.00 GRUNDLER      WOLFGANG       BASEL          SONNENSTR. 11       AB 2
 6.00 KNOLL         MONIKA         FRANKFURT      BAUMALLEE 12        AB 3
 7.00 LIEDL         ERIKA          MUENCHEN       IN DER BREITE 1     AB 5
 8.00 WAGNER        JOHANN         AUGSBURG       AM SEE 45           AB 4
 9.00
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
18.00
19.00
20.00
21.00
22.00
23.00

write ´pers.dat´;halt                                        0000.10:001(0)
```

```
//modify-lms-defaults (source-attributes=*keep) ————————————————————————— (6)
//add-element from-file=pers.dat, to-elem=(,perdat) ————————————————————— (7)
INPUT  FILE
OUTPUT LIBRARY= :1OSN:$USER.BSP6.BIB
         ADD :1OSN:$USER.PERS.DAT AS (D)PERDAT/@(0001)/2013-03-01
```

(5) EDT is called in order to generate or process a file.

Subsequently the data is entered and stored as SAM file PERS.DAT by means of WRITE. HALT terminates EDT and returns control to LMS.

(6) The file attributes of the EDT file are retained.

(7) File PERS.DAT added to the library as a D-type member having the name PERDAT.

```
//show-element-attributes (type=d)
INPUT  LIBRARY= :1OSN:$USER.BSP6.BIB
TYP NAME   VER (VAR#) DATE
(D) PERDAT @   (0001) 2013-03-01
       1 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//extract-element (element=perdat) ——————————————————————————————————— (8)
INPUT  LIBRARY= :1OSN:$USER.BSP6.BIB
OUTPUT FILE
         EXTRACT (D)PERDAT/@(0001)/2013-03-01 AS :1OSN:$USER.PERDAT
//exec-sys-cmd (show-file-attributes perdat,information=*all-attr) ———————— (9)
```

```
00000003 :1OSN:$USER.PERDAT
  ------------------------------ HISTORY      ----------------------------------
  CRE-DATE   = 2013-03-01  ACC-DATE   = 2013-03-01  CHANG-DATE = 2013-03-01
  CRE-TIME   =   10:32:56  ACC-TIME   =   10:32:56  CHANG-TIME =   10:32:56
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------ SECURITY     ----------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2013-03-01
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO          ENCRYPTION = *NONE
  ------------------------------ BACKUP       ----------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------ ORGANIZATION ----------------------------------
  FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (V,N)       REC-SIZE   = 0
  AVAIL      = *STD
  WORK-FILE  = *NO         F-PREFORM  = *K          SO-MIGR    = *ALLOWED
  ------------------------------ ALLOCATION   ----------------------------------
  SUPPORT    = PUB         S-ALLOC    = 34          HIGH-US-PA = 1
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       1OSN.2       D34211-2
  NUM-OF-EXT = 1
:1OSN: PUBLIC:      1 FILE  RES=        3  FREE=        2  REL=        0 PAGES
```

(8) The directory of library BSP6.BIB for member type D is to be listed.

(9) Member PERDAT is output as file PERDAT. Since no file attributes have been specified for this file, LMS generates a SAM file in accordance with the file attributes stored.

(10) (The file attributes of the generated file are listed.

```
//extract-element (element=perdat), to-file=persdat, -
//                file-attributes=*parameters(access-method=*isam)
INPUT  LIBRARY= :1OSN:$USER.BSP6.BIB
OUTPUT FILE
         EXTRACT (D)PERDAT/@(0001)/2013-03-01 AS :1OSN:$USER.PERSDAT
//exec-sys-cmd (show-file-attributes persdat,information=all) ---------------------- (10)
```

```
00000003 :1OSN:$USER.PERSDAT
  ------------------------------ HISTORY -------------------------------------
  CRE-DATE   = 2013-03-01  ACC-DATE   = 2013-03-01  CHANG-DATE = 2013-03-01
  CRE-TIME   =   10:32:56  ACC-TIME   =   10:32:56  CHANG-TIME =   10:32:56
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------ SECURITY ------------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2013-03-01
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO          ENCRYPTION = *NONE
  ------------------------------ BACKUP --------------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------ ORGANIZATION --------------------------------
  FILE-STRUC = ISAM        BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (V,N)       REC-SIZE   = 0
  KEY-LEN    = 8           KEY-POS    = 5
  AVAIL      = *STD
  WORK-FILE  = *NO          F-PREFORM  = *K          SO-MIGR    = *ALLOWED
  ------------------------------ ALLOCATION ----------------------------------
  SUPPORT    = PUB         S-ALLOC    = 9           HIGH-US-PA = 3
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       1OSN.1       D34211-2
  NUM-OF-EXT = 1
:1OSN: PUBLIC:      1 FILE  RES=        3  FREE=        1  REL=        0 PAGES
//end ------------------------------------------------------------------------- (11)
```

(11) Member PERDAT is created as a file having the name PERSDAT. The ACCESS-METHOD operand is then used to specify that the file PERSDAT is an ISAM file.

(12) The file attributes of the generated file are listed.

(13) LMS is terminated.

9.7 Outputting comparison statistics

All members of several libraries are compared. After the comparison, the relevant comparison statistics and, at the end, the grand total of the comparisons performed in the LMS run are output.

This LMS run is performed without a global library, i.e. no OPEN-LIBRARY is specified. All required libraries are defined locally in statements.

```
/start-lms ──────────────────────────────────────────────────────────────────── (1)
//modify-logging-parameters logging=*maximum ──────────────────────────────────── (2)
//compare-element primary-elem=(lib=lib.all.v2,element=*,type=s), -
//              secondary-elem=(lib=lib.sou.v1,element=*,type=s), -
//              compare-parameters = (information=*summary) ──────────────────── (3)
FUNCTION          = C O M P A R E
PAR         COMPARE= 00001/32764/L/SUM
PRIMARY     LIBRARY= :1OSQ:$USER.LIB.ALL.V2
SECONDARY   LIBRARY= :1OSQ:$USER.LIB.SOU.V1
------------------------------------------------------------------------------
PRIMARY     ELEMENT= (S)EINAUS/@(0001)/2011-02-19
SECONDARY   ELEMENT= (S)EINAUS/@(0001)/2011-02-19
RESULT: S     PRIMARY=       8 INSERTED=       - (    -)  DELETED=       - (    -)
            SECONDARY=       8     SAME=       8 (    1)
------------------------------------------------------------------------------
PRIMARY     ELEMENT= (S)ERFASS/@(0002)/2011-02-19
SECONDARY   ELEMENT= (S)ERFASS/@(0001)/2011-02-19
RESULT: S     PRIMARY=      14 INSERTED=       - (    -)  DELETED=       - (    -)
            SECONDARY=      14     SAME=      14 (    1)
------------------------------------------------------------------------------
PRIMARY     ELEMENT= (S)PROT/@(0001)/2011-02-19
SECONDARY   ELEMENT= (S)PROT/@(0001)/2011-02-19
RESULT: S     PRIMARY=       5 INSERTED=       - (    -)  DELETED=       - (    -)
            SECONDARY=       5     SAME=       5 (    1) ──────────────────────── (4)
```

(1) LMS is called.

(2) In addition to error messages, positive acknowledgments are also logged.

(3) All S-type members of library LIB.ALL.V2 are compared with the members of library LIB.SOU.V1.

(4) The result of the comparison is output.

```
//show-statistics number=*c0 ————————————————————————————————————————— (5)
AREA C0
             PRIM.    PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.   SEC.
STATISTIC    ELEM.    LINES    LINES    LINES    LINES    LINES    LINES  ELEM.
S (SAME)         3       27        -       27        -        -       27      3
C (CHANGED)      0        0        0        0        0        0        0      0
I (INSERTED)     0        0        0        -        -        0        -      -
D (DELETED)      -        -        -        -        0        0        0      0
————————————————————————————————————————————————————————————————————————
   TOTAL         3       27        0       27        0        0       27      3
//compare-element primary-elem=(lib=lib.all.v2,element=*,type=m), -
//              secondary-elem=(lib=lib.mac.v1,element=*,type=m), -
//              compare-param =(information=*summary) ————————————————————— (6)
FUNCTION          = C O M P A R E
PAR         COMPARE= 00001/32764/L/SUM
PRIMARY     LIBRARY= :1OSQ:$USER.LIB.ALL.V2
SECONDARY   LIBRARY= :1OSQ:$USER.LIB.MAC.V1
————————————————————————————————————————————————————————————————————————
PRIMARY     ELEMENT= (M)MAC1/@(0001)/2011-02-19
SECONDARY   ELEMENT= (M)MAC1/@(0001)/2011-02-19
RESULT: S     PRIMARY=      5 INSERTED=      - (    -)  DELETED=      - (    -)
            SECONDARY=      5     SAME=      5 (    1)
————————————————————————————————————————————————————————————————————————
PRIMARY     ELEMENT= (M)MAC2/@(0001)/2011-02-19
SECONDARY   ELEMENT= (M)MAC2/@(0001)/2011-02-19
RESULT: S     PRIMARY=      5 INSERTED=      - (    -)  DELETED=      - (    -)
            SECONDARY=      5     SAME=      5 (    1)
————————————————————————————————————————————————————————————————————————
PRIMARY     ELEMENT= (M)MUC1/@(0001)/2011-02-19
RESULT: I     PRIMARY=      5 INSERTED=      5 (    1)  DELETED=      - (    -)
            SECONDARY=      -     SAME=      - (    -)
————————————————————————————————————————————————————————————————————————
PRIMARY     ELEMENT= (M)MUC2/@(0001)/2011-02-19
RESULT: I     PRIMARY=      5 INSERTED=      5 (    1)  DELETED=      - (    -)
            SECONDARY=      -     SAME=      - (    -) ————————————————————— (7)
```

(5) The current comparison statistics, contained in area C0, are output. LMS then performs the following internal actions:

- Area C0 is added to area C1.
- Area C0 is cleared again so as to accommodate the result of the next comparison.

(6) All M-type members of library LIB.ALL.V2 are compared with the members of macro library LIB.MAC.V1.

(7) The result of the comparison is output.

```
//show-statistics number=*c0 ------------------------------------------------------------ (8)
AREA C0
                PRIM.     PRIM.      INS.      SAME       DEL.  INS+DEL      SEC.    SEC.
STATISTIC       ELEM.     LINES     LINES     LINES     LINES     LINES     LINES   ELEM.
S (SAME)            2        10         -        10         -         -        10       2
C (CHANGED)         0         0         0         0         0         0         0       0
I (INSERTED)        2        10        10         -         -        10         -       -
D (DELETED)         -         -         -         -         0         0         0       0
-------------------------------------------------------------------------------------------
   TOTAL            4        20        10        10         0        10        10       2
//show-statistics number=*c1 ------------------------------------------------------------ (9)
AREA C1
                PRIM.     PRIM.      INS.      SAME       DEL.  INS+DEL      SEC.    SEC.
STATISTIC       ELEM.     LINES     LINES     LINES     LINES     LINES     LINES   ELEM.
S (SAME)            5        37         -        37         -         -        37       5
C (CHANGED)         0         0         0         0         0         0         0       0
I (INSERTED)        2        10        10         -         -        10         -       -
D (DELETED)         -         -         -         -         0         0         0       0
-------------------------------------------------------------------------------------------
   TOTAL            7        47        10        37         0        10        37       5
//end ---------------------------------------------------------------------------------- (10)
```

(8) The current comparison statistics, again contained in area C0, are output.

(9) Area C1 is output. This contains the overall statistics for all comparisons performed thus far in the present LMS run.

(10) LMS is terminated.

9.8 Branching to a user program while a member is being listed

The user program lists only the first 10 input records of a member.

If a member consists of less than 10 records, the program pads it out with additional records of its own to bring it up to 10.

```
/start-lms ——————————————————————————————————————————————————————————— (1)
//modify-logging-parameters logging=*maximum ———————————————————————————— (2)
//open-library library=use.lib,mode=*update ———————————————————————————— (3)
//show-element (element=uselst,type=s) ————————————————————————————————— (4)
INPUT  LIBRARY= :1OSQ:$USER.USE.LIB
INPUT  ELEMENT= (S)USELST/@(0001)/2011-02-19
*        TITLE 'USEREXIT FOR THE FUNCTION: LST'
*
*              1.) BY CONNECTING WITH THIS UNDERPROGRAM ONLY THE
*                  FIRST 10 RECORDS PER ELEMENT ARE LISTED.
*
*              2.) IF THE ELEMENT HAS LESS THAN 10 RECORDS,
*                  FURTHER RECORDS ARE INSERTED.
* INPUT FROM LMS: R1=A(PARAMETER LIST)
*                 R13=A(SAVEAREA), 18 WORDS
*                 R14=RETURN ADDRESS
*                 R15=A(USERPROGRAM)
*
PARDSEC  DSECT
AUFTRAG  DS    A              A(TASK FROM LMS)
*                             - 'BOE':START OF ELEMENT
*                             - 'REC':RECORD ORDERED
*                             - 'EOE':END OF ELEMENT
ANTWORT  DS    A              A(ANSWER FROM USERPROGRAM)
*                             - 'CON':CONTINUE
*                             - 'DEL':DELETE RECORD
*                             - 'INS':INSERT NEW RECORD
SATZ     DS    A              A(RECORD, INCL. 4 BYTE HEADER)
*                             -
```

(1) LMS is called.

(2) All messages and statements are logged.

(3) Library USE.LIB is assigned.

(4) User source program USELST is listed.

```
PARDSECL EQU   *-PARDSEC       L'DSECT
USELST   CSECT PAGE
         STM   0,15,0(13)      SAVE REGISTERS
         LR    10,15           BASE
         USING PARDSEC,1       LMS PARAMETERLIST
         USING USELST,10
         L     6,AUFTRAG       A(TASK)
         L     7,ANTWORT       A(ANSWER)
         L     8,SATZ          A(RECORD)
         CLC   0(3,6),REC      RECORD ORDERED?
         BE    DOSATZ          YES ---?
         CLC   0(3,6),BOE      START OF ELEMENT
         BE    DOBOE           YES ---?
         CLC   0(3,6),EOE      END OF ELEMENT
         BE    DOEOE           YES ---?
         B     RETURN
*
DOBOE    EQU   *
         ZAP   ANZAHL,P0       COUNTER := 0
         B     RETURN
*
DOSATZ   EQU   *
         CP    ANZAHL,P10      ALREADY 10 RECORDS LISTED?
         BNL   DODEL           YES (REST IGNORE) ---?
         AP    ANZAHL,P1       COUNTER := COUNTER +1
         B     DOCON
*
DOEOE    EQU   *
         CP    ANZAHL,P10      ALREADY 10 RECORDS LISTED?
         BNL   DOCON           YES (NO INSERT) ---?
         AP    ANZAHL,P1       COUNTER := COUNTER +1
         B     DOINS
*
DOINS    EQU   *
         MVC   0(3,7),INS      INSERT RECORD
         LA    9,INSSATZ
         ST    9,SATZ          A(RECORD TO BE INSERTED)
         B     RETURN
*
DODEL    EQU   *
         MVC   0(3,7),DEL      DELETE RECORD
         B     RETURN
*
DOCON    EQU   *
         MVC   0(3,7),CON      CONTINUE
*
```

```
RETURN   EQU   *
         LM    0,15,0(13)      RESTORE REGISTERS
         BR    14
         TITLE 'KONSTANTEN UND VARIABLE'
BOE      DC    'BOE'           START OF ELEMENT
REC      DC    'REC'           RECORD ORDERED
EOE      DC    'EOE'           END OF ELEMENT
CON      DC    'CON'           CONTINUE
DEL      DC    'DEL'           DELETE RECORD
INS      DC    'INS'           INSERT NEW RECORD
ANZAHL   DC    PL2'0'
P0       DC    PL2'0'
P1       DC    PL2'1'
P10      DC    PL2'10'
INSSATZ  DC    Y(INSSATZE-INSSATZ)
         DC    XL2'4040'
         DC    '**************** INSERT BY USER-PROGRAM  *************'
INSSATZE EQU   *
         LTORG
         END
NUMBER OF PROCESSED RECORDS IS      89
//activate-user-exit function=*show-elem,entry=uselst,lib=use.lib ————————————   (5)
//show-element (element=einaus,type=s) ———————————————————————————————————————   (6)
INPUT  LIBRARY= :1OSQ:$USER.USE.LIB
INPUT  ELEMENT= (S)EINAUS/@(0001)/2011-02-19
USER EXIT USELST IN USE.LIB IS ACTIVE
   #286  >          TITLE 'SEIZE OF DATES'
   #287  >          PRINT NOGEN
   #288  >  ERFAS   START
   #289  >          BALR  5,0
   #290  >          USING *,5
   #291  >          OPEN  DATEI,OUTPUT
   #292  >  LESEN   RDATA SATZ,ENDPGM
   #293  >          CLC   TEXT(4),=C'/EOF'
   #294  >          BE    ENDPGM
   #295  >          MVC   ATEXT,TEXT
NUMBER OF PROCESSED RECORDS IS     10
```

(5) Before listing an input record LMS branches to user program USELST, which resides in library USE.LIB.

(6) The first 10 records of member EINAUS of the assigned library USE.LIB are listed.

```
//show-user-exits ──────────────────────────────────────────────────────────── (7)
FUNCTION  ENTRY    LIBRARY                                                 INT
SHOW      USELST   USE.LIB                                                V1
//show-element (element=persdat,type=s) ──────────────────────────────────── (8)
INPUT  LIBRARY= :1OSQ:$USER.USE.LIB
INPUT  ELEMENT= (S)PERSDAT/@(0001)/2011-02-19
USER EXIT USELST IN USE.LIB IS ACTIVE
  #1113  >  BACH          SEBASTIAN      MUENCHEN       AUF DER HOEHE 7     AB 3
  #1114  >  BERGMANN      NORBERT        MUENCHEN       TORWEG 10           AB 5
  #1115  >  FINK          SUSANNE        NUERNBERG      RINGSTR. 23         AB 1
  #1116  >  MEYER         FRANZ          NUERNBERG      WASSERMUNGENWEG     AB 1
  #1117  >  GRUNDLER      WOLFGANG       BASEL          SONNENSTR. 11       AB 2
  #1118  >  KNOLL         MONIKA         FRANKFURT      BAUMALLEE 12        AB 3
  #1119  >  LIEDL         ERIKA          MUENCHEN       IN DER BREITE 1     AB 5
  #1120  >  WAGNER        JOHANN         AUGSBURG       AM SEE 45           AB 4
**************** INSERT BY USER-PROGRAM  *********
**************** INSERT BY USER-PROGRAM  *********
NUMBER OF PROCESSED RECORDS IS      10
//end ──────────────────────────────────────────────────────────────────────── (9)
```

(7) The active user exits are displayed.

(8) Member PERSDAT is listed. Since it is shorter than 10 records it is padded with records by the user program.

(9) LMS is terminated.

9.9 Granting and displaying protection attributes

Specific protection attributes are to be granted for a library and for certain members in this library.

```
/start-lms
//modify-logging-parameters logging=*maximum ————————————————————————————  (1)
//open-library library=bsp9.bib,mode=*update ————————————————————————————  (2)
LIBRARY IS CLEARED AND PREPARED
//modify-library-attributes administration=(user=*owner), -
//             init-elem-protection=(read=(user=*owner), -
//                                   write=(user=*owner,password=´P´),-
//                                   exec=(user=*owner)) ——————————————————  (3)
//show-library-attributes ——————————————————————————————————————————————  (4)
INPUT  LIBRARY= :1OSQ:$USER.BSP9.BIB
READ-PASS  = *NONE        READ-USER  = *OWNER   -     -
WR-PASS    = *YES         WR-USER    = *OWNER   -     -
EXEC-PASS  = *NONE        EXEC-USER  = *OWNER   -     -
ADMIN-PASS = *NONE        ADMIN-USER = *OWNER   -     -
FILE-SIZE  = 12           FREE-SIZE  = 4           FORMAT = NK2   UPAM-PROT = N
ACCESS-DATE= *NONE        WR-CONTROL = *NONE       STORAGE=*NONE
//modify-type-attributes type=d, -
//          init-elem-protection=(read=(user=(*owner,*group)), -
//                                write=(user=*owner)) ———————————————————  (5)
```

(1) In addition to error messages, positive acknowledgments are also logged.

(2) Library BSP9.BIB is opened for reading and writing.

(3) Administration authorization is granted for the library owner ’USER’. Only he/she may create, delete or rename members. Initial member protection is set. For all member types for which nothing more specific is set, members are created with this protection.

(4) Output the library attributes.

(5) A specific initial member protection is set for type D: read authorization is granted to the owner and the group, write authorization is granted to the owner of the library.

```
//add-element to-element=(element=test,type=s) ————————————————————————— (6)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP9.BIB
OUTPUT ELEMENT= (S)TEST/@(0001)/2013-03-01
         ADD (S)TEST/@(0001)/2013-03-01
//add-element to-element=(element=test,type=d) ————————————————————————— (7)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP9.BIB
OUTPUT ELEMENT= (D)TEST/@(0001)/2013-03-01
         ADD (D)TEST/@(0001)/2013-03-01
//copy-element element=(element=test,type=s), -
//             to-element=(element=test2,type=s) ————————————————————————— (8)
INPUT  LIBRARY= :1OSQ:$USER.BSP9.BIB
OUTPUT LIBRARY= :1OSQ:$USER.BSP9.BIB
         COPY (S)TEST/@(0001)/2013-03-01 AS (S)TEST2/@(0001)/2013-03-01
//modify-element-protection element=(element=test2,type=s), -
//                          new-protection=(read=(user=*all)) ——————————————— (9)
OUTPUT LIBRARY= :1OSQ:$USER.BSP9.BIB
         MODIFY (S)TEST2/@(0001)/2013-03-01
```

(6) An S-type member having the name TEST is created. The input is performed via *SYSDTA. The member automatically receives the protection applying to the library (see (3)).

(7) A further member is created under the same name but as type D. The input for this member is also performed via *SYSDTA. The member receives the protection defined for all D-type members (see (5)).

(8) S-type member TEST is copied. The new member is to be called TEST2. The type remains the same.

(9) Member TEST2 receives new protection; read authorization is extended to all users.

```
//show-element-attributes element=(element=*(version=*)), -
//                        information=*maximum ──────────────────────────────── (10)
INPUT  LIBRARY= :1OSQ:$USER.BSP9.BIB
TYPE       = D
NAME       = TEST
VERSION    = @                                          VARIANT    = 0001
USER-DATE  = 2013-03-01   CRE-DATE   = 2013-03-01   MOD-DATE   = 2013-03-01
USER-TIME  =   10:33:06   CRE-TIME   =   10:33:06   MOD-TIME   =   10:33:06
STORAGE    = *FULL
STATE      = *FREE
ELEM-SIZE  = 1
READ-PASS  = *NONE        READ-USER  = *OWNER *GROUP  -
WR-PASS    = *NONE        WR-USER    = *OWNER   -     -

       1 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYPE       = S
NAME       = TEST
VERSION    = @                                          VARIANT    = 0001
USER-DATE  = 2013-03-01   CRE-DATE   = 2013-03-01   MOD-DATE   = 2013-03-01
USER-TIME  =   10:33:06   CRE-TIME   =   10:33:06   MOD-TIME   =   10:33:06
STORAGE    = *FULL
STATE      = *FREE
ELEM-SIZE  = 1
READ-PASS  = *NONE        READ-USER  = *OWNER   -     -
WR-PASS    = *YES         WR-USER    = *OWNER   -     -
EXEC-PASS  = *NONE        EXEC-USER  = *OWNER   -     -

TYPE       = S
NAME       = TEST2
VERSION    = @                                          VARIANT    = 0001
USER-DATE  = 2013-03-01   CRE-DATE   = 2013-03-01   MOD-DATE   = 2013-03-01
USER-TIME  =   10:33:06   CRE-TIME   =   10:33:06   MOD-TIME   =   10:33:06
STORAGE    = *FULL
STATE      = *FREE
ELEM-SIZE  = 1
READ-PASS  = *NONE        READ-USER  = *OWNER *GROUP *OTHERS
WR-PASS    = *YES         WR-USER    = *OWNER   -     -
EXEC-PASS  = *NONE        EXEC-USER  = *OWNER   -     -
       2 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
──────────────────────────────────────────────────────────────────────────────

       3 ELEMENT(S) IN THIS TABLE OF CONTENTS
//end ──────────────────────────────────────────────────────────────────────── (11)
```

(10) All members are displayed with their protection attributes.

(11) The LMS run is terminated.

9.10 Automatic version incrementation with convention NONE

Member “test” is always added to the library under the same name, but with a different method of inclusion.

```
/start-lms
//modify-logging-parameters logging=*maximum ————————————————————————— (1)
//open-library library=bsp10.bib,mode=*update ———————————————————————— (2)
LIBRARY IS CLEARED AND PREPARED
//show-type-attributes type=s ————————————————————————————————————— (3)
INPUT  LIBRARY= :1OSQ:$USER.BSP10.BIB
TYPE       = S
SUPER-TYPE = *NONE
BASE-TYPE  = S
CONVENTION = *NONE
INIT-ELEM-P= *NONE
ADMINISTRAT= *NONE
STORAGE    = *NONE         WR-CONTROL = *NONE
//add-element to-element=(element=test(version=*increment),type=s) ———————— (4)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP10.BIB
OUTPUT ELEMENT= (S)TEST/001(0001)/2013-03-01
         ADD (S)TEST/001(0001)/2013-03-01
```

(1) In addition to error messages, positive acknowledgments are also logged.

(2) Library BSP10.BIB is opened for reading and writing; no S-type member is present.

(3) Display type attributes for member type S as a check. Member type S has the convention NONE.

(4) Create the first member via *SYSDTA. The member is created under type S with the name “test” through automatic version incrementation.

```
//add-element to-element=(element=test(version=*increment),type=s) ————————————— (5)
input2
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/002(0001)/2013-03-01
         ADD (S)TEST/002(0001)/2013-03-01
//add-element to-element=(element=test(version=a001),type=s) ———————————————————— (6)
input3
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/A001(0001)/2013-03-01
         ADD (S)TEST/A001(0001)/2013-03-01
//add-element to-element=(element=test(version=*increment,base=a*),type=s) ———————— (7)
input4
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/A002(0001)/2013-03-01
         ADD (S)TEST/A002(0001)/2013-03-01
//add-element to-elem=(elem=test(version=*highest-existing),type=s), -
//            write-mode=*any ———————————————————————————————————————————————— (8)
input5
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/002(0002)/2013-03-01
         ADD (S)TEST/002(0002)/2013-03-01 , OUTPUT REPLACED
//show-element-attributes ———————————————————————————————————————————————————— (9)
INPUT  LIBRARY= :1OSQ:$USER.BSP10.BIB
TYP NAME VER  (VAR#) DATE         NAME VER  (VAR#) DATE
(S) TEST A001 (0001) 2013-03-01   TEST A002 (0001) 2013-03-01
(S) TEST 001  (0001) 2013-03-01   TEST 002  (0002) 2013-03-01
       4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//end ————————————————————————————————————————————————————————————————————————— (10)
```

(5) Maintain the sequence with *INCREMENT, i.e. a second member with the same name and type but with a version incremented by one is created.

(6) Explicit creation of version A001.

(7) Create a further member through automatic version incrementation; the member is based on the version with prefix A*.

(8) Overwrite the highest version with type S and name “test”.

(9) Directory of library BSP10.BIB.

(10) Terminate the LMS run.

9.11 Automatic version incrementation with convention STD-SEQUENCE

Members are to be added to a library in a preset version format, whereby the version is to be automatically incremented.

```
/start-lms
//modify-logging-parameters logging=*maximum ————————————————————————— (1)
//open-library library=bsp11.bib,mode=*update ———————————————————————— (2)
LIBRARY IS CLEARED AND PREPARED
//modify-type-attributes type=s,convention=*std-sequence(example=v001) ———————— (3)
//show-type-attributes type=s ————————————————————————————————————— (4)
INPUT  LIBRARY= :1OSQ:$USER.BSP11.BIB
TYPE       = S
SUPER-TYPE = *NONE
BASE-TYPE  = S
CONVENTION = *STD-SEQUENCE
EXAMPLE    = V001
INIT-ELEM-P= *NONE
ADMINISTRAT= *NONE
STORAGE    = *NONE          WR-CONTROL = *NONE
//add to-element=library-element(element=test(version=*increment),type=s) ———————— (5)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP11.BIB
OUTPUT ELEMENT= (S)TEST/V001(0001)/2013-03-01
         ADD (S)TEST/V001(0001)/2013-03-01
```

(1) In addition to error messages, positive acknowledgments are also logged.

(2) Library BSP11.BIB is opened for reading and writing; no S-type member is present.

(3) Set up the convention STD-SEQUENCE for type S with the example given in EXAMPLE for the version format.

(4) Display type attributes for member type S as a check.

(5) Create the first member via *SYSDTA. The member is created under type S with the name “test” through automatic version incrementation.

```
//add-element to-element=(element=test(version=*increment),type=s) ————————————— (6)
input1
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/V002(0001)/2013-03-01
         ADD (S)TEST/V002(0001)/2013-03-01
//add to-element=library-element(element=test(version=w001),type=s) ————————————— (7)
input1
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/W001(0001)/2013-03-01
         ADD (S)TEST/W001(0001)/2013-03-01
//add to-element=library-element(element=test(version=999),type=s) ————————————— (8)
input1
*END
OUTPUT ELEMENT= (S)TEST/999/2013-03-01
%  LMS0024 ERROR ON LIBRARY ':1OSQ:$USER.BSP11.BIB', *** PLAM ERROR CODE '0476'.
%  PLA0476 VERSION OR PREFIX NOT ACCORDING TO EFFECTIVE CONVENTION
      NO ADD (S)TEST/999/2013-03-01 , ERROR OCCURRED
SKIPPED:input1
//add to-element=(element=test(version=*highest-existing),type=s), -
//    write-mode=*any ———————————————————————————————————————————————————— (9)
input1
*END
INPUT  SYSDTA
OUTPUT ELEMENT= (S)TEST/W001(0002)/2013-03-01
         ADD (S)TEST/W001(0002)/2013-03-01 , OUTPUT REPLACED
```

(6) Maintain the sequence with *INCREMENT, i.e. a second member with the same name and type but with a version incremented by one is created.

(7) Maintain the sequence on changing version, i.e. a third member is created with the same name and type but, through explicit specification of the version, with version W001.

(8) An attempt is made to add a further member that does not have a version format corresponding to the convention STD-SEQUENCE. This statement is rejected by LMS.

(9) Overwrite the highest version with type S and name “test”.

```
//show-element-attributes ------------------------------------------------------------ (10)
INPUT  LIBRARY= :1OSQ:$USER.BSP11.BIB
TYP NAME VER  (VAR#) DATE         NAME VER  (VAR#) DATE
(S) TEST V001 (0001) 2013-03-01   TEST V002 (0001) 2013-03-01
(S) TEST W001 (0002) 2013-03-01
       3 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//end -------------------------------------------------------------------------------- (11)
```

(10) Overview of members present.

(11) Terminate the LMS run.

9.12 Automatic version incrementation with convention STD-TREE

```
/start-lms
//modify-logging-parameters logging=*maximum ——————————————————————————— (1)
//open-library library=bsp12.bib,mode=*update ———————————————————————— (2)
LIBRARY IS CLEARED AND PREPARED
//modify-type-attributes type=s,convention=*std-tree ———————————————————— (3)
//show-type-attributes type=s ——————————————————————————————————————— (4)
INPUT  LIBRARY= :1OSQ:$USER.BSP12.BIB
TYPE       = S
SUPER-TYPE = *NONE
BASE-TYPE  = S
CONVENTION = *STD-TREE
INIT-ELEM-P= *NONE
ADMINISTRAT= *NONE
STORAGE    = *NONE          WR-CONTROL = *NONE
//add to-element=library-element(element=test(version=*increment),type=s) ———— (5)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP12.BIB
OUTPUT ELEMENT= (S)TEST/001.001(0001)/2013-03-01
         ADD (S)TEST/001.001(0001)/2013-03-01
```

(1) In addition to error messages, positive acknowledgments are also logged.

(2) Library BSP12.BIB is opened for reading and writing; no S-type member is present.

(3) Set up the convention STD-TREE for member type S.

(4) Display type attributes for member type S as a check.

(5) Create the first member via *SYSDTA. The member is created under type S with the name “test” through automatic version incrementation.

```
//add to-element=library-element(element=test(version=*increment),type=s) ————————  (6)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP12.BIB
OUTPUT ELEMENT= (S)TEST/001.002(0001)/2013-03-01
         ADD (S)TEST/001.002(0001)/2013-03-01
//add to-element=(element=test(version=*increment,base=1.1),type=s) ————————————————  (7)
*END

INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP12.BIB
OUTPUT ELEMENT= (S)TEST/001.001.001.001(0001)/2013-03-01
%  LMS0095 INPUT DATA RECORDS MISSING
          ADD (S)TEST/001.001.001.001(0001)/2013-03-01
//add-element to-element=library-element(element=test(version=2.1),type=s ————————  (8)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP12.BIB
OUTPUT ELEMENT= (S)TEST/002.001(0001)/2013-03-01
         ADD (S)TEST/002.001(0001)/2013-03-01
//add to-element=(element=test(version=*increment,base=1.1.1.*),type=s) ————————————  (9)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP12.BIB
OUTPUT ELEMENT= (S)TEST/001.001.001.002(0001)/2013-03-01
         ADD (S)TEST/001.001.001.002(0001)/2013-03-01
//add to-element=(element=test(version=*highest-existing),type=s), -
//    write-mode=*any ——————————————————————————————————————————————————————————  (10)
input1
*END
INPUT  SYSDTA
OUTPUT LIBRARY= :1OSQ:$USER.BSP12.BIB
OUTPUT ELEMENT= (S)TEST/002.001(0002)/2013-03-01
         ADD (S)TEST/002.001(0002)/2013-03-01 , OUTPUT REPLACED
```

(6) Maintain the sequence with *INCREMENT, i.e. a second member with the same name and type but with a version incremented by one is created.

(7) Open a side branch on version 1.1

(8) Create a member with the same type and name, but with an explicitly specified version.

(9) Maintain the side branch with prefix 1.1.1.

(10) Overwrite the highest version with type A and name “test”.

```
//show-element-attributes ------------------------------------------------------------ (11)
INPUT  LIBRARY= :1OSQ:$USER.BSP12.BIB
TYP NAME VERSION         (VAR#) DATE
(S) TEST 001.001 . . . . (0001) 2013-03-01
(S) TEST 001.001.001.001 (0001) 2013-03-01
(S) TEST 001.001.001.002 (0001) 2013-03-01
(S) TEST 001.002 . . . . (0001) 2013-03-01
(S) TEST 002.001 . . . . (0002) 2013-03-01
       5 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//end ------------------------------------------------------------------------------- (12)
```

(11) Output directory of library BSP12.BIB. The library contains five members with the same name and type but with differing version and variant numbers.

(12) Terminate the LMS run.

9.13 make run

The following example of the make functionality consists of several procedures.

The procedure MAINPROCEDURE has the following parameters:

TARGET:	Target component of the make runs
SUCC-PROC:	Continuation processing
SELECT:	Consideration of the time stamp
PROCEDURE:	Generated procedure
MAKEFILE:	Description of the program system itself
MAKEDEFAULTS:	Description of the global defaults

The parameters TARGET, SUCC-PROC, SELECT and PROCEDURE are to be specified in the same way as in BEGIN-MAKE. If the procedure is called with the default parameters, the target component of the make run is the first target in MAKEDEFAULTS or MAKEFILE. The make run generates a procedure which is then called with INCLUDE-PROCEDURE.

The MAKEDEFAULTS file contains default settings and standard actions, i.e. things that can be shared by a number of program systems.

The MAKEFILE file contains the actual definitions of the program system, the dependencies.
The program system contains a phase file which is linked from two object modules. The object modules are compiled from S-type members of the same names with the aid of two M-type members. The pseudo-targets CLEAR and PRINT can be used for special program system actions (clearing and printing).

MAINPROCEDURE procedure:

```
/DECLARE-PARAMETER (TARGET(INI-VALUE=´*FIRST-TARGET´),-
/                   SUCC-PROC(INI-VALUE=´*CREATE-PROC´),-
/                   SELECT(INI-VALUE=´*MODIFIED´),-
/                   PROCEDURE(INI-VALUE=´#P´),-
/                   MAKEFILE(INI-VALUE=´MAKEFILE´),-
/                   MAKEDEFAULTS(INI-VALUE=´MAKEDEFAULTS´))
/SET-VAR SUBO=SUBSTR(´&(PROCEDURE)´,1,2)   „FILE OR BIB.-ELEMENT“ ——————————   (1)
/BEGIN-BLOCK DATA-INSERTION=YES
/START-LMS
```

(1) Use of the /INCLUDE-PROCEDURE command is permitted.

```
/IF (SUBO=´*L´)
// MODIFY-LMS-DEFAULTS TYPE=J
// DELETE-ELEMENT &(PROCEDURE)    „BIB.-ELEMENT DELETE“
// STEP
/ELSE
// EXEC-SYSTEM-CMD DELETE-FILE &(PROCEDURE)  „FILE DELETE“
// STEP
/END-IF
//BEGIN-MAKE TARGET=&(TARGET), - -------------------------------------------------  (2)
//           SELECT=&(SELECT), -
//           SUCCESS-PROCESSING=&(SUCC-PROC), -
//           PROCEDURE=&(PROCEDURE)
/INCLUDE-PROCEDURE NAME=&MAKEDEFAULTS ------------------------------------------  (3)
/INCLUDE-PROCEDURE NAME=&MAKEFILE
/END-BLOCK
//END-MAKE ----------------------------------------------------------------------  (4)
//END
/IF (SUBSTR(´&(SUCC-PROC)´,2,2) <> ´CR´)  „NO CREATE-PROC“
/   EXIT-PROC  „SUBSEQUENT PROCESSING ALREADY EXECUTED“
/END-IF
/INC-PROC &(PROCEDURE)  „EXECUTE, MAYBE NOT EXISTING“
/SET-JOB-STEP
/EXIT-PROCEDURE
```

(2) The make run is started.

(3) MAKEDEFAULTS and MAKEFILE are called.

(4) The sequence of make substatements is concluded and so also the make run. Continuation processing as specified in the BEGIN-MAKE statement (in this case the default INCLUDE-PROCEDURE) is initiated.

MAKEDEFAULTS file:

```
/RESUME-PROGRAM
//MODIFY-MAKE-DEFAULTS LIBRARY=BSPLIB,-
//   CURRENT-TARGET-VAR=CURT,FROM-OBJECTS-VAR=ALLOBJ ————————————————————————  (5)
//SET-STD-ACTION - ————————————————————————————————————————————————————————  (6)
//   TARGET-TYPE=R,FROM-TYPE=S,-
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,COMPILE),(&&(CURT.ELEMENT))'
/HOLD-PROGRAM ————————————————————————————————————————————————————————————  (7)
/EXIT-PROC
```

(5) The default library and make variables are set. These settings are valid for the subsequent statements.

(6) A standard action (compilation) is set for the transition from member type S to member type R.

(7) Preparation for branching back out of the procedure.

MAKEFILE file:

```
/RESUME-PROGRAM
//SET-DEPENDENCY - ------------------------------------------------------------  (8)
//   TARGET-OBJECT=PROG, -
//   FROM-OBJECT=(*LIB(,TEIL1,R),*LIB(,TEIL2,R)), -
//   ACTION='/CALL-PROCEDURE LINK,(OBJVAR=ALLOBJ)'
//SET-DEPENDENCY - ------------------------------------------------------------  (9)
//   TARGET-OBJECT=*LIB(,*,R),-
//   FROM-OBJECT=(*LIB(,*,S), *LIB(,GLOBALDATA,M), *LIB(,HEADERS,M)) ------------  (10)
//SET-DEPENDENCY - ------------------------------------------------------------  (11)
//   TARGET-OBJECT=CLEAR,-
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,DELETE.TEMPS)'
//SET-DEPENDENCY - ------------------------------------------------------------  (12)
//   TARGET-OBJECT=PRINT,-
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,PRINT.LISTFILES)'
//SET-PREPROCESSING - ---------------------------------------------------------  (13)
//   ACTION=('/CALL-PROCEDURE *LIB(BSPLIB,INIT)',-
//           '/DECL-VAR ALLOBJ(TYPE=STRING),MULT-ELEM=LIST,SCOPE=TASK')
//SET-POSTPROCESSING - --------------------------------------------------------  (14)
//   ACTION='/CALL-PROCEDURE *LIB(BSPLIB,STOP)'
/HOLD-PROGRAM
/EXIT-PROC
```

(8) Definition of the dependency “Phase is linked from object modules”. The phase file PROG is the first target of the make run (*FIRST-TARGET).

(9) Definition of the dependency “Object module generated from source of same name; further components (type M) required.” The standard action specified in (3) is used.

(10) “Of the same name” is formulated through the use of make selection and construction. The selection is specified at TARGET-OBJECT, and the construction at FROM-OBJECT. The LMS restriction that the construction must contain at least one wildcard symbol does not apply to make construction specifications. *LIB(,HEADERS,M) contains no wildcard symbol.

(11) The target file CLEAR is not dependent on any FROM-OBJECT, i.e. is never current. If CLEAR is specified as the target of the make run, clearing actions are initiated for the program system. The CLEAR actions should not be executed every time the program system is updated.

(12) As in (11). The printing of results is defined as the action for PRINT.

(13) When a procedure is generated, the ´/CALL INIT´ action is to be placed at its beginning. In addition, the variable ALLOBJ is to be defined globally for the task since it is to be imported into the LINK procedure later. The declaration is not overwritten during execution of the generated procedure.

(14) When a procedure is generated, the ´/CALL STOP´ action is to be placed at its end.

Written in a fashion similar to that of UNIX-make, the MAKEFILE file could look like this:

MAKEFILE file:

```
/RESUME-PROGRAM
//SET-DEPENDENCY PROG, (*LIB(,TEIL1,R),*LIB(,TEIL2,R)), -
//   '/CALL-PROCEDURE *LIB(BSPLIB,LINK),(OBJVAR=ALLOBJ)'
//SET-DEPENDENCY *LIB(,*,R),(*LIB(,*,S),*LIB(,GLOBALDATA,M),*LIB(,HEADERS,M))
//SET-DEPENDENCY CLEAR, *NONE,-
//   '/CALL-PROCEDURE *LIB(BSPLIB,DELETE.TEMPS)'
//SET-DEPENDENCY PRINT, *NONE,-
//   '/CALL-PROCEDURE *LIB(BSPLIB,PRINT.LISTFILES)'
//SET-PREPROCESSING ('/CALL-PROCEDURE *LIB(BSPLIB,INIT)',-
//   '/DECL-VAR ALLOBJ(TYPE=STRUC),MULT-ELEM=LIST,SCOPE=TASK')
//SET-POSTPROCESSING '/CALL-PROCEDURE *LIB(BSPLIB,STOP)'
/HOLD-PROGRAM
/EXIT-PROC
```

If it is necessary to generate the program system in its entirety, the required procedure would look something like this:

```
/DECL-VAR SYSLMSMAKE(TYPE=STRUC(DEF=*DYN)) ------------------------------------------  (1)
/CALL-PROCEDURE *LIB(BSPLIB,INIT)
/DECL-VAR ALLOBJ(TYPE=STRUC),MULT-ELEM=*LIST,SCOPE=*TASK
/IF-BLOCK-ERROR ---------------------------------------------------------------------  (2)
/EXIT-PROCEDURE
/END-IF
/DECLARE-VARIABLE CURT(TYPE=*STRUC) -------------------------------------------------  (3)
/IF-BLOCK-ERROR
/END-IF
...
/SET-VARIABLE CURT.ELEM = 'TEIL1' ---------------------------------------------------  (4)
...
/CALL-PROCEDURE *LIB(BSPLIB,COMPILE),(&(CURT.ELEM))
...
/SET-VARIABLE CURT.ELEM = 'TEIL2'
...
/CALL-PROCEDURE *LIB(BSPLIB,COMPILE),(&(CURT.ELEM))
...
/DECLARE-VARIABLE ALLOBJ(TYPE=*STRUC),MULT-ELEM=*LIST -------------------------------  (5)
/IF-BLOCK-ERROR
/END-IF
/FREE-VARIABLE ALLOBJ
...
/SET-VARIABLE SYSLMSMAKE.ELEM = 'TEIL1'
...
/SET-VARIABLE ALLOBJ = SYSLMSMAKE,MODE=APPEND ---------------------------------------  (6)
/SET-VARIABLE SYSLMSMAKE.ELEM = 'TEIL2'
...
/SET-VARIABLE ALLOBJ = SYSLMSMAKE,MODE=APPEND
/CALL-PROCEDURE *LIB(BSPLIB,LINK),(OBJVAR=ALLOBJ)
...
/CALL-PROCEDURE *LIB(BSPLIB,STOP)
```

(1) The make-internal auxiliary variables for list creation are declared.

(2) This sequence of statements is inserted after each action (except POSTPROCESSING) to ensure clean error exiting.

(3) CURT is declared (errors are tolerated since CURT can be declared, for example, globally).

(4) The other components of the S variables and the S list variable ALLOBJ are also set.

(5) An empty S list variable for ALLOBJ is declared. ALLOBJ is also handled before the other procedure calls, but its result is not used in them.

(6) The FROM-OBJECTS are appended to the S list variable ALLOBJ.

The procedures called in MAKEFILE have the following interfaces:

1. INIT and STOP have no parameters.
2. COMPILE has only one parameter, i.e. a member name. For the sake of simplicity, no library specification or other information is entered.
3. LINK allows the entry of the name of an S list variable containing all the objects that are to be linked. The list variable is handled as follows:

```
/DECL-PAR NAME=INPUTVAR(TYPE=STRING)
/DECL-VAR I(TYPE=STRUC(DEF=*DYN))
/IMPORT-VAR &(INPUTVAR) ------------------------------------------------------------  (1)
...
/FOR I=*LIST(&INPUTVAR) ------------------------------------------------------------  (2)
/    DO-SOMETHING ... &(I) ---------------------------------------------------------  (3)
/END-FOR
```

(1) The specified list variables (which must be defined globally for the task) are imported.

(2) The list variables are listed in variable I.

(3) Variable I is used either directly in SDF entries or in /SEND-DATA or /SEND-STATEMENT.

9.14 Using the output in S variables

This procedure compares two libraries.

All the members which exist in the newer library (NEW-LIB parameter) are deleted from the older library (OLD-LIB parameter).

```
/BEGIN-PARAMETER-DECLARATION
/DECLARE-PARAMETER NAME=OLD-LIB(INITIAL-VALUE=*PROMPT)
/DECLARE-PARAMETER NAME=NEW-LIB(INITIAL-VALUE=*PROMPT)
/END-PARAMETER-DECLARATION
/DECLARE-VARIABLE NAME=NEW-ELEMENTS(TYPE=*STRUC), -
/                 MULTIPLE-ELEMENTS=*LIST ------------------------------------------  (1)
/BEGIN-BLOCK DATA-INSERTION=*YES
/START-LMS
//SHOW-ELEM-ATTR  -
//       ELEM=*LIB-ELEM(LIB=&NEW-LIB,ELEM=*ALL,TYPE=*ALL),-
//       INF=*MIN,STRUCTURE-OUTPUT=NEW-ELEMENTS ---------------------------------  (2)
/DECLARE-VARIABLE NAME=LOOP(TYPE=*STRUC) -------------------------------------------  (3)
/FOR LOOP=*LIST(NEW-ELEMENTS)
//DELETE-ELEM  *LIB-ELEM(LIB=&OLD-LIB, -
//                       ELEM=&(LOOP.ELEM)(VERS=&(LOOP.VERSION)), -
//                       TYPE=&(LOOP.TYPE)) -----------------------------------------  (4)
//STEP   "if member in &OLD-LIB does not exist"
/END-FOR
//END
/END-BLOCK
```

(1) The variables for the output of //SHOW-ELEMENT-ATTRIBUTES are declared.

(2) The output of //SHOW-ELEMENT-ATTRIBUTES is placed in the variables.

(3) The loop variables are declared.

(4) The members in a loop are deleted; the values of the loop variables are used for member, version and type.

9.15 Library lists

The following example illustrates how library lists are embedded into a development environment.

When several developers are working on a shared database, it is usual for data to be stored locally as well as in the central data pool. Data is then transferred to and fro by means of the borrowing mechanism.

When compiling a local variant, it is a good idea to combine central and local data in a library list, so that the exact storage location does not have to be known for each library member.

The figure below shows the central library and a local library, both of which belong to the development environment. Source members have the member type S, while include members have the type M. The result is PROGRAM with type L. With the help of library lists, PROGRAM can be updated both centrally and locally with the same make file. The library list with which PROGRAM is updated is to receive the name SYSPLAMALT-PROGRAM (library lists must begin with "SYSPLAMALT-"). If used locally, SYSPLAMALT-PROGRAM must consist of two libraries, if used centrally only of one.

– locally: SYSPLAMALT-PROGRAM = '(LIB,$CENTRAL.LIB)'
  (first look locally, then under $CENTRAL)
– centrally: SYSPLAMALT-PROGRAM = '($CENTRAL.LIB)'
  (look only under $CENTRAL)

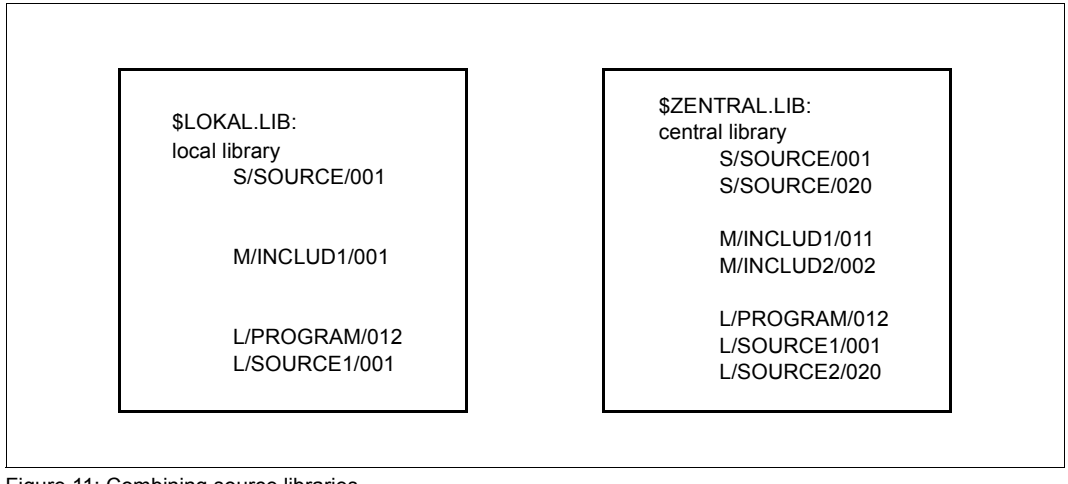


Figure 11: Combining source libraries

```
//begin-make
//modify-make-defaults library=sysplamalt-program,-
//       from-objects-var=from
```

```
//set-dependency *l(lib,program,l), "$LOCAL.LIB od. $CENTRAL.LIB"-
//                 (*l(,source1,l),-
//                  *l(,source2,l)),'/call bind'
//set-dependency *l(,source1,l),-
//                  (*l(,source1,s),-
//                  *l(,includ1,m),-
//                  *l(,includ2,m)),-
//                  '/call compile lib=&&(from#1.lib),elem=source1'
//set-dependency *l(,source2,l),-
//                  (*l(,source2,s),-
//                  *l(,includ2,m)),-
//                  '/call compile lib=&&(from#1.lib),elem=source2'
//end-make
```

In the BIND and COMPILE procedures, the library list is used in two different ways. BIND uses the library list on the input side directly:

```
...
//include-module element=source1,library=sysplamalt-program
//include-module element=source2,library=sysplamalt-program
...
```

With local use, the local library then has priority over the central library.

The COMPILE procedure has a LIB parameter which is supplied with the hit library via the make S variable FROM; the compiler then works directly with this value. Include processing operates with the library list SYSPLAMALT-PROGRAM. Local includes then have priority over central ones.

10 Appendix

10.1 Supplementary information in LMS messages

The following supplementary information appears in various combinations in addition to the actual text of the LMS messages:

Supplementary information	Meaning
FUNCTION TERMINATED	Function is aborted
(LIBRARY INPUT)	Library input
(LIBRARY OUTPUT)	Library output
(LISTING-MEMBER-OUTPUT)	Log output to member
STATEMENT(S) IS (ARE) SKIPPED	The entire statement is ignored
SYSLST (LISTING)	Log output to SYSLST
OUTPUT-LIBRARY LOCKED	The output library is locked
OPENERROR ON LIBRARY/ELEMENT	Error occurred when library/member was opened
OUTPUT-LIBRARY MISSING	The output library does not exist
RECFORM=F	The library member has records of fixed length
KEYPOS>5	The KEYPOS is greater than 5
KEYLEN>8	The KEYLEN is greater than 8 (applies only when calling EDT)
KEYLEN>16	The KEYLEN is greater than 16 (applies only when calling EDOR)
KEYS DO EXIST IN ELEMENT	The library member to be extended has been added using SOURCE-ATTRIBUTES=KEEP and contains the ISAM keys
DIFFERENT FILETYPE/ VALUE PROPAGATION (MIN/MAX)	The file type or the “VALUE PROPAGATION” of the file to be added does not match that of the library member to be extended
DIFFERENT RECORD FORMAT	The record format of the file to be added does not match the record format of the library member to be extended

Supplementary information	Meaning
DIFFERENT RECORD SIZE	The record length of the file to be added does not match the record length of the library member to be extended
DIFFERENT KEYPOSITION	The position of the keys of the file to be added does not match the position of the keys of the library member to extended
DIFFERENT KEYLENGTH	The length of the keys of the file to be added does not match the length of the keys of the library member to be extended
DIFFERENT LOGLENGTH	The LOGLENGTH of the file to be added does not match the LOGLENGTH of the library member to be extended
DIFFERENT VALUE LENGTH	The VALLEN value of the file to be added does not match the VALLEN of the library member to be extended
OUTPUT-LIBRARY IS NOT A PLAM-LIBRARY	The output library is not a program library
FIXED RECORD FORMAT ON INPUT-FILE	The library member to be extended contains no file attributes and the input file has fixed-length records
KEYPOSITION≠5 ON INPUT-FILE	The library member to be extended contains no file attributes and the input file has a key position which is not equal to 5
KEYLENGTH>16 ON INPUT-FILE	The library member to be extended contains no file attributes and the input file has a key length greater than 16
RECORD SIZE>2032 ON INPUT-FILE	The file to be added has a record length of more than 2032 bytes

10.2 Messages of the AMCB access routine

The messages issued by the internal LMS access routine AMCB have the following format:

```
{AMCB} : xxxx
```

where

xxxx is the AMCB error key.

xxxx	**Meaning of AMCB error codes**
0000	No error
0001	DMS error
0002	Illegal op code
0003	File name missing in control block
0004	No / modified FCB address, or FCB address for FOP with DMS-OPEN points to active FCB
0005	Incorrect op code sequence
0006	Library type invalid
0007	Contradictory LIB types in control block
0008	Library has been repaired
0009	Library must be repaired
0010	Address not within member limits
0011	On writing: library limit reached
0012	Contradictory information in control block and FCB
0013	Supplementary information missing
0014	Supplementary information missing
0015	Record too long
0017	Last member in library has been deleted
0019	This is not an LMS library
0022	Module not complete (e.g. no END record)
0023	Incorrect record type in module
0050	Overwrite error
0051	Insufficient memory

xxxx	Meaning of AMCB error codes
0052	Member has been overwritten
0053	Input file is empty
0054	Empty file is replaced
0062	Function not implemented
0063	No files present
0066	First record not an ESD record
0100	Illegal program file
0101	DMS error
0102	Unknown file type
0107	Neither file name nor link name entered
0108	User error
0109	Open error
0111	No free space in file table
0112	FSTAT error
0118	No empty file for CREATE
0119	Open for empty file
0120	File name invalid
0121	File ID has no entry in file table
0122	Requested open status different from actual status
0123	No further space for FCB
0124	No further space for access indicator in file table
0125	Second access to output library
0126	No further space for link table entry
0127	Link table entry missing
0131	Library still open for CTL or PRT
0136	Access error, e.g. file locked
0150	Access method not known

10.3 Old LMS subprogram interface

For compatibility reasons, the old subprogram interface described here is still supported.

The new subprogram interface is described in the manual “LMS Subroutine Interface” [1].

If LMS is called as a subprogram with independent dialog, control is returned to the program that issued the call after the END statement is processed. LMS remains loaded after the END statement is processed. In all other respects, the LMS run is the same as when the program is loaded by means of the /START-EXECUTABLE-PROGRAM command. Whenever LMS is called, it logs on its own STXIT routine.

When control is returned to the main program, LMS’s STXIT routine is logged off; the main program must then log its own STXIT routine on again.

When calling the subprogram interface, be sure to observe the following register conventions:

Register 1	must be zero.
Register 13	contains the address of an 18-word save area which must be made available by the calling program. LMS uses this area to store the register contents of the calling program.
Register 14	contains the return address.
Register 15	contains the entry address LMSUPSDF.

Before control is returned to the calling program, LMS stores the following return codes in register 15:

X'00'	LMS terminated normally.
X'04'	LMS terminated abnormally.

Example

LMS is called as a subprogram from the program UPROG. In LMS, a member is output from a program library to a file. After the LMS run has terminated, control is returned to the user program.

```
/START-LMS
//OPEN UEB.BIB,U ------------------------------------------------------------  (1)
//SHOW-ELEM *LIB(,LMSCALL,S) ------------------------------------------------  (2)
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)LMSCALL/(0001)/1995-08-12
UPROG    START
         BALR  3,0
         USING *,3
         MVC   OUTPUT,ANMELD
AUFRUF   WROUT OUT,TERM
         LA    14,RUECK ------------------------------------------------------  (3)
         LA    1,0 -----------------------------------------------------------  (4)
         LA    13,SAVE -------------------------------------------------------  (5)
         L     15,=V(LMSUPSDF) -----------------------------------------------  (6)
         BALR  14,15
RUECK    MVC   OUTPUT,ABMELD
         WROUT OUT,TERM
*
*
TERM     TERM
*
*
SAVE     DS    18F
*
ANMELD   DC    '********** *LMS ACTIVATED NOW ************'
ABMELD   DC    '***** LMS TERMINATED NOW - RETURNING TO PROGRAM *****'
OUT      DC    Y(ENDE-OUT)
         DS    CL2
         DC    X'01'
OUTPUT   DS    CL50
ENDE     EQU   *
*
*
         END   UPROG
NUMBER OF PROCESSED RECORDS IS      29
//END
/
.
.
.
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIB=UEB.BIB,ELEM=LMSCALL)
%  BLS0500 PROGRAM 'LMSCALL', VERSION '007' OF '95-08-12' LOADED
********** *LMS ACTIVATED NOW ************
%  LMS0310 LMS VERSION ´03.4B00´ STARTED -------------------------------------  (7)
```

```
//OPEN UEB.BIB,U
//SHOW-E-ATTR
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
TYP NAME    VER (VAR#) DATE
(C) LMSCALL 007 (0001) 1995-08-12
     1 (C)-ELEMENT(S) IN THIS TABLE OF CONTENTS
//EXTRACT-ELEM *LIB(,LMSCALL,S),LMSCALL1
INPUT  LIBRARY= :N:$USER.UEB.BIB,DUF2EV=DISK
OUTPUT FILE
         SEL (S)LMSCALL/(0001)/1995-08-12 AS LMSCALL1
//END ------------------------------------------------------------------------  (8)
%  LMS0311 LMS VERSION ´03.4B00´  TERMINATED NORMALLY
***** LMS TERMINATED - RETURNING TO PROGRAM *****
```

(1) The program library UEB.BIB is designated as the input and output library.

(2) The source program LMSCALL is listed.

(3) The return address is loaded in register 14.

(4) Register 1 is set to 0.

(5) The address of the save area is loaded in register 13.

(6) The entry address LMSUPSDF is loaded in register 15.

(7) LMS is called from within the user program.

(8) Following termination of the LMS run, control is returned to the user program.

10.4 Migrating old library formats

LMSSDF only processes the library format. The old library formats shown below can be converted into the PLAM library format as follows. If these conversions fail, this means that the library format is unknown.

1. Converting from LMR, MLU and COBLUR libraries

```
/EXEC $LMSCONV     or /EXEC $LMS
$LIB libold
$LIB libnew,NEW
$DUP* *
$END
```

   libold is the library in LMR, MLU or COBLUR format which is to be converted.
   libnew is a newly created library in PLAM format.

2. Converting from FMS libraries

```
/EXEC $LMS
$LIB libnew,NEW
$ADDS FMS=libold(*)
$END
```

   libold is the library in FMS format which is to be converted.
   libnew is a newly created library in PLAM format.
   `ADDx FMS=libold(...)` uses the original date for the target member .
   `ADDx FMS=libold(...)>...` assigns today's date for the target member.

10.5 Product components

Logical ID	File name	Function
SYSPRG.ISP	LMS	Starter phase LMS (ISP)
SYSPRG.SDF	LMSSDF	Starter phase LMSSDF
SYSLNK	SYSLNK.LMS.034	LMS module library
SYSLIB	SYSLIB.LMS.034	User interface for LMS-UP
SYSMES	SYSMES.LMS.034	Message file
SYSSSC	SYSSSC.LMS.034	Subsystem declaration
SYSSSC	SYSSSC.LMS.034.PRELOAD	Subsystem declarations to preload the private part of LMS (see Release Notice)
SYSSII	SYSSII.LMS.034	Structure information ffr IMON
SYSSDF	SYSSDF.LMS.034	SDF syntax file
SYSFGM.D	SYSFGM.LMS.034.D	Release Notice, German
SYSFGM.E	SYSFGM.LMS.034.E	Release Notice, English
SYSREP	SYSREP.LMS.034	Correction file
SYSRMS	SYSRMS.LMS.034	Correction depot for RMS
SYSNRF	SYSNRF.LMS.034	Noref file
SYSACF	SYSACF.LMS.034	Alias catalog for LMSLIB

Table 6: Product components

Related publications

You will find the manuals on the internet at *http://manuals.ts.fujitsu.com*. You can order printed copies of those manuals which are displayed with an order number.

[1] **LMS**
(BS2000/OSD)
Subroutine Interface
User Guide

[2] **ARCHIVE** (BS2000/OSD)
User Guide

[3] **SDF** (BS2000/OSD)
SDF Dialog Interface
User Guide

[4] **BS2000/OSD-BC**
Commands
User Guide

[5] **BINDER** (BS2000/OSD)
User Guide

[6] **SECOS** (BS2000/OSD)
Security Control System
User Guide

[7] **SECOS** (BS2000/OSD)
Security Control System - Audit
User Guide

[8] **JV** (BS2000/OSD)
Job Variables
User Guide

[9] **BS2000/OSD-BC**
Introductory Guide to DMS
User Guide

[10] **EDT** (BS2000/OSD)
Statements
User Guide

[11] **EDT** (BS2000/OSD)
Unicode Mode Statements
User Guide

[12] **SDF-P** (BS2000/OSD)
Programming in the Command Language
User Guide

[13] **AID** (BS2000)
Advanced Interactive Debugger
Core Manual
User Guide

[14] **openFT (BS2000)**
Enterprise File Transfer in the Open World
User Guide

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

[16] **C Library Functions** (BS2000/OSD)
for POSIX Applications
Reference Manual

Index

T