

LMS (BS2000/OSD) Version 3.3

Issue February 2008

Pages 3

Library Maintenance System

LMS is the central library management system in BS2000.

LMS is used to create and manage program libraries, as well as to manage and edit the elements contained in those libraries.

Program libraries are PAM files which are processed using the PLAM library access method (PLAM: Program Library Access Method). For that reason they are also referred to as PLAM libraries. The PLAM access method is included in the BS2000 basic configuration.

LMS makes an excellent software development support tool, providing the following standout functions: standardized program library system, element version management, the SDF statement language, protection mechanisms which can also be used for elements, and options for automating program development and library management.

LMS is a sophisticated and mature BS2000 standard product which supports program development, maintenance and documentation.

Program libraries

A program library is a file with a substructure. It contains elements ("members") and a table of contents listing the stored elements.

Each library has an entry in the system catalog. The user can specify the name and other file attributes, such as retention period or sharability.

Storing multiple elements in a library relieves the load on the system catalog, since only the library is entered there, and not each element. What's more, it also saves storage space, since the elements are stored in the library in compressed form.

Elements

An element is a logically related set of data, e.g. input/output data, a procedure, a link module or a source program. Each element in the library can be addressed individually.

An element in a program library is uniquely identified by its type, its name and a version identifier. Multiple versions relating to one element type and name can also be stored.

In addition to standard types for typical contents, users can define their own types using names up to 8 characters long (user types) in order to structure their libraries more efficiently and automate processing of library elements with the aid of procedures in a much more finely grained manner.

Version management

Experience teaches us that software elements need to be changed frequently. For this reason they must be easy to change and the different versions must be uniquely identified.

Symbolic version identifiers support library management automation. Examples available include: *HIGHEST, for the highest version identifier, and *INCREMENT, to increment an existing version identifier in accordance with a selectable rule.

LMS provides compact storage of differential sets (deltas): elements which have been created from a prior element due to changes generally differ only slightly from their predecessor. With LMS, only the differences compared with the predecessor then need to be physically stored. When such element versions are read, these deltas are merged in again at the relevant points, thus making the complete element available to the user once more.

Integration into the programming environment

The utility routines of the programming environment, such as EDT, compilers etc., can directly access program libraries.

Link-and-load modules can be stored directly in program libraries by all the compilers and by the BINDER link editor. However, LMS can also copy link-and-load modules that originate from the old linker-loader system with the TSOSLNK linkage editor into the program library.

The LMS subroutine interface provides users with convenient options for processing LMS libraries and their contents, from directly within a user program (Assembler, C, COBOL). In this case LMS is loaded dynamically.

Multiple access to program libraries

A library can be opened by one or more users for both read-only and write access.

An element can be read concurrently by a number of users, but can only be accessed for writing by one user at a time. While an element is open for writing, no other access –

including read-only – is possible to this element. Other elements of the library can be accessed.

Statement format

LMS supports the SDF statement format familiar from the BS2000 command language. Statements in ISP format continue to be supported for compatibility reasons. New functions are only provided in SDF format, however.

Basic functions

LMS provides the following basic functions:

- Creating and managing libraries
- Adding elements to a library, copying elements into another library, and outputting elements from a library into a file
- Managing, listing and editing elements
- Comparing elements
- Making LMS functions available from within EDT
- Maintaining a Last Access Date for elements
- Defining default settings in a start file.

Protection functions

LMS supports the use of passwords and basic access control lists (BACLs) to provide a comprehensive, multi-tiered protection concept for libraries and elements.

When the SECOS product is deployed, the GUARDS subsystem can be used to set up an extended, user-definable access control mechanism for libraries and elements.

Protection rights are transferred to the elements on request when files are added, and vice versa when elements are output into files.

All security-related actions relating to libraries and elements can be logged by linking PLAM to the SAT subsystem (part of the SECOS product).

When elements are deleted, data that is no longer required can be selectively overwritten with the value X'00' (security erase).

Borrowing procedure

The borrowing mechanism provides a means of controlling access to elements that are being modified e.g. by several developers working on a project.

An element can only be written if the writer has previously been entered as the current holder for the source version.

During writing, a history comprising the return timestamp, the holder's user ID and optionally a user-specific comment is added to the new element status.

The processing status of an element is output as an attribute in the table of contents and can also be used as a selection criterion.

make functionality

The make functionality familiar from and successfully employed in other operating systems is also available under LMS. Using make enables projects to be implemented efficiently, since only absolutely essential steps are performed. The interface is tailored to match the BS2000 system.

For a given target object, the make function describes the source objects on which this target depends, and the actions leading to the creation of the target.

Starting from the selected target object, all targets are created from new if the source objects of a target have changed since the last time the target was generated.

As well as the selected target object, all associated source objects represent further sub-targets which are handled similarly.

A BS2000 procedure is created and can be launched synchronously or asynchronously.

To allow the description to be reused, the sequence of LMS-specific make statements should be stored in a separate element, called a make file.

Output of library information to S variables

LMS supports the output of selected data to S variables. This enables easy and efficient creation of command procedures (S procedures), e.g. for automated library administration.

If structured variables and lists are to be used here, then the chargeable product SDF-P is a software prerequisite.

Reorganizing libraries

LMS reorganizes a library so that the maximum amount of disk space not currently used by elements is released.

This enables the amount of disk space required for a library to be reduced. In contrast to copying and subsequently deleting the old library, no additional disk space (to the full size of the library!) is required for reorganizing a library.

Unlike with the COPY method, however, after REORG-LIBRARY there can still be an amount of free space which, at a maximum, may be as large as the largest element in the library.

Technical requirement

Hardware

BS2000/OSD business server

Software

BS2000/OSD-BC V3.0 or higher

When using files >32 GB:

BS2000/OSD-BC V5.0 or higher

Operating mode

Dialog and batch mode

Implementation language

Assembler, SPL

User interface

Commands (English)

Message texts (English or German)

Installation

In accordance with installation notes in the readme file

Documentation

LMS User Guide

LMS Subroutine Interface

Training

See courses offering at:

<http://www.fujitsu-siemens.com/training>

Demands on the user

BS2000/OSD application knowledge

Conditions

This software product is supplied to the customer under our conditions against a single payment or installments.

Ordering and delivery

This software product may be obtained from your Fujitsu Siemens Computers regional office.