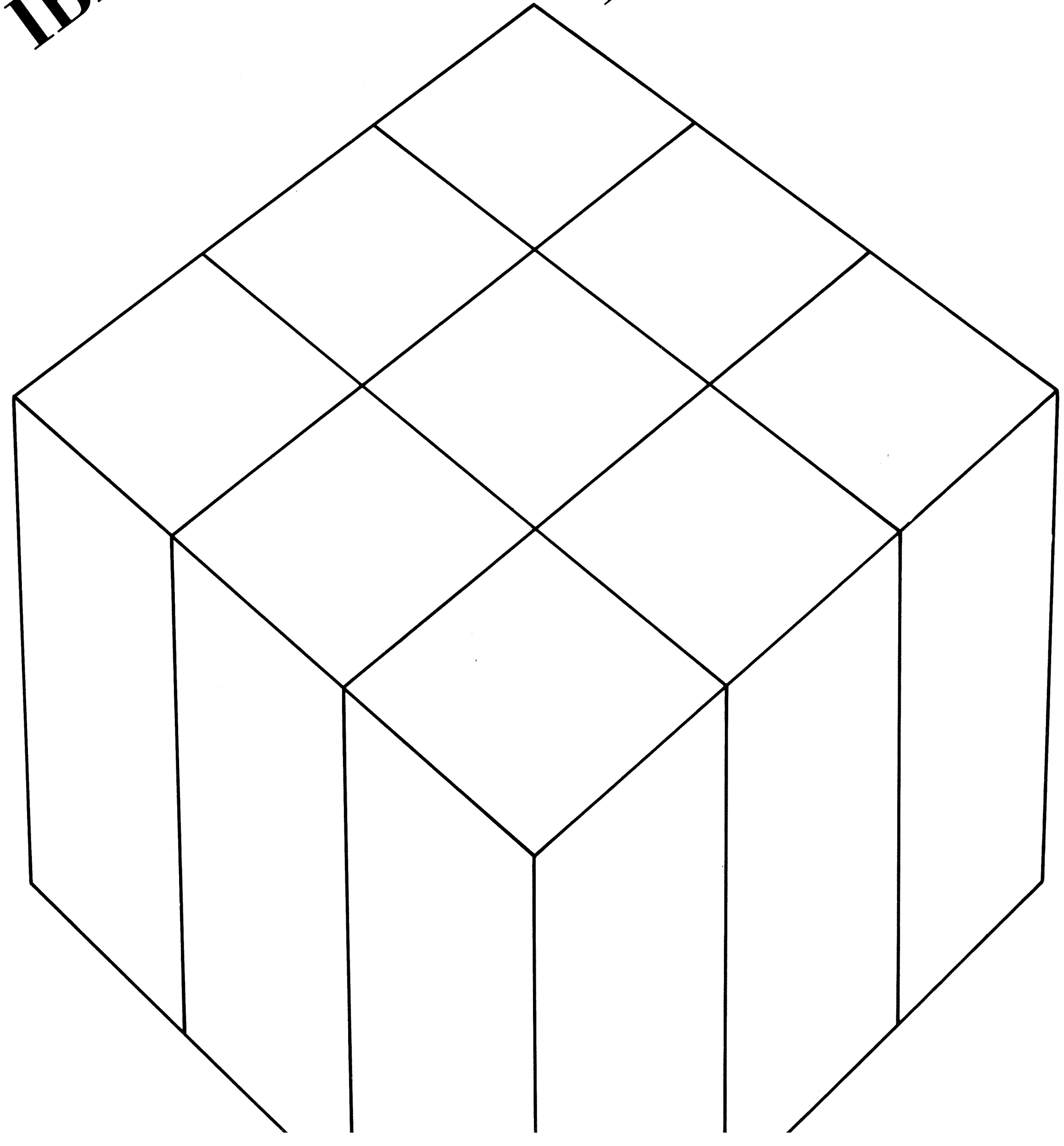
IBM Virtual Storage Extended Advanced Functions

Planning and Installation



IBM Virtual Storage Extended/ Advanced Functions

Planning and Installation

Version 2 Release 1

Program Number 5666-301

Order Number SC33-6193-02
File No. S370/4300-34

Third Edition (June 1987)

This edition is a major revision of SC33-6193-1. It edition applies to Version 2, Release 1 of IBM Virtual Storage Extended/Advanced Functions, Program Number 5666-301, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30XX and 4300 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

Preface

If you plan to install your VSE system by ordering the Virtual Storage Extended/System Package (referred to in this manual as "VSE/SP"), you may need this publication only for planning private libraries and for information about the options you can select for generating a location-tailored supervisor. All other planning and installation information you need is contained in the IBM publications:

VSE/System Package:

> *Planning*, SC33-6177
> *Installation and Use*, SC33-6178

You need *this publication* to plan, install, and service your VSE system if you order Virtual Storage Extended/Advanced Functions (referred to in this manual as "VSE/Advanced Functions" or "VSE/AF") plus a set of individual licensed programs. The publication describes how to make VSE/Advanced Functions operational after it has been restored from tape to disk. IBM recommends you to read through each chapter that applies to your task.

The publication discusses the Maintain System History program (MSHP), the IBM tool for installing VSE systems. It describes how to use MSHP to install VSE/Advanced Functions or a licensed program or feature on top of VSE/Advanced Functions. It describes how to install IBM supplied service changes such as PTFs or APARs.

To use this publication, you should be familiar with the operational concepts of VSE/Advanced Functions as described in the IBM publication *VSE/Advanced Functions System Management Guide*, SC33-6191.

You may have to consult the publications listed below when you prepare jobs for the installation of VSE/Advanced Functions, of a licensed program, or of a service change:

> *IBM VSE/Advanced Functions*:
>
> > *System Control Statements*, SC33-6198
> > *Maintain System History Program, Reference*, SC33-6199
>
> *Guide to the DOS/VSE Assembler*, GC33-4010

Given below is a list of non-VSE/Advanced Functions publications that are referred to in this publication or that you may have to consult:

IBM VSE/Access Control–Logging and Reporting, General Information, SH12-5130

IBM VSE/Fast Copy Data Set, Installation Reference, SC33-6082

IBM VSE/POWER, Installation and Operations Guide, SH12-5329

IBM Device Support Facilities, GC35-0033

Program Directory (as shipped by IBM with the distribution tape)

For other related publications, refer to Appendix D or to *IBM System/370, 30XX and 4300 Processors Bibliography*, GC20-0001.

The publication is organized in chapters as follows:

Chapter 1 – Introduces you to the tasks of planning a system and of installing licensed programs and service changes. The chapter gives you an overview of the activities that are involved; it summarizes the activities that can be done before system installation.

Chapter 2 – Provides information you need to plan for the private libraries you need for your own programs, and to generate a supervisor tailored to your requirements.

Chapter 3 – Tells you how to migrate the private libraries of a Version-1 system to Version-2 format sublibraries. Gives instructions for the conversion of affected job streams.

Chapter 4 – Gives a general description of the Maintain System History Program (MSHP); tells, in general terms, how MSHP works.

Chapter 5 – Expands on the pre-installation considerations given in Chapter 1. Gives a step-by-step procedure for installing VSE/Advanced Functions either stand-alone or on-line.

Chapter 6 – Describes how to install licensed programs and features under control of MSHP.

Chapter 7 – Describes the procedures for installing PTFs and APAR or local fixes.

The publication includes appendixes as listed below.

Appendix A – Lists the generation options used to generate the supervisors shipped by IBM on the distribution tape.

Appendix B – Describes the supervisor generation macros and their operands.

Appendix C – Lists the minimum machine requirements for installing and using VSE/Advanced Functions. Includes a list of supported processors and I/O devices.

Appendix D – Lists the relevant VSE/Advanced Functions publications for each user task. Provides a mapping chart showing which Version 2 publications correspond to which Version 1 publications.

Appendix E – Describes the job control scanner, a programmed migration tool.

The publication includes a list of abbreviations.

Contents

Figures

Summary of Changes

Amendments for VSE/Advanced Functions Version 2 Release 1

This publication has the same overall organization as the IBM publication *VSE/Advanced Functions Planning and Installation*, SC33-6096-1. However, the publication had to be revised almost completely to cover planning for and installation of VSE/Advanced Functions 2.1. The last edition of the manual, with the form number SC33-6193-1 contained these major changes, and the present edition contains further enhancements.

Major changes in the publications are listed below:

- A new chapter has been added dealing with migration.
- Link and delete procedures are no longer available; the appendix that listed these procedures has been dropped.
- The manual includes a list of supported processors and I/O devices. Such a list used to be available as an appendix of a *General Information* manual for the product. The manual has been dropped.

Following is a list of functions and support newly implemented for VSE/Advanced Functions 2.1:

- Hardware support

  The list below includes IBM processors and I/O devices that have been announced as supported since the availability of VSE/Advanced Functions 2.1.0.

– IBM 4361	A processor with ECPS:VSE.
– IBM 4381	A processor of the System/370 architecture.
– IBM 3290	A terminal device supported like an IBM 3270.
– IBM 3370 Model 2	An FBA (fixed block architecture) type disk storage device.
– IBM 3380	A CKD (count-key-data) type disk storage device with two models of similar characteristics but different capacities.
– IBM 3480	A two-drive, buffered tape device.
– IBM 3820	A page printer for use under control of ACF/VTAM.
– IBM 4248	A line printer supported as a printer of the PRT1 class and in native mode.

- Support of the IBM 9370 processors and their related I/O devices:
  - IBM 9332 – An FBA disk device;
  - IBM 9335 – An FBA disk device;
  - IBM 9347 – A streaming-mode tape device;
  - IBM 4224 – A 3268-compatible terminal printer;
  - IBM 4234 – A 3262-3/13-compatible terminal printer;
  - IBM 4245-12D/20D
- Support of the IBM 3720 Communications Adapter.
- Up to 16 megabytes of processor storage also when operating in System/370 mode.
- Performance improvements

  Virtual addressability extension – When operating in 370 mode, the address range available for the execution of system and application programs can have a size of up to 40 megabytes. For operation in ECPS:VSE mode, this maximum size is 16 megabytes, the same as in the past.

  Parallel page I/O – If the page data set is distributed over two or more disk volumes on separate drives, the system reads from and writes to these extents in parallel. This function is transparent to user-written programs.

  Virtual I/O support – A compile-and-go or assemble-and-go operation makes use of a virtual I/O area. This reduces I/O activity if sufficient processor storage is available. Data transfers from and to this area are accomplished at near page-I/O speed.

  Faster output on PRT1 and IBM 3800/3200 printers – If this output is controlled by logical IOCS, there is only one device request per print line instead of up to four as in the past.

  Larger block size for input via SYSIPT – The system accepts a block size specification large enough to support input spooling by VSE/POWER of 128-byte records from a diskette I/O unit.

  Device-independent full-track support – For a program making use of this support, the system reads or writes a full track at a time if a CKD-type disk device is used. The system reads or writes a block of up to 32K (minus 7) bytes if an FBA-type disk device is used.

  GETVIS area subpooling – This support, available on a macro level, helps reduce fragmentation (and thus wasting) of the available GETVIS space.

Channel-to-channel attachment (CTCA) support – The IPL command ADD now allows you to define the CTCA to the VSE system.

- Usability improvements

  New librarian – There is only one library service program and, therefore, always the same program call for any library service request. There is only one library organization for any types of library members (such as phases, modules, or procedures).

  Following are some of the new librarian's advantages over the library service programs available in the past:

  - Space freed by deletion of a library member (a phase or a relocatable module, for example) is reused. Condense runs are no longer required.

  - Private libraries may be distributed over two or more extents on two or more disk volumes of the same type. Private libraries may reside in VSAM managed space and thus be extended dynamically.

  - A larger block size for writing data into a library improves overall disk space utilization for libraries on CKD disk devices. This can be significant if the libraries reside on devices with a large track capacity.

  - An easy-to-use command language is available. Highlights of this new language are:

    - Free format within the character positions 1 and 72.
    - Meaningful command verbs.
    - Short forms for command verbs and keywords.
    - The use of generic names for member-name specifications.
    - Command continuation on one or more subsequent lines.

  - Protection of library contents down to an individual member level is available through the system's access control function.

  - Requests for library services may be submitted at the system console.

  Conditional job control language – It provides for conditional execution of individual job steps of a job. Conditions for execution can be set by user-written programs. The job control program evaluates these conditions on the basis of user specifications.

  Parameters in procedures – Procedures may include parameters. Actual values to be used instead of these parameters can be specified either in the procedure call or in certain job control statements.

  Nested procedures – Procedures may be nested in up to 15 levels.

  VM linkage enhancements in a 370-mode supervisor – Some of the enhancements that were available in supervisors generated with

VM=YES are now also available in a supervisor generated for operation in 370 mode; they are activated automatically whenever the supervisor is used for operation under VM/System Product.

Implicit OPEN of the hard-copy and recorder files – Failure to open these files explicitly during system start-up no longer results in a system wait.

Job accounting – The system now records the duration of a job step at the end of the step. This recording is in units of 1/300 of a second.

Automatic ADD for the SYSRES and SYSLOG devices – Start-Up procedures need no longer include ADD statements for the SYSRES and SYSLOG devices.

Storage dump evaluation – The *Information/Analysis* program provides support for dump management and for printing storage dumps generated by the system. If the *Interactive System Productivity Facility* (ISPF) is installed on top of VSE/Advanced Functions, then the program provides, in addition, support for displaying dump data on a display terminal.

End-of-extent exit – A user-exit routine, if defined, receives control whenever an end-of-last-extent condition occurs while a sequential output file on disk is being processed. This applies also to sequential files accessed by programs using physical IOCS. To use the support, affected programs must be reassembled.

Tape-file extension support – This support can be used to extend a tape-output file defined in an assembler-language program by DTFMT. To use the support, the program must be reassembled.

ANSI-tape support – In accordance with the ANSI standard X3.27-1977, Level 3, VSE/Advanced Functions:

- Denies access to files on these tapes if they are access protected. If this is not desirable, a user-written routine may be cataloged to cause a different action.
- Checks HDR2, EOF2, and EOV2 labels in addition to HDR1, EOF1, and EOV1 labels.

Tape I/O module in the shared virtual area – This eliminates the need for a tape I/O module to be linked with the application program.

Standard-labeled work files – Label processing for work files on tapes is the same as label processing for data files on tape.

Bypass automatic loading of print buffers – To make use of the auto-start feature of an IBM 4361, it may be desirable to have the system bypass automatic loading of print buffers during IPL. An IPL option is available to request this bypass.

Documentation – The number of orderable publications has been reduced. Part of the product's library has been rearranged so that closely related information can be found in one book.

The *Messages and Codes* publication is shared with the library for VSE/System Package, a pre-configured VSE system. The shared publication documents all the messages that a user of VSE/Advanced Functions may encounter in a typical VSE environment. The same approach has been taken for information on diagnosing problems; the publication *Guide for Solving Problems*, which is shared with the library for VSE/System Package, enables a user of VSE/Advanced Functions to start problem isolation under total system aspects.

- Availability improvements

  Standardization of the system – The number of supervisor generation options has been reduced to twelve. Operational flexibility is maintained by allowing users to activate or deactivate specific supervisor functions during system start-up.

  DASD sharing by up to 31 processors – DASD sharing was limited to four processors in the past.

  Delayed cancel support – A request to cancel certain system programs may be delayed to ensure an orderly program-end condition. The programs are:

  Job control
  Librarian
  Linkage editor

  Removal of hard-wait conditions – The following conditions will no longer result in a hard wait:

  - A full channel-check error queue.
  - Failure of two channels at the same time.
  - Channel-check during the recording of a channel check.

  ABEND disposition in DLBL statements for VSAM files – In addition to the close disposition, a disposition can be specified to take effect if a program terminates abnormally while a VSAM file is still open.

- Data protection improvements

  Extended access-control – Access control can be defined separately for any of the following resources:

  - A complete library
  - A complete sublibrary
  - A member stored in a sublibrary
  - A file containing data

  For a protected resource, various access types may be defined by user classes. These access types are:

- *Connect* (to a library or sublibrary) – Additional access rights must be defined for a sublibrary in case of a connect to a library or for members in case of a connect to a sublibrary.
- *Read* (which includes connect).
- *Update* (which includes read).
- *Alter* (which includes update) – allows a user to create, to rename, and to delete.

• Installability improvements

  The *Maintain System History Program*, the system's installation tool, provides support for:

  - Multiple release installation – Two or more releases of the same version of a licensed program (not VSE/Advanced Functions) can be installed and serviced under control of the program.
  - Multiple-target library service – Service changes can be installed on two or more libraries (or sublibraries) in one service run.
  - Product backup – If a product resides in one or more separate sublibraries, the program can create a backup copy of the product together with the related history file entries.

  In addition, the following improvements have been made to MSHP:

  - A PATCH function has been added;

  - The maximum size of the history file has been increased;

  - Multiple commands can be passed when MSHP is called.

• Serviceability improvement

  Extended recording of I/O errors – This recording (in the system's recorder file) now includes all I/O errors that occur during an I/O operation started by a user program. The recording also includes I/O errors that occur during disk I/O started by a system task. This recording takes place if neither:

  - A user error routine exists, nor
  - The return of sense information is requested.

• Removed support

  - For System/370 processing units Models 115 and 125.
  - Paper-tape I/O.
  - The COPYSERV program – The new librarian includes a similar service.
  - The programs BACKUP and RESTORE – The new librarian includes the required backup and restore support.

- The program PDZAP – For programs under control of MSHP, MSHP's alter function (a subfunction of CORRECT) may be used instead.
- The recording of IPL reason codes.
- The processing of DOS-format label information statements.

Chapter 1. Introduction

If you plan to install VSE/System Package, a pre-generated and pre-configured VSE system, you may need this publication only when planning private libraries and generating the supervisor to meet your own requirements. For users of VSE/System Package, the primary IBM publications for planning and installation are:

VSE/System Package Planning;

VSE/System Package Installation and Use.

VSE/Advanced Functions Planning and Installation tells you what you need to know to:

- Plan, install and service your VSE system if you do not order VSE/System Package;
- Make VSE/Advanced Functions operational after restoring it from tape to disk;
- Migrate to Version 2 of VSE/Advanced Functions from a Version 1 release or from a release of DOS/VS (or even DOS).

The publication gives a short description of the Maintain System History program (MSHP), the IBM tool for installing VSE systems. It describes how to use MSHP to install VSE/Advanced Functions, or other licensed programs or features on top of VSE/Advanced Functions, or IBM supplied service changes.

Planning the System

Careful planning can prevent your having to change the system later on. Areas that require your attention in planning the system are:

- The libraries: their sizes and their distribution over the available disk space;
- The system files and system work-files: their sizes and their placement on the available disks;
- The supervisor: its size and the required options.

Planning the system is discussed in Chapter 2.

Migrating to Version 2 of VSE/Advanced Functions

Version 2 of VSE/Advanced Functions includes a new librarian supporting a new library concept. Therefore, when you migrate to Version 2, you must convert your private libraries to the new library format. You must also check your cataloged job streams for possible incompatibilities.

Reading Chapter 3 helps you plan these migration activities.

The Maintain System History Program

This program (MSHP) controls the installation of VSE/Advanced Functions to a great extent. It also controls the installation of licensed programs or features and of service changes. MSHP maintains an on-line system history file for this purpose.

You should familiarize yourself with the functions of MSHP and its use as an installation tool. The program is described in Chapter 4. How to use the program for system installation is shown in Chapter 5; for licensed program or feature installation, in Chapter 6; and for the installation of service changes, in Chapter 7.

Installing VSE/Advanced Functions

Planning the installation process helps avoid the need for corrections during and after installation. You may contact your IBM representative to set up a planning meeting for system installation, if you wish.

At the end of the installation process as described in Chapter 5 you will have installed VSE/Advanced Functions. In addition, you will have installed the following system control programs (SCPs), which are co-requisites for VSE/Advanced Functions:

Device Support Facilities
Environment Recording Editing and Printing (EREP)
VSE/Online Test Executive Program (VSE/OLTEP)

You will probably have to install additional licensed programs to adapt your computer system to your data processing requirements. This installation is discussed in Chapter 6.

Given below are a summary of the contents of the distribution tape and a discussion of installation-related activities you can do before actual installation. An overview of the installation process itself follows.

The VSE/Advanced Functions Distribution Tape: The code as shipped includes all the programs and routines you need to install VSE/Advanced Functions (or other licensed programs later on). It includes a number of loadable supervisors from which you must select one. For details about these supervisors, refer to Appendix A.

Besides these supervisors and the code for VSE/Advanced Functions, the tape contains the following stand-alone programs:

Device Support Facilities (for a description, see the IBM publication *Device Support Facilities*).

The RESTORE support, which you need only for restoring VSE/Advanced Functions from the shipment tape to disk (how to do this is explained later in this publication under "The Stand-Alone Installation Procedure").

The FASTCOPY utility (for a description, see the IBM publication *VSE/Advanced Functions System Management Guide*).

Format Emulated Extent (for a description, see the IBM publication *VSE/Advanced Functions System Management Guide.*

VSE/Advanced Functions is shipped in two libraries as follows:

- PRODUCTION library

  This library contains the programs and data which you must have on-line at all times in order to use VSE for productive work.

  The library consists of a number of sublibraries that contain the product's phases, object modules, macro definitions and procedures. These sublibraries include the SCP support that is necessary to run VSE/Advanced Functions. The sublibraries do not include object modules or source members of code which is available as loadable phases.

- GENERATION library

  This library contains the programs which you need for program modification, such as the assembly of your own supervisor. The generation library also consists of a number of sublibraries. These sublibraries contain the object modules and source-code members (if applicable) of the phases and modules in the production library.

For a complete listing of the library contents, refer to the *Program Directory* for VSE/Advanced Functions.

Preparing the Installation: Careful preparation of the installation can speed up the actual installation process. Here is a summary of the things that you can do in advance:

- Ensure that the hardware requirements are met.

  These requirements are summarized in Appendix C.

  Note: Some system programs (SDAID, for example) internally use the hardware address 000. Therefore, to avoid device assignment errors, do not configure your system with an I/O device at that address.

- Initialize required disk volumes.

  You can do this in parallel with other data processing if:

  - You are migrating to this release from a previous release of VSE/Advanced Functions, and
  - The disk devices involved in the process of system installation are supported by both this release and the release from which you are migrating.

- Back up your current libraries and the system history file to provide an input tape for library conversion.

  If you are migrating from DOS/VS Release 34 or from a Version 1 release of VSE/Advanced Functions, the *System Utilities* manual for your current release tells you how to use the BACKUP program for this purpose. For more information about library conversion, see Chapter 3.

The Installation Process: MSHP is involved in this process. This ensures that your system's history file reflects the change level of VSE/Advanced Functions when the installation is complete. The procedures given in Chapter 5 show when and how to use MSHP during system installation. For details about MSHP control statements, refer to the IBM publication *VSE/Advanced Functions Maintain System History Program Reference.*

Following is a summary of the major activities for installing VSE/Advanced Functions (using the INSTALL SYSRES function of MSHP):

- Restore the system from tape to disk.

  This requires one or more initialized disk volumes to be on-line.

  The distribution tape includes stand-alone support for restoring the PRODUCTION library of VSE/Advanced Functions from tape to disk.

- Start up the restored system.

  After this step, you are working with your new system.

- Store permanent label information.

  You may want to use the applicable IBM supplied procedure for storing label information. The label information defined by the DLBL and EXTENT statements in the supplied procedures must be adjusted to reflect the volume serial numbers used at your location. The procedures may also have to be adjusted to reflect different file sizes. The contents of the IBM-supplied procedures are listed in the *Program Directory*.

- Restore and personalize the history file.

  A system history file reflecting the installed IBM components and their current change levels is a requirement for program service. Personalizing the file means supplying MSHP with information about your installation. This information is used later to identify list output of the contents of the history file.

- Assemble and catalog your own supervisor.

  You cannot assemble a supervisor if you decide to install only the PRODUCTION library of VSE/Advanced Functions. In that case, use one of the IBM supplied supervisors instead. For a listing of options used to generate the IBM supplied supervisors, refer to Appendix A.

  If you install the GENERATION library also, you can assemble and catalog a supervisor of your own. Appendix B gives a description of the available supervisor generation macros and the options that may be specified for supervisor assembly.

- Assemble and catalog I/O modules.

  You might want to skip this step altogether.

- Produce a backup copy of the newly installed system.

- Obtain hard-copy records of the newly installed system.

Installing a Licensed Program

An IBM licensed program is normally shipped as a DLIB (distribution library) tape. The tape contains, besides the code of the product, a history file that includes the installation and service data for the product. MSHP restores the product from the distribution tape into the library which you specify.

The product you want to install may be shipped with a GENERATION library in addition to a PRODUCTION library. You can restore each of these product libraries into different sublibraries or even different libraries in your system.

MSHP merges the history file restored from the distribution tape into your system history file. It records the residency of any restored library in the merged history file.

If, for any reason, you copy or move an installed product into a different library (or sublibrary), you have to indicate the product's new residence to MSHP.

How to install a licensed program is described in Chapter 6; a sample job for defining the residence of a system component is given in Chapter 7.

Installing Service Changes

Normally, IBM distributes a service change for a component of a VSE system in the form of a program temporary fix (PTF) or an authorized program analysis report (APAR) fix. A PTF contains one or more phases, modules, or macros replacing existing phases, modules, and macros, respectively. An APAR fix is an update to an individual phase, module, or macro.

You install service changes under control of MSHP. How to use the program for this purpose is described in Chapter 7.

Maps and Listings Produced During Installation

Make a point of saving all linkage editor output to SYSLST (including maps) obtained during system installation and during a future update. These maps provide information about the load address (relocation) of each component. Save also all assembly listings produced during installation, in particular supervisor assembly listings. You may need these maps and listings for problem determination if a program or system malfunction occurs.

Always make a printout of the system history file after the installation of a product or a service change. This listing is a complete record of the change levels of all the components in your system.

Reading Sample Jobs and Prompting Examples

In the sample jobs shown as part of the installation procedures:

- Character strings in uppercase are to be coded as shown.
- Character strings in lowercase represent variable data which you have to supply.

Explanations which you may need to help you understand the sample jobs are given as numbered notes either below the samples or as part of the associated text. References to these notes are given as numbers in parentheses to the left of the statements to which the notes apply.

In the examples showing allocation of disk extents, provide appropriate values for the following symbols:

number1 =	The number of the start track or block relative to zero (the beginning of a disk volume).
number2 =	The number of tracks or blocks to be reserved for the file that is to be defined (a library or sublibrary, or a history file, for example.

Chapter 2. Planning the System

This chapter discusses the following major considerations for system planning:

- Private libraries for your system – See “Private Libraries” on page 2-2 below.
- The system files and system-work files – See “The System and System-Work Files” on page 2-10.
- The size of the system's label-information area – See “The Label-Information Area” on page 2-18.
- The size of your virtual storage – See “The Size of Virtual Storage” on page 2-20.
- Processor storage for the partitions – See “Processor Storage for the Partitions” on page 2-26.
- The functions to be available with your system's supervisor(s) – if this consideration should be of significance to your data processing requirements – See “The Supervisor” on page 2-26.

The first four points in the above list apply to the planning of files on disk. Take into consideration that, under the VSE system, the volume table of contents (VTOC) on a CKD disks must be contained within one cylinder of the volume. Avoid having too many small files on one CKD volume.

In the formulas given as an aid for estimating space requirements, necessary rounding is indicated like this:

$\lceil \texttt{expression} \rceil$ = round to next higher integer.

$\lfloor \texttt{expression} \rfloor$ = round to next lower integer.

Private Libraries

If you install VSE/Advanced Functions as part of VSE/System Package, the libraries to contain the IBM-supplied program products have already been set up. The distribution of these libraries over the required disk volumes is already taken care of.

If you install VSE/Advanced Functions separately, planning for additional program products and programs of your own is an essential aspect of optimum performance. Such planning ensures that:

- The libraries are large enough to allow for future additions.
- The libraries are accessed by the system with maximum efficiency.
- No disk space is wasted by retaining on-line any code that is not required for daily operation. An example is a product's GENERATION library (if it is to be used at your location). Such a library may be taken off-line after the necessary generation activities are complete.

The major considerations involved in planning your system's libraries to your specific needs are:

- Which libraries are required to be on-line.
- How many disk drives are available and where on these devices should the libraries be placed.
- How large should each of the libraries be and what should they contain.

The answers to the above questions vary from location to location. Therefore, this section just gives a discussion of the advantages of private libraries and of general planning considerations; it provides formulas for estimating the required disk space.

System Library Versus Private Library

A VSE system cannot operate without a *system library*. This library, the system residence file, contains the IBM-supplied supervisors and the phases that keep your computer system operational. The library includes the relocatable modules, macros, and procedures needed for this purpose.

Although you may catalog additional members in the system library, IBM recommends that you catalog these members in private libraries. This eases migration to another release, and it can help speed up system service activities.

Using *private libraries* adds a great deal of flexibility; it may well improve overall system performance. This becomes apparent by a few examples of the use of private libraries:

- Having private libraries with sublibraries on separate disk drives requires less disk arm movement for locating a specific library member. This means faster access to libraries in general.

- Private libraries are useful in a testing environment where you want to keep working copies of your programs intact in one library while you test modifications to the same program from another.

- For any of your system's partitions, you can chain (concatenate) the sublibraries of the libraries that you maintain on-line. This establishes search orders for the system programs that retrieve library members (phases, modules, books, or procedures). By placing sublibraries containing frequently used members at the head of a search chain, you can speed up the retrieval of library members in general.

- A number of small sublibraries instead of a large one eases the task of program service for the libraries.

- Private libraries may reside in VSAM-managed space if VSE/VSAM and its SAM space-management feature are part of your VSE system. By defining these libraries as extendable files, you can minimize the risk of getting into a library-full situation.

  How to define a library in VSAM-managed space is described in the IBM publication *VSE/Advanced Functions System Management Guide.*

To estimate the total disk space needed for your libraries, you must, of course, consider the requirements of your own code. This section gives you general guidance for planning the libraries. It includes formulas that help you estimate the disk space needed for programs and data of your own. For the disk space requirements of IBM supplied code, refer to the *Program Directory* that you receive with the shipment package.

General Planning Considerations

To plan your private libraries, determine their contents for daily use and then establish the required disk space. Make some allowance for possible future expansions of the libraries as a result of newly developed or installed application programs. Detailed planning such as this should eliminate library overflow situations; it should eliminate the need for defining and creating new libraries.

Consider the following before you decide on the contents and size of your libraries:

- The number of library members (such as phases and procedures) you want to be on-line and how you plan to group them. You may, for example, group them by applications.

- The amount of available disk space and the number of available disk drives.

- The amount of IBM supplied code to be retained on-line.

The system residence file (which contains most of the PRODUCTION library) must always be on-line. An installed program product's PRODUCTION library must be on-line whenever this program product is to be used.

You may want to use the GENERATION library of a program product for program modification. You can take this library off-line after having completed the necessary generation activities. This approach frees disk space for use as work space or for data.

Taking a GENERATION library off-line requires a library-backup run if this library was modified during the generation activities; it means no more than retaining the distribution tape if the library remained unchanged. A product's GENERATION library may be restored to disk again when the library is required for the installation of service changes.

- The need for a dump library

  For efficient use of Information/Analysis, a dump evaluation aid of VSE/Advanced Functions, you should plan for and define a dump library. If this library exists, the system writes an automatic dump into the dump sublibrary defined for the dump-requesting partition.

  IBM recommends that you define your dump library with a separate sublibrary for each of your system's partitions, using naming conventions in the librarian DEFINE command as follows:

  ```
  DEFINE SUBLIB=SYSDUMP.BG SYSDUMP.F1 ... SYSDUMP.Fn
  ```

  Consider defining your dump library as secured (by way of the DSF operand in the DLBL statement); this avoids that stored dump information is overwritten inadvertently. Consider also having this library access protected (by using the system's access control function); this avoids that unauthorized persons can gain access to sensitive data which might be on-line as part of a storage dump.

- The need for cataloging IOCS modules

  This requirement varies from location to location; it depends on the kind of application programs that are developed in your area; it may arise if your migration to a new release involves also the installation of a new I/O device other than disk or tape.

  IBM supplies IOCS modules needed by the VSE supported compilers. Since these modules may be linked also to user-written programs, there may be no need for you to catalog IOCS modules of your own. For a list of IOCS modules shipped with VSE/Advanced Functions, see the *Program Directory* (Attachment D.0 under the identifier 30102 in the "Component Name" column).

  IOCS modules may be assembled and cataloged at any time; there is no need for you to plan this as part of the installation process. However, if you catalog IOCS modules of your own, you should do this under MSHP control. This ensures that the system lists any module that may be affected by a service change to the applicable module-build macro.

Estimating Disk Space Requirements

On-line disk space may not be a problem at your location. In this case, you can define a certain number of disk volumes to be used as private libraries and as the system's dump library. The librarian manages the available space. By periodically requesting a listing of the directory of each of your libraries (by way of a librarian LISTDIR request), you can check how much of the space is in use.

If on-line disk space may become a problem, it is advisable that you estimate the required disk space while planning your system.

Estimating Disk Space for Private Libraries

The disk space required for a library is the sum of control information and library-member space. Both control information and members are stored by the librarian in units (library blocks) of 1K bytes.

Space for Control Information: The control information for a library consists of:

1. A library header

   It contains, in one library block, the library descriptor plus one index entry per sublibrary for up to eight sublibraries. If a library has more than eight sublibraries, the librarian needs one extra block for the next ten sublibraries; and so on.

2. Free space maps

   A library includes one such map per extent. Every bit of this area represents one library block in the extent. The formulas below help you estimate the space requirement for these maps. In these formulas:

   ```
   s = The space requirement in number of library blocks.
   t = The number of 1024-byte blocks that fit into the asso-
       ciated library extent.
   ```

   The formulas are:

   - For the first extent of a library:

     If $t < 6777$, then: $s = 1$

     If $t > 6776$, then: $s = 1 + \lceil (t - 6776) / 7880 \rceil$

- For each additional extent of a library:

  $$\text{If } t < 7353\text{, then: } s = 1$$

  $$\text{If } t > 7352\text{, then: } s = 1 + \lceil (t - 7352) / 7880 \rceil$$

3. Library directory

   For each of its sublibraries, a library always has a member directory (the first-level index); it normally has higher level indexes. Figure 2-1 on page 2-7 shows how required indexes are set up by the librarian.

   The space you need for the indexes depends on the number and type of members that you want to have stored in a sublibrary. The formulas below may help you estimate this space; the space actually needed may vary, however, depending on the sequence in which members of the various types are stored. The formulas are:

   a. Requirement for the first-level (member) index:

      $$nl = \lceil mp / 5 + mo / 7 \rceil$$

      ```
      where mp = Members of type phase
            mo = Other-type members
            nl = Number of required library blocks
      ```

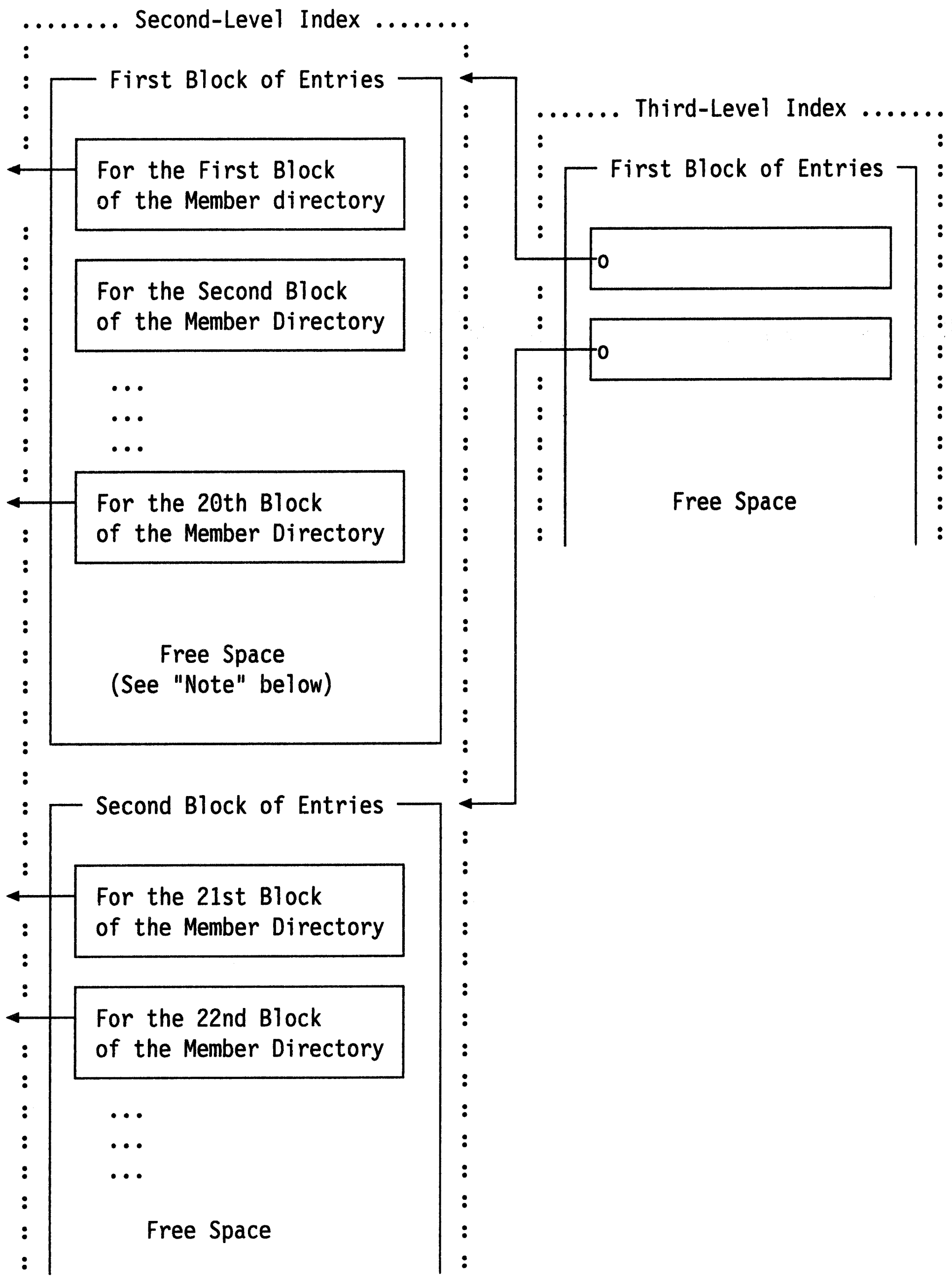
   b. Requirement for higher-level indexes:

      Library blocks (nh) needed to maintain higher level indexes:

      $$nh = \lceil nl / 29 + 1 \rceil$$

      ```
      where nh = The number of required library blocks
            nl = The number of library blocks needed for the
                 next lower index level
      ```

      As Figure 2-1 shows, the librarian sets up a next higher level index when the index of a certain level exceeds two blocks.


Second-Level Index
First Block of Entries
For the First Block of the Member directory
For the Second Block of the Member Directory
For the 20th Block of the Member Directory
Free Space (See "Note" below)
Second Block of Entries
For the 21st Block of the Member Directory
For the 22nd Block of the Member Directory
Free Space
Third-Level Index
First Block of Entries
Free Space

Note: When a directory block becomes full, the librarian splits the block by copying the upper half of the entry into a new block.

Figure 2-1. Library-Index Structure

Member Space: The member space you need depends primarily on the size of your members. It may depend on their number if you have many small ones because each member is stored in its sublibrary on library-block boundary. It follows then that space not used in the last block of a member remains unused.

To do a fairly accurate estimate of the required member space, proceed as indicated below. In the given formulas

np = Number of library blocks for members of type phase
no = Number of library blocks for members of type other than phase
s = Size (in number of bytes) of a member to be stored in the sublibrary

1. Find the requirement, in number of library blocks, for each member of type phase (np). Use the formula:

$$np = \lceil s / 988 \rceil$$

2. Find the requirement, in number of library blocks, for each member of type other than phase (no). Use the formula:

$$no = \lceil s / 960 \rceil$$

3. Add the results of your calculations for steps 1 and 2.

Consider possible library expansion as a result of cataloging additional members into existing or newly created libraries.

Estimating Disk Space for the Dump Library

One of the dump sublibraries, normally the one for partition BG, is used for accommodating a stand-alone dump. You may have to load a stand-alone dump of your largest address space into this sublibrary either for printing the dump data or for evaluating this data online. The remaining sublibraries need to hold only a dump of the partition plus the supervisor.

Your dump library should be large enough to hold (1) a stand-alone dump of your largest address space and (2) a dump of the largest partition in any of the other address spaces plus the supervisor. This can well meet your requirements because:

1. You do not define the size of a sublibrary – it is extended dynamically by the librarian if additional disk space is needed.
2. A dump can be deleted when it is no longer required.

For information about estimating the sizes of the partitions and the supervisor, refer to "The Size of the Partitions" on page 2-24 and to Figure 2-9 on page 2-21, respectively. Given below is a formula that helps you estimate the disk space needed to store dumps in the various dump sublibraries:

$$n = \lceil ((s + sva + ps) / 988) \rceil + pn + 3$$

where: n = Number of library blocks for the sublibrary
pn = Number of partitions
ps = Total size of partitions
s = Size of the supervisor
sva = Size of the SVA

To get the number of required tracks if your dump library resides on a CKD disk, divide n by the number of library blocks per track. A table under "Defining Library Space" on page 2-10 gives you the number of library blocks per track.

To get the number of required FBA blocks if your dump library resides on an FBA disk, multiply n by 2.

Following is an example for estimating the disk space needed to accommodate a stand-alone dump. The example assumes an environment as indicated below:

- Library on an IBM 3350.
- Largest address space (accommodating five partitions) as follows:

```
Partition F1           2,048K  *    * Defined as a shared
Partition F2           1,536K  *      partition
Partition F3             128K  *
Partition F4           3,072K
Partition F5           1,024K
                       ------
Total:                 7,808K =  7,995,392 for ps
The Supervisor           512K =    524,288 for s

The SVA                4,096K =  4,194,304 for sva
                                ----------
Total for use in the formula:   12,713,984
```

The number of library blocks (n) therefore is:

$$n = \lceil (12,713,984 / 988) \rceil + 5 + 3 = 12,877$$

The number of tracks (t) to be reserved then is:

$$t = \lceil 12,877 / 15 \rceil = 859$$

To determine the disk space needed to accommodate an automatic dump of a certain partition – F3 for the above environment, for example – use the same formula:

```
  Supervisor             512K =   524,288 for s
  Partition F3           128K =   131,072 for ps
  Size of the SVA      4,096K = 4,194,304 for sva
                                ---------
Total for use in the formula:   = 4,849,664
```

$$n = \lceil (4,849,664 / 988) \rceil + 1 + 3 = 4,913$$

The number of tracks (t) to be reserved then is:

$$t = \lceil 4,913 / 15 \rceil = 328$$

Defining Library Space

You do this by way of a set of DLBL and EXTENT statements for each library that you plan. To actually build the library (and its sublibraries), you use the librarian command DEFINE. For more information about building a library, refer to the IBM publication *VSE/Advanced Functions System Management Guide.*

Following is a table showing the number of library blocks per track on the various CKD disk devices. The table helps in working out the size definitions that have to be given (in the EXTENT statements) in number of tracks.

Device Type	Blocks per Track
IBM 2314/2319	6
IBM 3330/3333	11
IBM 3340/3344	7
IBM 3350	15
IBM 3375	25
IBM 3380	31

For a library on an FBA disk, including the IBM 9332 and IBM 9335, reserve two FBA blocks per required library block.

The System and System-Work Files

If you install VSE/Advanced Functions as part of VSE/SP, these files are set up, and corresponding label information is stored in the system library as shipped by IBM. In that case, you may safely skip this section.

The system residence (SYSRES) file is one of the system files; its residence and size has to be taken into consideration. The file's residence is location dependent; its size is given in the *Program Directory*, which you receive with the VSE/Advanced Functions tape. The residence of the other system and system-work files and also their sizes deserve some thought. The system files besides SYSRES are:

Recorder file
Hard-copy file
History file
Page data set
Lock communication file

The label-information procedures included in the SYSRES file of VSE/Advanced Functions specify near-minimum sizes for the system files listed in Figure 2-2 on page 2-11. For the remaining system files, size considerations are discussed separately under suitable headings.

Recorder File

The recorder file contains recovery-management support statistics. These statistics are used primarily by IBM service personnel to analyze the performance of your system, should there be a need. The information in the file is related, for example, to machine errors.

The system logical unit used for the file is SYSREC; the name of the file is IJSYSRC. You define the file (by DLBL and EXTENT statements) as one disk extent on a device of the type supported for system residence.

The system creates the file in either of the following ways: on first occurrence of a JOB statement after a SET RF=CREATE during system start-up; implicitly should this SET command be omitted.

For size considerations, refer to Figure 2-2.

Size values are given in number of tracks for CKD disks and in number of blocks for FBA disks.

IBM recommends that you plan the size of these files as large as or larger than recommended in the "Initial Planning" columns of this illustration.

	Recorder File		Hard-Copy File		History File	
Device Type	IBM Defined	Initial Planning	IBM Defined	Initial Planning	IBM Defined	Initial Planning
FBA (blocks)	200	1100	600	2000	400	2000
IBM 2314/2319	20	80	40	160	80	160
IBM 3330/3333	19	57	38	57	57	95
IBM 3340/3344	24	60	60	96	36	96
IBM 3350	30	30	30	60	90	90
IBM 3375	24	24	36	36	36	36
IBM 3380	75	75	75	75	15	30

Figure 2-2. Sizes of System Files

Hard-Copy File

A hard-copy file is required if your system operates with a display-type console.

The file contains all lines that are written to the system's display-type operator console. It includes the messages whose display was suppressed by VSE/Operator Communication Facility (VSE/OCCF) or which were routed by this program to another destination.

The file, one disk extent, must reside on the same disk volume as the recorder file. The system logical unit used for the file is SYSREC; the name of the file is IJSYSCN. You define the file by DLBL and EXTENT statements.

The system creates the file either on first occurrence of a JOB statement after a SET HC=CREATE during system start-up or implicitly should this SET command be omitted.

For size considerations, refer to Figure 2-2 on page 2-11.

History File

The Maintain System History program (MSHP) uses this file. MSHP records, in the file, the installation of additional program products and of any service changes for VSE/Advanced Functions and these products. VSE/Advanced Functions as shipped by IBM includes an up-to-date history file for the product. This file is a pre-requisite for program service by IBM.

The file is a disk extent with the file name IJSYSHF. There are no restrictions as to which disk device is to hold the file, but it is good practice to place the file on the same disk volume together with the recorder file and the hard-copy file.

For size considerations, refer to Figure 2-2 on page 2-11.

Page Data Set

The page data set, a file on disk, accommodates paged-out pages of programs that run in virtual mode. The size of this file depends on the size of your system's virtual storage; this size is discussed under "The Size of Virtual Storage" on page 2-20.

You define the *cpge data set* through the IPL command DPD during system start-up. In this command, you can specify:

- The address of the disk drive to be used or the applicable volume serial number.

- For a CKD disk

  - The file's start address as cylinder number relative to zero.
  - The number of cylinders to be used.

- For an FBA disk

  - The file's start address as a block number relative to zero.
  - The number of blocks to be used.

If your system runs as a virtual machine under VM/SP with a MODE=VM supervisor, you do not define a page data set.

You may want to operate under VM/SP with a MODE=370 supervisor to make use of the VAE support. In this case, you have to define a page data set because the page-control function of your VSE supervisor may not be suppressed.

The page data set can reside on any disk device that is supported as a system residence device. The data set may be spread over up to 15 extents, up to three per volume.

The table below shows how much storage can be mapped onto one cylinder of the possible CKD disk devices:

	Storage	
Device Type	370 Mode	E Mode
IBM 2314/2319	80K	120K
IBM 3330/3333	220K	190K
IBM 3340/3344	192K	144K
IBM 3350	480K	304K
IBM 3375	384K	336K
IBM 3340	600K	540K

If your page data set is to reside on one or more FBA disks, reserve four FBA blocks per 2K bytes of your virtual storage.

Residence of your page data set may affect your system's performance. Place the file on a volume (or volumes) not frequently accessed by your applications or by system programs. If you distribute the file over two or more volumes, the system performs page-in and page-out in parallel. This can improve system availability considerably.

Note: Avoid having the page data set share a disk volume with VSE/POWER queues.

An extension of the page data set accommodates a *virtual I/O work area* for the linkage editor. The linkage editor uses this area for improved assemble/compile-link-and-go processing. By default, the system reserves for virtual I/O use:

- An area of 64K bytes at the upper end of the system GETVIS area (as part of the SVA). This area is referred to as VPOOL area. A VPOOL area of the default size normally is large enough. However, if a number of link-and-go operations may take place in parallel, consider a larger VPOOL area by adding one or more increments of

  32K bytes for operation in E or in VM mode.
  64K bytes for operation in 370 mode.

- An area of the same size on the page-data-set volume immediately behind the page data set. The area is referred to as VIO area and must be as large as or larger than the VPOOL area. Define a VIO area large enough to hold your largest application program that you expect to be developed on your system. The area can have a size of up to 40M bytes.

You define the VPOOL and VIO areas during system start up either in your IPL ASI procedure or in response to message 0I03D during system start-up. This definition is in the form:

```
VPOOL={nK|nM} VIO={nK|nM}
```

If you intend to use IBM VSE/POWER Version 2.3, define additional VIO space. The IBM publication *VSE/POWER Installation and Operation Guide* provides a formula for calculating the appropriate value for your system.

Lock-Communication File

You need this file only if two or more computer systems at your location share data or libraries on disk. You create the file (and cause it to be opened) by means of the IPL command DLF during system start-up. In this command, you specify:

- The address of the disk drive to be used or the applicable volume serial number. For system start-up, this disk drive must be defined with the SHR (share) option.
- For a CKD disk
  - The file's start address as cylinder number relative to zero.
  - The number of cylinders to be used.
- For an FBA disk
  - The file's start address as a block number relative to zero.
  - The number of blocks to be used.

Space considerations:

	CKD Disk	FBA Disk
Default size	1 cylinder	80 blocks
Smallest size	1 cylinder	30 blocks

If the file resides on a CKD disk device, one cylinder should meet your location's requirements.

If the file resides on an FBA disk device, you may use the following values as a guidance for planning:

Number of Sharing Processors	No. of Blocks
Up to 4	80
Up to 10	270
Up to 20	790
Up to 31	1,540

The number of resources accommodated in a certain size disk area is a function of this size and the number of sharing processors as is explained in Figure 2-3.

For a lock-communication file on an FBA disk:

$$n = \lfloor 508 / (12 + m) \rfloor \times b$$

For a lock-communication file on a CKD disk:

$$n = \lfloor 508 / (12 + m) \rfloor \times c \times d$$

where: b = Number of FBA blocks

c = Number of cylinders

c = Number of cylinders

d = Device-dependent multiplier as shown:

Device	Multiplier
IBM 3330/3333	380
IBM 3340/3344	144
IBM 3350	810
IBM 3375	480
IBM 3380	690

m = Number of processors

n = Number of resources that can be locked

Figure 2-3. Size Estimation for the Lock-Communication File

System Work Files

System work files on disk are needed for compiling (or the assembly of) source statements and for preparing input to the linkage editor. These files use logical-unit and file names as follows:

Logical Unit Name	File Name
SYSLNK	IJSYSLN
SYS001	IJSYS01
SYS002	IJSYS02
SYS003	IJSYS03
SYS004	IJSYS04
SYS005	IJSYS05
SYS006	IJSYS06

The work files are defined by way of DLBL and EXTENT statements.

Figure 2-4 gives the logical unit requirements for the assembler and the compilers most frequently used on a VSE system.

	Assembler	DOS/VS COBOL	DOS/VS RPG II	DOS PL/I Optimizer
SYSLNK	L	L	L	L
SYS001	M	M	M	M
SYS002	M	M	M	M
SYS003	M	M		
SYS004		M		
SYS005		O		
SYS006		O		

where: M = Mandatory
O = Optional
L = Required when link-editing is necessary

Figure 2-4. System Work File Requirements of Language Translators

The size requirements of the files vary; how to estimate these requirements for the linkage editor and assembler work files is discussed in separate sections below. For more information about estimating these requirements for other language translators, refer to the installation or planning documentation available for these products.

Consider defining your system work files in VSAM-managed space. The automatic file-extension capability of VSE/VSAM enables you to use space more efficiently: you do not have to keep large amounts of disk space tied up; there is no need for defining these work files separately for each partition in which they are needed. For each work file you use, you request a certain amount of space that is valid for most of your jobs. You define this amount, an average size, as primary allocation. VSAM then performs secondary allocations whenever there is a need for more than the average size.

Following are the formulas you can use for estimating assembler and linkage editor work-file space not in VSAM-managed data space. You may use the formulas for estimating average sizes of these work files in VSAM-managed space.

Work Files for the Assembler

The formulas that you can use to estimate the required disk space are given in Figure 2-5 on page 2-17.

1. Work file on:

   a. SYS001 - whichever is the larger of:

      60 x (ts + ds)
      60 x (ds + ls)

   b. SYS002 - whichever is the larger of:

      60 x (ts + ds)
      40 x txt

   c. SYS003:

      If option NOXREF is in effect: 60 x ys
      If option XREF is in effect: 100 x ys

   d. SYSLNK:

      15 x txt

2. Convert the results to number of tracks (for a file on a CKD disk) or blocks (for a file on an FBA disk) by dividing them by a device-dependent divisor as shown:

Device Type	Divisor	Device Type	Divisor
IBM 2314/2319	6,000	IBM 3375	34,000
IBM 3330/3333	12,000	IBM 3380	47,000
IBM 3340/3344	6,000	IBM FBA disk	500
IBM 3350	16,000		

Legend:

ds	= Number of statements in source macro definitions.
ls	= Number of statements in library macro definitions called by the program that is being processed.
ts	= Total number of source statements.
txt	= ys minus ds minus number of comment statements.
ys	= Total number of statements on SYSLST.

Figure 2-5. Formulas for Estimating the Size of Assembler Work Files

Work Files for the Linkage Editor

Placing linkage editor work files on a 2311 is not recommended.

The linkage editor uses work files as follows: SYSLNK for input and SYS001 for processing. The space requirements are as follows:

- For the SYSLNK file

  Refer to Figure 2-6, which shows how the system makes use of the disk space available as SYSLNK file. Job control writes the input records in blocks of four; a language translator using a different blocking factor or writing to the file unblocked may need some additional space. To give you a feel, the illustration shows the figures for writing unblocked and with a blocking factor of 4 side by side.

- For the SYS001 file

  Any allocation made for SYS001 for use by a language translator is more than enough for use of this work file by the linkage editor.

	Unblocked Records		Records in Blocks of Four	
Device Type	Per Track	Per Two FBA Blocks	Per Track	Per Two FBA Blocks
IBM 2314/2319	39	-	64	-
IBM 3330/3333	60	-	112	-
IBM 3340/3344	34	-	68	-
IBM 3350	72	-	148	-
IBM 3375	75	-	192	-
IBM 3380	83		228	-
IBM FBA disk	-	11		12

Figure 2-6. SYSLNK Space Utilization on CKD and FBA Disks

The Label-Information Area

Entries in the label-information area point the operating system to the related files on disk and, optionally, on tape and diskette. You define the area's residence and, if desirable, also its size by way of the IPL command DLA.

Generally, the default sizes are large enough. However, if your location has many disk files not in VSAM managed space, you should consider defining an area larger than the default size. Figure 2-7 on page 2-19 gives default and maximum sizes by disk device types.

	Default Size		Maximum Size	
	Cylinders	Blocks	Cylinders	Blocks
IBM 2314/2319	2	-	12	-
IBM 3330/3333	2	-	13	-
IBM 3340/3344	3	-	20	-
IBM 3350	1	-	8	-
IBM 3375	3	-	20	-
IBM 3380	2	-	16	-
IBM FBA disk	-	200	-	992

Figure 2-7. Size of the Label-Information Area

VSE/Advanced Functions as shipped includes cataloged procedures for placing standard label information into the defined label-information area. Figure 2-8 lists the files for which these procedures provide label information, their file-IDs and the corresponding logical unit names.

File Name	File-Identifier	Logical Unit Name
IJSYSRS	A5666301.PRODUCTION.LIBRARY	N/A
GENLIB	A5666301.GENERATION.LIBRARY	N/A
IJSYSRC	VSE/AF.RECORDER.FILE	SYSREC
IJSYSCN	VSE/AF.HARDCOPY.FILE	SYSREC
IJSYSHF	A5666301.SYSTEM.HISTORY.FILE	SYSREC
IJSYSLN	VSE/AF.SYSLNK.FILE	SYSLNK
IJSYS01	VSE/AF.WORK-FILE.1	SYS001
IJSYS02	VSE/AF.WORK-FILE.2	SYS002
IJSYS03	VSE/AF.WORK-FILE.3	SYS003
IJSYS04	VSE/AF.WORK-FILE.4	SYS004
BLNXTRN	INFOANA.ROUTINE	SYS025
BLNDMF	DUMP.MANAGEMENT.FILE	SYS026
IJSYSIN	DTTEPTF (see note)	SYSIN

Note: SYSIN labels for SYSRDR and SYSIPT input from a diskette unit (needed for cardless operation).

Figure 2-8. Files With Standard Label Information

The supplied label information assumes that you accept the default library sizes when you restore VSE/Advanced Functions from tape to disk. If you use different sizes, prepare label information of your own and have the system store it in the label-information area.

The *Program Directory* shipped with the program package lists the contents of the IBM supplied label-information procedures. How to store label information is described in the IBM publication *VSE/Advanced Functions System Management Guide.*

The Size of Virtual Storage

The size of a system's virtual storage is defined during system start-up. Give some thought to this size when you plan your system. The method of defining virtual storage is different for the various modes of operation:

- In ECPS:VSE Mode

  If you have an IBM 4341 and use it in ECPS:VSE mode, the size of your system's virtual storage is always 16,384K (16M) bytes.

  If your processor is not an IBM 4341, you define the size of your virtual storage as VSE STORAGE SIZE on the PROGRAM LOAD display during system start up. The system uses the defined value for formatting the page data set.

- VM Mode

  You define your system's virtual storage to VM/SP. This may be up to 16 megabytes.

- 370 Mode

  Virtual storage is composed of a real address space and one to three virtual address spaces.

  The size of the real address space normally is equal to the size of the available processor storage.

  You define the size(s) of your virtual address space(s) during system start-up in your IPL procedure or in response to a system message. This definition is in the form VSIZE=nK or VSIZE=nM; it determines the size of your system's page data set.

  The maximum possible virtual storage size is 40 megabytes. Each of the virtual address spaces may be up to 16 megabytes.

To estimate the *total size of virtual storage*, add the sizes of the following:

Your system's supervisor
Your system's shared virtual area
The system's partitions

The Size of the Supervisor

The size of your system's supervisor depends on the functional support you want to have available and on the scope of this support. Figure 2-9 shows the sizes of IBM supplied supervisors after system start-up with a selected set of options specified by way of IPL commands. The values shown in the illustration should give you a feel for the amount of virtual storage to be reserved for the supervisor.

Make allowance for additional requirements such as space for I/O tables (which is not included in Figure 2-9) or additional support activated during system start-up. Consider adding to the given supervisor size 64K in 370 mode and one or two multiples of 32K in ECPS:VSE or VM mode.

The various support options you can select or suppress are discussed later in the section "The Supervisor" on page 2-26.

The sizes include the support for IPL-defined options as follows:

Option	Command	Operand
Label-information area, 1 cylinder on an IBM 3330	DLA	NCYL=1
The page data set formatted as one extent	DPD	N/A
100 SDL entries for user phases in the SVA	SVA	SDL=100
An SVA phase area of 512K	SVA	PSIZE=512K
100 sublibraries accessible - the default.	SYS	N/A

All values of storage size are given as number of K bytes.

Processor Storage	Virtual Storage	Supervisor Sizes			SVA SIZES		
		MODE=E	MODE=370	MODE=VM	MODE=E	MODE=370	MODE=VM
1,024	16,384 *	272	384	N/A	1,108	1,142	N/A
4,096	16,384 *	296	384	236	1,108	1,142	1,140
8,192	16,384 *	328	448	236	1,108	1,142	1,140
12,288	16,384 *	360	512	236	1,108	1,152	1,140
16,384	16,384 *	392	512	236	1,108	1,152	1,140

* Up to 40,960 if MODE=370.
Same as the storage defined for the virtual-guest machine if MODE=VM.

Figure 2-9. Sizes of IBM-Supplied Supervisors and the SVA After System Start-Up

The Size of the Shared Virtual Area

This area (SVA) accommodates subareas as shown in Figure 2-10.

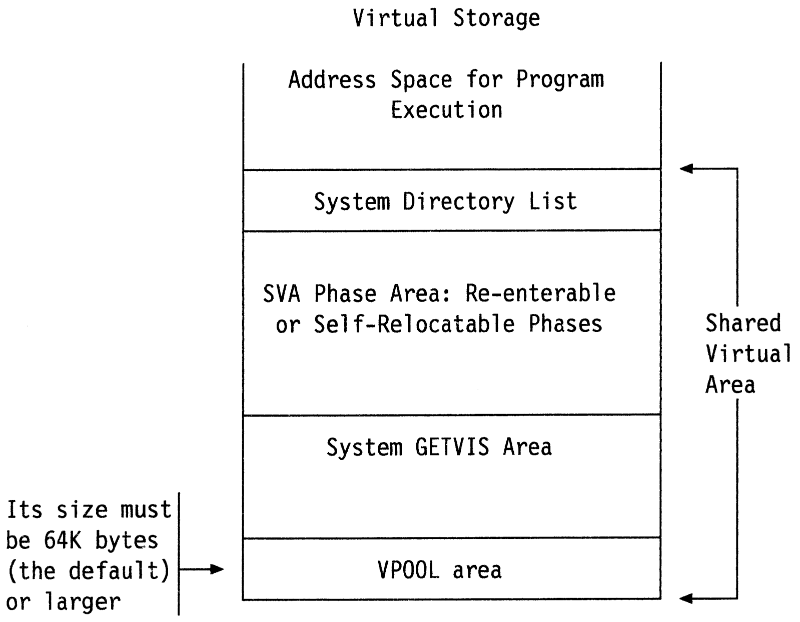

Virtual Storage
Address Space for Program Execution
System Directory List
SVA Phase Area: Re-enterable or Self-Relocatable Phases
System GETVIS Area
VPOOL area
Shared Virtual Area
Its size must be 64K bytes (the default) or larger

Figure 2-10. Layout of the Shared Virtual Area

The size of the SVA is initially set by the system during start-up. You can request the system, also during system start-up, to allocate additional space to the SVA.
However, such space is no longer available as address space for program execution in partitions.

Installed program products such as VSE/VSAM may require additional space in the SVA. The system automatically reserves this space during system start-up if your system library includes load lists for these products. For more information about these load lists, refer to the planning and installation documentation of the products.

Figure 2-9 on page 2-21 shows how much space is allocated to the SVA after system start-up. The values given in Figure 2-9 help you plan the size of your own system's SVA. Add to the values in Figure 2-9 the space you need for SVA-resident phases of your own; make allowance for some extra requirements (a value of about 128K) if the scope of support activated during system start-up is larger than that listed in Figure 2-9.

The System Directory List (SDL): The SDL avoids searching one or more library directories on disk; if a requested phase resides in the SVA, that phase receives control at near page I/O speed.

The subarea contains copies of selected library directory entries. The associated phases may be resident in the SVA or in a library on disk.

The system creates one entry for each phase that it loads into the SVA automatically during system start-up. If you plan to have entries of your own to be stored in the SDL, define their number during system start-up in the SDL operand of the IPL command SVA.

The size of your system's SDL (in number of bytes) is 72 times the number of entries stored in the SDL.

The SVA Phase Area: This subarea contains selected phases of VSE/Advanced Functions and of installed program products. The area may contain phases of your own. A phase that resides in this subarea may be used concurrently by programs in two or more partitions if that phase is re-enterable and relocatable. Having phases in the SVA speeds up processing because it:

- Avoids loading these phases from disk

  A phase resident in the SVA does not have to be loaded from its sublibrary on disk for each execution. This saves disk I/O operations. Even if a requested SVA phase is on the page data set, you save at least one library-directory search operation for the load request.

- Reduces processor storage demands

  One copy of a phase that is shared between two or more partitions has less impact on the page pool than two or more copies of this phase in storage at the same time.

If you plan to have phases of your own reside in the SVA, you define the required space during system start-up in the PSIZE operand of the IPL command SVA. The system loads phases into the SVA on doubleword boundary.

The System GETVIS Area: The system uses this subarea to dynamically acquire virtual storage for its own use. The system reserves a certain amount of GETVIS space based on the options used for supervisor generation and for system start-up.

System GETVIS space needed for program products is to be defined during system start-up in the GETVIS operand of the IPL command SVA. For their space requirements, refer to the planning and installation documentation available with these products.

VPOOL Area: Planning aspects regarding this area is given under "Page Data Set" on page 2-12.

The Size of the Partitions

The space between the lower limit of the SVA (an area discussed above) and the upper limit of the supervisor is available for use by the system's partitions.

You define the amount of virtual storage for your system's partitions in the ALLOC command. This command is normally submitted in the start-up procedure for the system's BG partition. Your ALLOC command must include a value that defines the size of the BG partition.

Virtual Storage for the Partitions

The area you have available for your system's partitions is:

```
The total size of virtual storage as defined
minus:
  The size of the supervisor
  The size of the SVA
```

The total size of virtual storage depends, to a certain extent on your mode of operation:

- In ECPS:VSE Mode

  The maximum size of your virtual storage is 16M (16,384K) bytes. It is always 16M bytes if your processor is an IBM 4341 used in ECPS:VSE mode; else your operator may set the desired value during initial microprogram load (IML).

- In VM mode

  The maximum size you can specify is 16M bytes.

- In 370 mode

  You define the total size of virtual storage in your VSIZE specification. You can specify up to 40,960K (40M bytes).

  The area available for use by the partitions has to be distributed to them in two or three address spaces if your specification is larger than 16,384K. This is discussed under "Partitions in Multiple Address Spaces" below. However, to properly plan the distribution of your partitions over the system's address spaces, you should be familiar with the 370 mode virtual-storage concept as described in the IBM publication *VSE/Advanced Functions System Management Guide.*

Partitions in Multiple Address Spaces

In planning for operation in 370 mode with two or more address spaces, determine, at first, the sizes of your shared partitions; they affect the space available for unshared (private) partitions in your system's address spaces.

IBM program products that must run in a "shared" partition are:

Advanced Communications Function for VTAM (ACF/VTAM)
VSE/POWER
VSE/Operator Communication Control Facility (VSE/OCCF)

Assume you plan to operate with a system that requires storage space for shared areas as follows:

```
Supervisor                  448K
SVA                       4,096K
Partition F1 (ACF/VTAM)   2,048K
Partition F2 (VSE/POWER)  1,536K
Partition F3 (VSE/OCCF)     128K
                          ------
Total                     8,256K
```

This means that 8,256K bytes of each of your address spaces are no longer available for allocation to private partitions. Therefore, you have a total of

```
16,384 - 8,256K = 8,128K bytes
```

per address space for the partitions BG and F4 through FB.

Partition Sizes

The smallest size of a partition is 128K bytes. Certain system functions require a larger partition size, for example:

- The librarian program requires at least 256K bytes;
- The DOSVSDMP program requires at least 166K bytes;
- The partition you want to use for system service under control of MSHP (normally the BG partition) must have size of at least 640K bytes.

The sizes of your partitions must coincide with implementation-defined boundaries. If your values do not fall on these boundaries, the system rounds your specifications to the next higher possible boundary. Your values fall on boundaries if they can be divided by

64K when operating in 370 mode.
4K when operating in VM mode.
2K when operating in ECPS:VSE mode.

Define a partition large enough to accommodate the largest program which is to run in this partition. Add to this size a minimum GETVIS space of

48K bytes. If a program issues many GETVIS requests, that is, requests for dynamic allocation of virtual storage, more than 48K bytes may be needed to run this program.

For applications written in assembler language, the GETVIS macros may be changed to make use of the new GETVIS area subpooling concept. More information about GETVIS area subpooling is given in the IBM publication *VSE/Advanced Functions Application Programming: Macro Reference.*

If you plan to make use of the device-independent full track support, consider an extra requirement for I/O buffers of applicable track-capacity size. You need one such buffer for each of the files that may make use of the support at the same time.

Processor Storage for the Partitions

This is a planning consideration from a performance point of view if you have many programs that either:

Are to be executed in real mode (without paging), or
Fix a relatively large number of pages in processor storage.

Page frames fixed (or used for operation in real mode) are no longer part of the page pool. These frames are not available for use by programs running in virtual mode.

In 370 mode, processor-storage allocations to partitions are done in contiguous areas if these areas are to be used for programs running in real mode (with virtual addresses being the same as the real addresses). These areas are in address space R (for real). As long as no program runs in real mode, the related page frames are part of the page pool.

You define your processor storage requirements by way of an ALLOC command in the system start-up procedure for partition BG.

The Supervisor

The program package you receive includes pregenerated supervisors ready to be loaded during system start-up. For a list of the options used to generate these supervisors, refer to Appendix A. There may be a need for you to tailor your own supervisor. You may, for example, have storage constraints, or the applicable IBM supplied supervisors do not include the required functional support.

This section discusses the options you can specify either as operands in supervisor generation macros or in IPL commands. These options are presented by topics regardless of the involved generation macro or IPL command. In other words, related options are presented together. For the formats of the generation macros, turn to Appendix B; the formats of IPL commands are given in the IBM publication *VSE/Advanced Functions System Control Statements.*

Multiprogramming and Multitasking Options

Number of Partitions: You define the number of partitions which your system is to support in the NPARTS=n operand of the SUPVR generation macro. In selecting this number, consider the kind of applications that have to be processed. Assume, for example that you want to run concurrently the following types of programs:

Test cases (assemble/compile, link edit, and execute)
Daily application programs
On-line program development and data entry
Telecommunication application programs.

For an environment such as this, you should generate a system with at least seven partitions. You may have to define more, if many programs are to be executed at the same time. Telecommunication applications using ACF/VTAM require, for example, one partition for ACF/VTAM and one additional partition for each application using ACF/VTAM. For job-input and -output spooling, handled by VSE/POWER, you need another partition. Your terminal control program (CICS/VS or VSE/ICCF) needs one of your system's partitions.

Since the number of partitions supported by the supervisor cannot be changed without a regeneration run, consider specifying support for more partitions than you currently require.

The IBM supplied supervisors include support for 12 partitions, the maximum number.

Shared Versus Private Partitions: This is a planning consideration if your system is to operate in 370 mode with two or more address spaces. In this environment, the following IBM programs, if used, have to run in shared partitions:

ACF/VTAM
VSE/POWER
VSE/OCCF

Programs of your own communicating with each other across partition boundaries must run in private partitions of the same address space.

Number of Tasks: A supervisor includes support for concurrent attachment of up to 31 subtasks per partition up to a total of 208 subtasks for all partitions. For a discussion of the multitasking concept, see the IBM publication *VSE/Advanced Functions System Management Guide.*

Dynamic Partition Balancing: You may want to make use of the partition balancing support if some of your applications in low priority partitions tend to "sit there for ages." You use the PRTY command to define two or more of your system's partitions for partition balancing. As a result, these partitions receive a near-equal share of processor time.

The system examines each of these partitions' share after the elapse of a pre-set time. The system uses a time value specific to your processor. You can change this value by way of an MSECS command during system start-up or during operation.

The greater a time interval you specify, the smaller is the processing overhead caused by partition balancing. However, a point may be reached where the balancing effect is lost. The optimum value for the time interval depends on location-specific conditions such as job mix and available processor storage.

Library Options

Second Level Directory (SLD): The librarian automatically builds an SLD for the members of type phase stored in the system sublibrary IJSYSRS.SYSLIB. This directory resides in the system GETVIS area, which is part of the SVA. The librarian builds, also in the system GETVIS area, a second-level directory for members of type phase stored in sublibraries referred to in a LIBDEF statement. The librarian's system GETVIS storage requirement is discussed in the next paragraph.

Number of Sublibraries: To define the number of sublibraries that are to be supported, you use the SUBLIB=n operand of the IPL command SYS.

Distributing your on-line program data over a large number of small sublibraries may well result in faster program retrieval. A large number of sublibraries, on the other hand, requires system GETVIS space in the SVA; it may produce extra system overhead in terms of page-in operations and page fixing during program-load operations. The table below relates the number of used sublibraries to required system GETVIS space:

No. of Sublibraries	System GETVIS Requirements
10	51K bytes
100	63K bytes
200	76K bytes
500	116K bytes

Telecommunication Support

Telecommunication support in a VSE system is available through the IBM program products:

Advanced Communications Function for VTAM (ACF/VTAM)
Basic Telecommunication Access Method - Extended Support (BTAM-ES)

ACF/VTAM, in order to run, requires a sufficiently large partition of its own. For estimating the size of this (shared) partition, refer to the product's planning and installation documentation.

BTAM-ES requires a certain amount of processor storage (for page fixing) and adds to the storage requirement of the application program that uses BTAM-ES. For estimating the storage requirements of BTAM-ES, refer to that product's planning and installation documentation.

Operation as a Virtual Machine Under VM/SP

Your VSE system can run as a virtual machine under VM/SP. You use for this purpose a MODE=370 or a MODE=VM supervisor.

Using a MODE=370 Supervisor: Consider using a MODE=370 supervisor under VM/SP if, for example, you want to take advantage of the VAE support also on your virtual VSE. Another possible reason: you may want to use the same supervisor for both stand-alone operation and operation under VM/SP.

Under VM/SP, a MODE=370 supervisor activates VM/SP-related support as follows:

Pseudo page-fault handling
CPCOM/CPCLOSE processing
CP AUTOPOLL ON function
Suppression of paging for pageable supervisor functions

Run your VSE system with the MODE=370 supervisor as a V=R guest machine if there is no need for using the VAE support under VM/SP; this avoids paging under control of VM/SP.

Using a MODE= VM Supervisor: This supervisor includes support which adjusts program execution to the operating conditions for a virtual machine. A MODE=VM supervisor avoids, for example, load leveling and paging and also page fixing and page freeing. This supervisor does not include the related programming overhead.

A VSE system with a MODE=VM supervisor provides subsystem support for the program IBM product VM/VTAM Communications Network Application (VM/VCNA). Through this support, virtual machines under VM/SP have access to the facilities of SNA (system network architecture).

A VSE system with a MODE=VM supervisor can operate only as a virtual machine. It does not support magnetic ink character readers or optical reader/sorters.

Data Protection

VSE/Advanced Functions includes a number of programmed data protection aids. This section discusses these aids and provides planning information as applicable.

Access Control Function

Concepts: You can activate the access-control support during system start-up (by way of the SEC operand in the IPL command SYS). If you do this, the system checks access requests for authority to use the resource that is to be accessed.

The support provides access control for resources as follows:

- Data stored in files – Access to this data can be controlled on a file level.
- Data stored in libraries – Access to this data can be controlled on a library, sublibrary, or member level.

To perform the desired access control, the system needs an access-control table. You can use the DTSECTAB macro for building this table (how to do this is described in *VSE/Advanced Functions System Management Guide*). The system loads the access-control table into the SVA during system start-up; Figure 2-11 on page 2-31 shows how the system makes use of the table. A discussion of specific planning items for access control follows.

Number of User Profiles: To access a protected resource, a program (or terminal user) must supply the correct user-ID and password. The system compares them with the user-IDs and passwords stored in the access-control table. Hence, your system's access-control table must include one entry, a user profile, for every authorized user. For a user, you can define access authority for up to 32 resource classes.

Number of Resource Profiles: You can define the resources to be protected individually or in groups. However, a resource protected individually should not be protected also as part of a group.

Group protection is accomplished by generic specification of associated resources. Example: assume that access control is required for a set of programs named PAYRLIN, PAYRLLST, PAYRLPR, and PAYRLINQ; a resource-profile specification of PAYRL* protects all four of these programs.

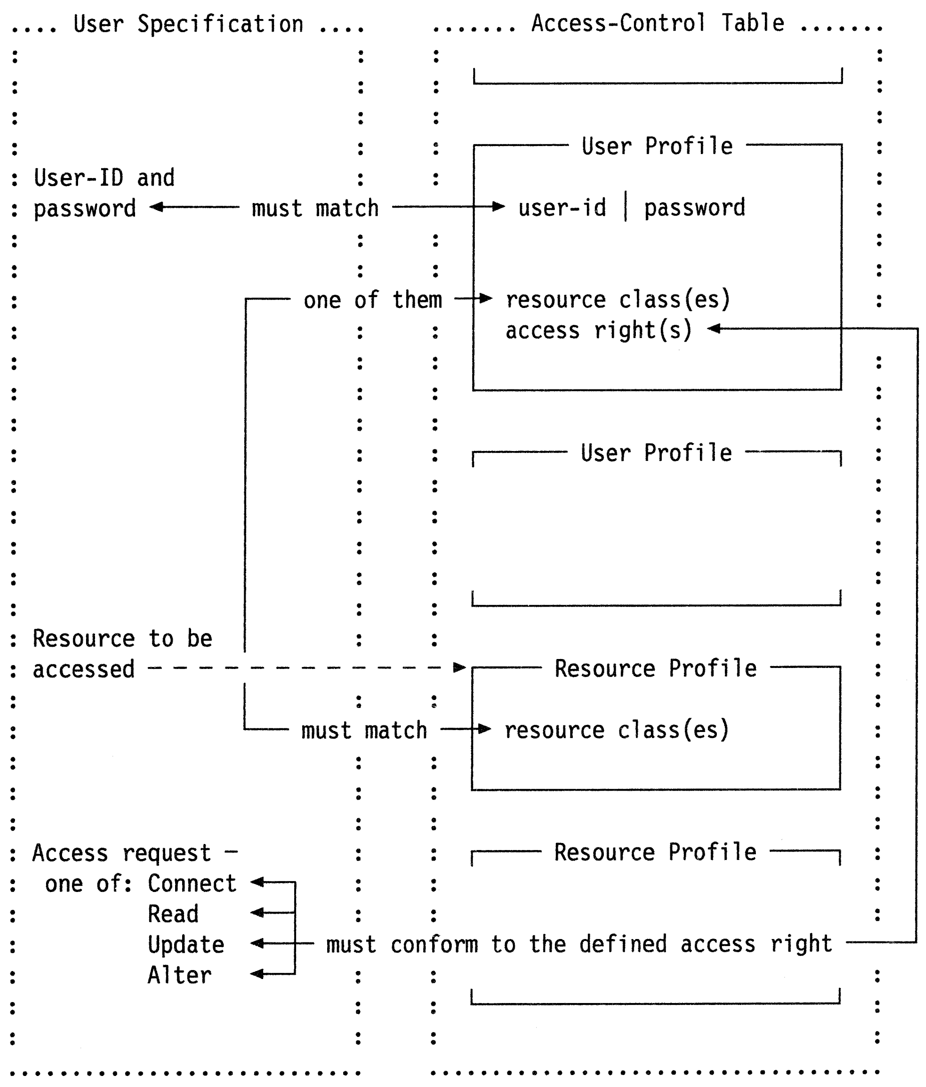


Figure 2-11. Concept of Access Control

The system needs one resource profile per protected resource or group of resources.

Space Requirement: The access-control table requires about 3K of SVA space per 100 profile entries.

Access Rights: One person at your location should be appointed to act as security administrator. Define this person as such to the system by specifying AUTH=YES in the DTSECTAB macro for his or her user profile. AUTH=YES authorizes the person to access and alter any of the system's protected resources, including the access-control table (which should also be access-controlled). For other authorized users, access rights may be defined as shown in Figure 2-12 on page 2-32.

Universal Access Rights: Any of the access rights listed in Figure 2-12 can be defined as universal for a library, a sublibrary, or a library member. If, for example, a universal UPDATE access right is defined for a sublibrary, then any user authorized to use the system can update this sublibrary.

By default (if no applicable DTSECTAB entry exists), the system library has a universal access right of CONNECT and the system sublibrary one of READ. This provides for the most efficient use of the system sublibrary.

Access Right	Library	Sublibrary	Member	File
ALT	Create, delete and rename.	Create, delete and rename.	Create, delete and rename.	Same as UPD.
UPD	Update contents. Create, delete and rename (ALT) sublibraries in it.	Update contents. Catalog, delete and rename (ALT) members in it.	Update contents. Add, delete and replace lines.	Create, delete and rename. Add, delete and change records.
READ	Read only for library and all sublibraries in it.	Read only for sublibrary and all members in it.	Read only.	Read only. (See Note)
CON	Access to sublibraries in it, if user has access right for these sublibraries individually.	Access to members in it, if user has access right for these members individually.	Not Applicable.	Not Applicable.

Figure 2-12. Access Rights and their Scope of Authorization

Note: The access right READ does not allow you to read BAM files. For all accesses to such files, the access right UPD is required.

You may, if this is desirable, protect individual members of the system sublibrary against unauthorized read access. The recommended approach is to define:

1. A universal access right of CONNECT for the sublibrary.

2. A generic universal access right of READ for all members in the sublibrary (by specifying IJSYSRS.SYSLIB.* in a member-type resource profile). This gives read access for all user classes to all members in the sublibrary, except to those members for which a separate user profile is defined (see the next step).

3. READ access right by class in the resource profiles of selected members or groups of members in the sublibrary. This restricts READ access to those members that have a matching class definition.

4. UPDATE access by class in the resource profiles of members that may be updated by one or more users of this class.

Generic definitions may be used in the resource profiles. The generic specification having the largest overlap with the resource name is used for checking. See the IBM publication *VSE/Advanced Functions System Management Guide* for details of generic specification.

Implementation of this approach involves some considerations regarding function and performance:

- System start-up procedures cataloged in the system sublibrary may not be defined as accessible only to a certain user or group of users. In other words, these members must be included in an unrestricted universal READ access definition.

- To keep the processing overhead to a minimum, each job should include a temporary LIBDEF statement for any sublibrary that it needs to access. This method gives access to individual members also to user classes that have a READ or higher access right for the entire sublibrary.

  If you do not supply the LIBDEF statement, then the system's access control function checks the DTSECTAB table on a member level for each member-access request from the job. Therefore, you must ensure that all members stored in the sublibrary are covered by DTSECTAB entries of both user- and resource-profile types.

The system actually denies access to a phase in the system sublibrary in spite of a high individual access right (UPDATE, for example) if:

1. You do not supply the temporary LIBDEF statement for a job, and
2. The phase to be accessed is not defined as accessible by a user of the class specified for the job.

Implementation of the approach may result in slower system performance. This applies, in particular, to jobs which you submit without a LIBDEF statement making the system sublibrary accessible.

General Considerations: Following is a list of IBM programs, by catalog names, which you should consider for protection by the access-control function:

ASSEMBLY	Assembler program
CLRDK	Clear Disk utility
DITTO	VSE/Data Interfile Transfer, Testing and Operations utility
DOSVSDMP	Dump utility
FCOPY	VSE/Fast Copy Data Set program (for CKD disks)
FCOPYFB	VSE/Fast Copy Data Set program (for FBA disks)
ICKDSF	Device Support Facilities
IDCAMS	VSE/VSAM's Access Method Services program
IKQVEDA	VSE/VSAM service program
IKQVDU	VSE/VSAM service program
INFOANA	Information/Analysis
INTTP	Initialize Tape program
LIBR	Librarian program
LNKEDT	Linkage editor program
LVTOC	List Volume Table of Contents program
MSHP	Maintain System History program

Consider, in addition, the following as a data protection measure at your location:

- Provide for logging and reporting of access to protected resources. This is further discussed under "Logging and Reporting" below.

- Have your security administrator keep the following under lock and key:

  - The DOSVSDMP-generated stand-alone dump tape.

  - The IBM-supplied distribution tape, which includes a copy of:

    The stand-alone Device Support Facilities (ICKDSF) program.
    The stand-alone Fast Copy Disk (FCOPY) program.

- Have programs executed with NODUMP specified in the // OPTION statement if they process sensitive data. This avoids exposure of such data should a cancel condition occur.

- Programs running from system start-up to shutdown (VSE/POWER or ACF/VTAM for example) may, at times, hold sensitive data in their buffers. If this is the case at your location, instruct your operator not to request a dump of storage should such a program fail. The recommended approach is to reproduce the error with no sensitive data being processed and then have the operator request a dump of storage.

Logging and Reporting: A separate program product, IBM VSE/Access Control—Logging and Reporting (ACLR), is available. If this program is installed and active (in a partition of its own), the system can log any and all accesses to protected resources into a log data set on disk.

The program, which your security administrator can use for auditing the access to protected resources, can run in a partition of smallest size. For more information about the program, refer to *IBM VSE/Access Control – Logging and Reporting General Information.*

IPL User Exit

At the end of the IPL procedure, the system passes control to a routine cataloged in the system library as phase $SYSOPEN. This is a dummy phase for VSE/Advanced Functions as shipped, and you may replace the phase by a user-written exit routine. For information about coding this routine, see the IBM publication *VSE/Advanced Functions System Management Guide.*

Consider using this exit for performing certain integrity and security checks. For example:

- Whether the right system volume was mounted.
- Whether the correct date was entered for the new work session. This is important, for instance, when you work with labeled files because a correct creation date ensures protection until the expiration date for an affected file.
- Whether the operator has entered the correct security code in response to a message to the console. The routine could provide for two retries (in case of a typing error). It could be coded to go into a continuous loop if a wrong security code is submitted on third try. In this loop, the routine could write blank line after blank line to the console. To get out of the loop, the operator would have to reset the processor and perform system start-up from the beginning.

Label Records

Using label records (or just "labels" for short) helps you to ensure that the correct data volumes are mounted for processing. It assists in protecting data against inadvertent destruction and, to a certain extent, against unauthorized access and use.

You can use labels to protect files stored on magnetic tape, on disk, or on diskette. Each file on a volume has one or more file labels; each volume has one volume label. Volume and file labels are mandatory for disks and diskettes, they are optional for tapes.

The system's label processing routines check whether the correct volume is mounted and whether the file to be accessed is still active. This protects active, labeled files against being overwritten and destroyed.

The TLBL statement specifies label information for a file on magnetic tape; the DLBL and EXTENT statements specify this information for a file on disk or on a diskette. For more information about submitting label information, see the IBM publication *VSE/Advanced Functions System Management Guide.*

Resource Locking

In a multitasking operation, a mechanism is needed to prevent two or more tasks from using the same resource in an uncontrolled way. The system's lock-management support protects user-defined and system resources against concurrent use by different tasks in different partitions of the same or of sharing systems.

Two levels of sharing can be controlled using this support:

- Exclusive use of a specific resource.
- Shared use of a specific resource.

The following types of resources can be protected by this support:

Disk volumes
Libraries
Files
Control blocks

For the resources to be protected, you should set up symbolic names that are to be used in the programs which access these resources. A file name, a volume serial number, or a disk start address are examples of suitable symbolic names.

The support requires a lock-control block for each resource whose use is to be shared. You define this block by a DTL or a GENDTL macro.

A successful lock request (via the LOCK macro) means that the resource is locked for the task or partition which issued the request. By issuing the UNLOCK macro in the program, you can either release the resource completely or, together with the MODDTL macro, lower the lock-control level from "exclusive use" to "shared use".

For more information about using the various macros, see the IBM publication *VSE/Advanced Functions Application Programming: Macro Reference.*

DASD File Protection

This support prevents programs from reading or writing data outside the limits of their disk files. DASD file protection, which you can activate during IPL, is discussed in more detail in section "DASD File Protection" on page 2-39.

Track Hold (TRKHLD) Option

The track-hold support prevents two or more programs (or two or more tasks) from updating data on the same track of a CKD-type disk or within the range of accessed blocks on an FBA-type disk. The track-hold support, a supervisor generation option, is discussed in more detail in section "Track Hold Option" on page 2-40.

Data-Secured Files

By coding the DSF operand in the DLBL statement, you indicate that the associated file is secured. If this file is to be accessed, the system issues a warning message at the console. The operator then has to decide whether the file can be accessed as requested. The total of these warning messages would make up a record of accesses to secured files.

IBM recommends that you use the access-control function to protect the file.

Job Accounting

You can request the system to provide accounting information on a job and job-step level. This information is useful for charging system use, monitoring system utilization, planning new applications, and so on. You activate this support during system start-up by specifying JA=YES in the IPL command SYS.

The support causes job tables to be built, one per partition, and accounting data to be accumulated in those tables.

You can write and catalog a routine that stores or prints this data. As an alternative, you may make use of the job accounting support available through VSE/POWER.

For more information about using the available job accounting support, see the IBM publication *VSE/Advanced Functions System Management Guide* or, if VSE/POWER is used at your location, *VSE/POWER Installation and Operations Guide.*

Timer Services

The following timer services are available:

Time-of-day clock
Interval timer
Task Timer

The support for interval-timer and task-timer require the clock comparator and the CPU timer, which are optional for System/370 Models 135 and 145.

Time-of-Day Clock: The time-of-day clock (TOD), a standard hardware component in all IBM processing units, provides a consistent measure of elapsed time; it is suitable for a time-of-day indication when required. Making use of this support requires no planning.

Interval Timer: Your programs can use this timer through available macros. They request the system to cause an external interrupt when the specified time has elapsed. Making use of this support requires no planning.

Task Timer: This service can be used only by the main task of the partition that owns the timer. The task can use the timer to escape from processing and enter an exit routine after a specified period of time. The system decrements the specified time interval only while the owning task is active.

The IBM supplied supervisors do not include this support. If the support is required, generate a supervisor of your own with the TTIME operand specified in FOPT.

For information about using this support, refer to the IBM publication *VSE/Advanced Functions Application Programming: Macro User's Guide.*

Disk-I/O Options

Disk-I/O oriented options are available as follows:

DASD sharing across systems
DASD file protection
Track hold
Rotational position sensing

DASD Sharing Across Systems

IBM supplied supervisors include this support. It allows two or more (up to 31) VSE systems to be linked such that they share one or more disk volumes.

Sharing disk volumes across systems requires resources accessed from one system to be protected against interfering access from another. Support for this kind of resource control is established if each sharing system runs under a supervisor generated with DASDSHR=YES in FOPT. The support requires a lock-communication file, which is discussed under "Lock-Communication File" on page 2-14.

For disk data to be shared, the following requirements must be met:

- If the system operates stand-alone (not under VM/SP):

  The data to be shared must reside on an IBM disk drive that is accessible by the sharing processors. Normally, this requires a two-channel switch on the control unit or a string switch on the string.

- If the system operates under control of VM/SP:

  The disk drive with the lock file must be defined to VM/SP with virtual RESERVE/RELEASE support. This enables two or more VSE guest machines to share this file.

DASD File Protection

IBM supplied supervisors include this support.

A program may produce addresses outside the limits of the currently accessed disk extent if:

It uses DAM of LIOCS or user-written channel programs.
It includes a randomizing access algorithm.

The support prevents read or write access to locations outside the currently accessed disk extent as follows:

- For an extent on a CKD disk – a program can neither read from nor write to a location outside the extent.
- For an extent on an FBA disk – a program cannot write to a location outside the extent. A program can read from a location outside the extent if:
  - The program accesses the disk data using user-written channel programs.
  - These channel programs include Define Extent commands with all write operations inhibited in the mask byte of the command parameters.

You activate this support by specifying DASDFP=YES in the IPL command SYS.

DASDFP gives protection based on programmer logical units. If two disk files are open in the same partition and use the same programmer logical unit, the DASDFP option does not give any protection to either of the two files.

If a program uses physical IOCS (it includes user-written channel programs), the following must be observed:

- The disk file to be accessed is to be defined by a DTFPH macro.
- The file must be opened using the OPEN or OPENR macro.
- Each channel program must begin with a long seek (X'07') command or a define extent (X'03') command, and it may not contain chained long seeks.

Notes:

1. *For CKD devices, no protection is given to partially allocated cylinders. Files to be protected must begin and end on cylinder boundaries.*
2. *Activating the support does not prevent file contention between partitions or within partitions if the same logical unit is used. Thus, more than one partition may access the same file at the same time and may even attempt*

to update the same record at the same time. The track hold option (TRKHLD – see below) can solve problems of this kind.

Track Hold Option

IBM supplied supervisors include track-hold support, allowing 12 disk areas to be held at a time.

If properly used, the support protects data on disk against being updated by more than one active task in the system at the same time. You obtain this support by specifying TRKHLD=n in FOPT (where n is the number of hold requests that may be active at a time).

The support is invoked from within the program that makes use of track-hold protection. However, for the support to be effective, all programs accessing the same file must request this protection.

For FBA devices, the support protects the range of blocks which contains the accessed data; for CKD devices, it protects the track that contains this data.

A deadlock occurs if one task is waiting for a disk area held by another task and that other task is waiting for a disk area held by the first. To avoid this, ensure that every task is programmed such that it does not attempt to hold more than one disk area at a time. A deadlock may also occur if the number of disk areas requested to be held at a certain point in time exceeds the number specified in the TRKHLD operand of FOPT.

Rotational Position Sensing (RPS)

The RPS feature is not available for IBM disk devices of type 23xx; it is optionally available on IBM 3340s.

IBM supplied supervisors include RPS support.

The feature provides for overlapping positioning operations on one device with requests for other devices on a block multiplexer channel. Your VSE system makes use of the feature if you specify RPS=YES in FOPT.

The feature improves channel utilization which, in turn, can increase system throughput. This is particularly true for large systems with much concurrent I/O activity. A disk device attached to a block multiplexer channel and equipped with RPS can disconnect from the channel during positioning operations. This channel can then handle requests for accessing other devices on the channel. Efficient use of RPS depends on each channel program's ability to free the accessed channel as early as possible.

To make efficient use of RPS support, user-written channel programs for access to a disk device with RPS will have to be recoded. These channel programs must include set sector CCWs and establish search arguments for the CCWs. If this is not done, programs with unchanged channel programs monopolize the channel and destroy the effectiveness of RPS.

For more information about RPS and its use under the VSE system, refer to the IBM publication *VSE/Advanced Functions Application Programming: Macro User's Guide.*

I/O Options

The available I/O options allow you to define the size of the channel queue and of various I/O-related buffers and tables.

The Channel Queue

The system uses a channel queue to schedule I/O operations. The system builds a 32-byte entry in the channel queue whenever a request for an I/O operation occurs. This entry remains in the queue until the operation is complete.

On completion of an I/O request, the system examines the queue to see whether another entry exists for the same channel. If so, the system initiates the operation requested by this other entry.

The number of channel queue entries to be allocated can be specified in the CHANQ=n operand of the IPL command SYS. By default, an IBM supplied supervisor includes a channel queue of either 255 entries or of n entries according to the formula below, whichever is the smaller. The formula is:

```
n = (2 + d) x 12 + u + 15

where d = number of disk devices
      u = number of I/O units defined by IPL ADD commands
```

No accurate formulas for determining the optimum size of this queue can be given. Specifying too small a channel queue may cause performance degradation, too large a channel queue causes a waste of storage space.

If the default results in acceptable system performance, no additional tuning of the value is necessary.

If you have performance problems, increase the size of the channel queue to a value between **n** in the above formula and 255.

Supervisor Buffers for I/O Processing

Supervisor buffer space is used for handling I/O requests from programs that run in virtual mode. You specify the number of buffers during system start-up in the BUFSIZE=n operand of the IPL command SYS. The amount of required buffer space depends on the number and type of concurrent I/O requests.

Generally, four times the number of specified channel queue entries gives a sufficient number of buffers. If your system is to operate with the default number of channel queue entries, IBM recommends that you specify 1000 initially and a larger value for system start-up later on, should there be a need.

Given below are the smallest values for a BUFSIZE = n specification and the default values set for an IBM-supplied supervisor:

	Default Value	Smallest Value
FASTTR=NO	60	10
FASTTR=YES		
in 370 mode	260	-
in ECPS:VSE mode	520	-

In a system operating in 370 mode, a buffer has a size of 72 bytes; in ECPS:VSE mode, this size is 36 bytes.

Fast-CCW-Translation or Fast-Function

This support is available only for operation in 370 mode or in ECPS:VSE mode. IBM supplied supervisors for these modes of operation include the support.

If you generate your own supervisor, you request the support by specifying FASTTR = YES in FOPT. This causes the generated supervisor to provide:

- Fast-function support if you specified MODE = E in SUPVR.
- Fast-CCW-translate support if you specified MODE = 370 in SUPVR.

The support works essentially the same way in either mode. The supervisor buffers used for servicing an I/O request are not released when this request is completed. Instead, the system saves the buffers and fixes the involved I/O areas in processor storage until the end of the job. This can speed up I/O processing if your program has frequent repetitive I/O requests. The overall effect on your system depends on various factors, however:

- The page pool is decreased in size because the I/O areas remain fixed.

- Additional supervisor buffers are required. As a rule of thumb, define additional buffers as follows:

  2000 for fast function use in ECPS:VSE
  1500 for fast CCW translate use in 370 mode.

If you do not specify enough buffers, or if the page pool becomes too small, the saved buffers and fixed I/O areas are released as the demand for page frames goes up.

The function can be suppressed just for the duration of a specific job (by // OPTION NOFASTTR); this is meaningful if, for example, a job is unlikely to reuse buffers and fixed I/O areas. The function can be suppressed completely for all jobs (by STDOPT FASTTR = NO); this can be useful if none of your programs gain by the fast CCW translate function being active.

Notes:

1. *In 370 mode, the system does not attempt fast translation of a channel program if the channel program:*

   Contains non-contiguous CCW strings.
   Includes translation requests from BTAM-ES.

2. *Specification of FASTTR=YES may result in slightly degraded performance if CICS/VS accesses SAM, ISAM, and DAM files.*

Device-Related Areas

The IOTAB generation macro requests space to be reserved for I/O-related tables. The involved operands refer to:

- The number of programmer logical-unit names to be reserved.
- The number of I/O devices that are part of your computer system.

IBM supplied supervisors include support for 254 I/O devices (the maximum) and for a pool of 1000 programmer logical units.

Before you can actually use your *I/O devices*, you must define each unit to the system, specifying characteristics such as channel and unit address and device type. You do this via the IPL ADD command during system start-up.

Since the number of attached I/O devices normally remains stable over a longer period of time, give this matter a thought already when you plan your system. The total number of devices defined by ADD commands may not exceed the total number of devices that you specify in IODEV=n of IOTAB.

Programmer logical-unit names used in programs are to be assigned to I/O devices by job control statements or commands. You use the operand NPGR of IOTAB to define the total number of programmer logical units that are to be available to the total of your system's partitions at any point in time.

This ends the discussion of system planning. However, if you migrate to this Version of VSE/Advanced Functions from a Version-1 release or a release of DOS/VS, read the next chapter, Chapter 3. This helps you plan your migration activities.

Chapter 3. Migration to Version 2 of VSE/Advanced Functions

Migration to Version 2 has two major aspects: program compatibility and conversion of libraries.

Program Compatibility

This section discusses program compatibility for groups of migrating users as follows:

- From a previous VSE release

  This is the group of migrators that have used a Version-1 VSE System IPO/E release or installed their system by individual program products based on a Version-1 release of VSE/Advanced Functions.

- From DOS/VS

- From DOS (Disk Operating System)

The chapter also discusses compatibility considerations for users of one of these operating systems under VM/SP.

Migrators from Version 1 of VSE/Advanced Functions

An *application program* that has run successfully on a Version-1 release will run successfully also on the new Version-2 release if this program:

- Interfaces with the operating system through IBM provided macros or through high-level language statements of a supported compiler.

- Does not access unsupported I/O devices.

For a program that was written for execution under Release 1 of VSE/Advanced Functions with the DOS/VSE SCP, the following additional restrictions apply:

- The program must not alter its DTF blocks generated for one of the following file definition macros:

DTFDA (direct access files)
DTFMT (tape files – see also "Note" below)
DTFPH for a tape file
DTFPR (print files)
DTFSD (sequential disk files)

Note: Following is a list of exceptions:

- The address of the end-of-file routine (EOFADDR=name).
- The rewind and unload indicators (REWIND=specification)
- The address of an error routine (ERROPT=name or WLRERR=name).
- The address of the label-processing routine (LABADDR=name).

- If the modules of the program include one or more DTFMT macros and the program need not be reassembled prior to a necessary link run, you may get the message

  ```
  UNRESOLVED EXTERNAL REFERENCE IJFxxxxx
  ```

  Ignore this message, except if the program uses the CNTRL macro for the affected tape file before this file is opened. To avoid any problem if this situation exists, run a job similar to the one below and re-link your program.

  ```
         // JOB     CATALOG IOMOD
         // EXEC    LIBR
  (1)       ACCESS  libname.sublibname
  (2)       CATALOG modulename.OBJ
            INCLUDE IJJTCTL
         .END
         /*
         /&
  ```

  ---

  (1) ACCESS libname.sublibname
  For libname.sublibname, give the qualified name of the sub-library into which your program is to be link-edited.

  (2) CATALOG modulename.OBJ
  For modulename, specify the name that was displayed in the UNRESOLVED EXTERNAL REFERENCE message.

- A program with a DTFPR macro for output to an IBM PRT1 printer or an IBM 3800 should be reassembled if the macro specifies first-character control (CTLCHR=YES or CTLCHR=ASA). If you do not reassemble the macro, the system's action is as follows:

  - If the first character of the first output record is a valid control character as defined by the American National Standards Institute (ANSII), the system uses the ANSII character set.
  - If the first character of the first output record is not a valid ANSII control character, the system uses the S/370 character set.

- If the program was written to depend on processing priorities set during supervisor generation, this program requires a similar setting of the processing priorities during system start-up.

- If the program was written in FORTRAN language, this program may have to be re-linked with the library of the new VS FORTRAN compiler (except when they have been re-linked previously with Release 3 of DOS FORTRAN Library Option 1).

Asynchronous operator communication, a supervisor generation option in Version 1, has become standard. If you have not used this option in the past, examine programs that write to or read from the SYSLOG device on an EXCP level. For these programs, SYSLOG-access restrictions exist as follows:

- The system does not return sense information, although this may be requested in the CCB macro.

- The system ignores a request for posting the CCB at device end.

- The system does not permit:

  - More than 31 CCWs chained together in one channel program for one SYSLOG operation.
  - A SYSLOG channel program that starts with a sense CCW, followed by a read CCW.
  - SYSLOG channel programs with CCWs that refer to indirect data address lists.

- A program issuing a SYSLOG read request before a preceding read request is completed may impact system throughput.

- A channel program for SYSLOG may fail if it uses TIC CCWs in combination with data chaining.

To avoid problems, IBM recommends that SYSLOG-access routines on an EXCP level be replaced by routines using the LIOCS macros DTFCN and PUTR. For more information about these macros, see the IBM publication *VSE/Advanced Functions Application Programming: Macro Reference.*

System start-up (ASI) procedures used in the past cannot be used under VSE/Advanced Functions 2.1 unchanged. An ASI IPL procedure is affected by changes to IPL commands as listed below:

Command	New in VSE/Advanced Functions 2.1
DEF	Operand SYSDMP={cuu|volid} The operand has been deleted.
SYS	Operand EXTENT=mK The operand has been deleted.

Operand CHANQ=n
This was a supervisor-generation option in the past; it causes space to be reserved for channel queue entries.

Operand DASDFP=YES
This was a supervisor-generation option in the past; it causes the DASD file-protect support to be activated.

Operand JA=YES
This was a supervisor-generation option in the past; it causes the job-accounting support to be activated.

Operand PAGEIN=n
The operand has been deleted.

Operand SEC=YES
This was a supervisor-generation option in the past; it causes the access-control function to be activated.

Operand SDSIZE=nk
A new operand; it causes processor storage to be reserved for the use of SDAID, a program for tracing programmed events.

The *programs* listed below are *no longer available.* If you have used the programs at your location, consider taking the action recommended as a bypass.

- ALTBLK

  Bypass: use the INSPECT function of the Device Support Facilities program.

- BACKUP

  Bypass: use the librarian's BACKUP function.

- COPYSERV – Copy Service

  Bypass: use the librarian's COMPARE function, a one-command service request.

- INTDK

  Bypass: use the INIT function of the Device Support Facilities program.

- PDZAP — Core Image Library Patch

  Bypass: use the librarian's UPDATE function to change the on-line source code; recompile (reassemble) and re-link your program. For phases stored under control of MSHP, use MSHP's CORRECT function.

- RESTORE

  Bypass: use the librarian's RESTORE function.

- SURFANAL

  Bypass: use the INIT function of the Device Support Facilities program.

Version 2 of VSE/Advanced Functions has a *new library support.* The involved migration effort is discussed in section "Conversion of Libraries" on page 3-8.

Link-and-go compilations or assemblies are restricted to one-phase programs. If this is a problem, catalog a multi-phase program in a test library.

Migrators from DOS/VS

Compatibility prerequisites listed above for users migrating from a previous VSE release apply also to you if you are migrating from DOS/VS. In addition, you should ensure that a program to be migrated to the current release of VSE/Advanced Functions:

- Has not been linked to an absolute address – If it has been, relink the program to obtain a relocatable version of it.
- Does not contain IDA lists if your VSE system is to operate in ECPS:VSE mode – If the program does, change it to replace the affected CCWs.
- Does not access libraries by user-written library-access routines – If it does, change the program to perform its functions without this access.
- Does not access the interval timer at location 80 (X'50') or the system time of day at location 84 (X'54') – If it does, change the program to use the GETIME macro instead.
- Does not include BTAM modules – If it does, re-link the program with a new BTAM–ES module that has been assembled using the current BTAM–ES BTMOD macro.
- Does not interface with QTAM (queued telecommunication access method) – If it does, change the program to use BTAM–ES or ACF/VTAM instead.
- For a DTFSD file or a DTFDI file on disk, does not include an error-exit routine that issues one or more imperative macros (such as GET or PUT) for this file – If it does, change the program to avoid imperative macros in the exit routine.
- Does not process data stored on mixed-parity (IBM 1401-processed) tapes – If it does, the program has to be rewritten.

- Does not make use of any of the following DOS/VS components:

  - System/360 Model 20 emulator on System/370
  - Deblock system utility – Use the deblock function of the OBJMAINT program instead.

  These components are not part of VSE/Advanced Functions. If a program makes use of the components, it has to be rewritten.

Migrators from DOS

DOS users who plan to change to a VSE system will have to consider the following in addition to the compatibility requirements given for migrators from DOS/VS:

- Files created under DOS can be processed on a VSE system if compatible I/O devices are used:

  - Programs written to process data for an IBM 2311 can be executed to process the same type of data through the use of a 2311 compatibility feature, if this is installed.
  - Programs written for I/O to and from an IBM 1052 console printer-keyboard can be executed if your processor has a 1052 compatibility feature installed.

- Existing assembler language source programs can be processed by the assembler available with the VSE system, provided that no user-written macros are called. If any of your programs calls such macros, then either:

  - Supply COPY instructions for the macro definitions at the beginning of all source decks in which the macros are used, or
  - Convert your library macros to edited macros and include them in the macro sublibrary.

- A previously compiled or assembled object program can be link-edited without modification, provided the program:

  - Accesses devices that are available on the computer system under VSE control.
  - Does not depend on processing unit circuitry for which there is no support in processors of the System/370 architecture or in processors with ECPS:VSE.
  - Is not processing-time dependent – you should execute such a program in real mode (without paging).
  - Does not force program checks.

- A program that was not written self-relocating has to be re-linked to make it relocatable.

- A user-written appendage routine must either run in real mode, or a program using such a routine is to be rewritten to eliminate this appendage routine.
- If a program modifies a channel program dynamically, you have these alternatives:
  - Run the program in real mode.
  - Rewrite the program using the PFIX macro (to fix the channel program in processor storage) and the EXCP macro specifying REAL.
- The DOS format label-information statements listed below are invalid; they have to be replaced by valid statements as indicated:

DOS Format	Replace by
VOL	–
DLAB	DLBL
XTENT	EXTENT
VOL	–
TPLAB	TLBL

Users in a Virtual Machine Environment

The operating characteristics of a VSE system under VM/SP are largely the same as those of a VSE system that operates stand-alone on a real machine. Therefore, the compatibility considerations which apply to migrators in a real-machine environment apply also to migrators on a virtual machine. However, the following should be considered:

- A program that functions correctly under VM/SP and a MODE=VM supervisor may fail if a supervisor generated for stand-alone operation is used.

  An example is the execution of a program that issues more than 255 PFIX requests for a page without a PFREE request for that page in between. This program functions correctly if it is executed in a virtual machine environment using a MODE=VM supervisor; the program fails when it runs under VSE/Advanced Functions in a stand-alone environment.

- A program that functions correctly on a real machine can be executed successfully also on a virtual VSE system with a supervisor for virtual operation; the reverse is not necessarily true.
- If your VSE has been generated to operate as a virtual machine, then you cannot use the VSE-generated stand-alone dump program; you have to use the CP dump of VM/SP.

Conversion of Libraries

Version 2 of VSE/Advanced Functions includes various library migration aids. These aids are discussed in context with the migration steps that make use of them.

There is no need to convert system libraries. However, if your current (pre-version-2) system libraries include user-written programs, you may have to spend some extra time and effort for migrating these programs.

There may also be a need to adapt the names of Version 1 object modules to the Version 2 librarian naming conventions. In Version 2 of VSE/Advanced functions, all member names must be alphameric. If any existing modules do not conform to this convention, you must rename them **before** migrating. The Version 2 librarian program does not handle member names in any other form, so a rename after migration is impossible.

The conversion of private libraries (and also of programs that may currently be cataloged in your system libraries) requires two basic steps:

- A format conversion.
- A conversion of existing job streams.

It may be desirable to use library members created under Version 2 of VSE/Advanced Functions on a computer system operating under control of Version 1. Version 2 includes support for the conversion of library members to the Version-1 library format.

Before you read this section, consider reading either or both of the following:

The chapter "Using the Libraries" in the IBM publication *VSE/Advanced Functions System Management Guide.*

The description of the librarian commands in the IBM publication *VSE/Advanced Functions System Control Statements.*

Format Conversion to Version 2

When you convert a library of the current format – a relocatable library, for example – to the Version-2 format, the Version-2 librarian converts it to a sublibrary of a defined library. If it is desirable, the librarian converts two or more libraries of the pre-version-2 format to just one sublibrary.

For converting your private libraries you can use two distinct methods:

- The restore-conversion method

  You create one or more backup tapes of your existing (pre-version-2) libraries under your current system. When the new (Version-2) system is operational, you restore these libraries under this system, using the librarian's RESTORE function. This is the recommended method for converting complete libraries; you can use this method if the current release level of your system is VSE/Advanced Functions Version 1 or earlier.

- Conversion via output to SYSPCH

  This is the recommended method for converting individual library members (user-written phases contained in the current system's system core image library, for example). It is the method you have to use if your current system's release level is earlier than DOS/VS Release 34.

Restore-Conversion Method

Proceed as follows:

1. Create a backup tape under the current system

   To do this, use the available BACKUP utility program. Refer to sample jobs provided as follows:

   - For creating the backup tape by way of the BACKUP statement method – see Figure 3-1 on page 3-10.

   - For creating the backup tape by way of the ASSGN statement method – see Figure 3-2 on page 3-11. When migrating from a release of the system older than Version 1.2 you must use this method.

   The sample jobs assume CKD-type disk devices. If FBA disks are used at your location, you have to define the libraries in the EXTENT statement by relative block number and number of blocks rather than relative track number and number of tracks.

2. Restore the libraries backed up by step 1

   You do this under control of the system's librarian after having installed and started VSE/Advanced Functions 2.1. A sample job stream for a restore run is given in Figure 3-3 on page 3-12. Run the restore job in the background partition. Restoring your libraries to disk is a step of the installation procedure for VSE/Advanced Functions 2.1.

```
    // JOB CREATE BACKUP
    // DLBL PGMDEV1,'COSTCTL.SYSTEM.PROJ1'
    // EXTENT ,PGMDEV,1,0,19,570
    // DLBL PRODVL1,'PAYROLL.APPLIC.PROGS'
    // EXTENT ,PRLIB1,1,0,19,95
    // DLBL PRODVL2,'INVENT.BILLING.PROGS'
    // EXTENT ,PRLIB2,1,0,19,114
      ... ┐
      ... │     Additional sets of DLBL and EXTENT statements
      ... ┘
(1) // ASSGN SYS006,281
    // EXEC BACKUP
       NOSA
(2)    BACKUP CL SEARCH=(PGMDEV1 PRODVL1 PRODVL2 ... )
       BACKUP RL SEARCH=(...
       BACKUP SL SEARCH=(...
    /*
    /&
```

---

(1) // ASSGN SYS006,281
Assign SYS006 to the tape drive on which you mount the output tape.

(2) BACKUP CL SEARCH=(PGMDEV1 PRODVL1 PRODVL2 ...)
You may specify as many private libraries of the same type as names fit on one statement line, but no more than 15. Supply additional BACKUP statements, if necessary.

Figure 3-1. Creating a Backup Tape for Migration - BACKUP-Statement Method

Assumption: A backup tape is to be created of a core image, a relocatable, and a source statement library, all of them related to a program development project.

```
    // JOB CREATE BACKUP
(1) // DLBL IJSYSCL,'COSTCTL.SYSTEM.PROJ1'
(1) // EXTENT SYS007,,1,0,19,551
(1) // ASSGN  SYS007,131
    // DLBL IJSYSRL,'PAYROLL.APPLIC.PROJ1'
    // EXTENT SYS008,,1,0,570,475
    // ASSGN  SYS008,131
    // DLBL IJSYSSL,'INVENT.BILLING.PROJ1'
    // EXTENT SYS009,,1,0,1045,1083
    // ASSGN  SYS009,131
(2) // ASSGN SYS006,281
    // EXEC BACKUP
       NOSA
    /*
    /&
```

```
(1) // DLBL IJSYSCL,'COSTCTL.SYSTEM.PROJ1'
    // EXTENT SYS007,,1,0,19,551
    // ASSGN  SYS007,131
```
The set of DLBL, EXTENT, and ASSGN statements which identifies the private core image library named COSTCTL.SYSTEM.PROJ1.

```
(2) // ASSGN SYS006,281
```
Assign SYS006 to the tape drive on which you mount the output tape.

Consider assigning an alternate tape unit to ensure uninterrupted copy-to-tape operation by the BACKUP program.

Figure 3-2. Creating a Backup Tape for Migration – ASSGN-Statement Method

The pre-version 2 produced backup tape is assumed to be mounted on the drive at address 281.

```
    // JOB     LIBRARY CONVERT-RESTORE - FROM BACKUP
(1) // OPTION  STDLABEL=ADD
    // DLBL    DEVLIBR,,99/365
    // EXTENT  ,DEVLIB,,19,7657
    // DLBL    PRODLIB,,99/365
    // EXTENT  ,PRDLIB,,19,7657
    // OPTION  USRLABEL
    // EXEC    LIBR
(2) DEFINE  LIB=(DEVLIBR PRODLIB)
(3) RESTORE OLDLIB=(PGMDEV1:DEVLIBR.PGMDEV1 -
                    PRODVL1:PRODLIB.PAYRLLPR -
                    PRODVL2:PRODLIB.INVBLLPR ...) -
            TAPE=281 LIST=YES
    /*
    // MTC RUN,281
    /&
```

---

(1) // OPTION STDLABEL=ADD
Causes the subsequent DLBL and EXTENT statements to be stored in the system's label-information area, accessible from any partition.

(2) DEFINE LIB=(DEVLIBR PRODLIB)
Formats the libraries defined by the preceding DLBL/EXTENT statements.

(3) RESTORE OLDLIB=(PGMDEV1:DEVLIBR.PGMDEV1 ...) TAPE=281 LIST=YES
There is no need to define sublibraries; the librarian creates them as required. In this example, the librarian creates sub-libraries and restores pre-version 2 private libraries as follows:

Sublibrary	In Library	Pre-Version 2 Library
PGMDEV1	DEVLIBR	PGMDEV1
PAYRLLPR	PRODLIB	PRODVL1
INVBLLPR	PRODLIB	PRODVL2

The RESTORE command must supply the library file names used for back-up. If ASSGN rather than BACKUP statements were used (see the preceding figure), the RESTORE command for the afore-mentioned backup example could be:

```
RESTORE OLDLIB=(IJSYSCL:DEVLIBR.PGMDEV1 -
                IJSYSRL:DEVLIBR.PGMDEV1 -
                IJSYSSL:DEVLIBR.PGMDEV1 ...) -
```

The command requests the librarian to convert the three pre-version 2 libraries to one sublibrary, DEVLIBR.PGMDEV1.

Figure 3-3. Restoring a Backup Tape for Migration

Conversion via Output to SYSPCH

This section gives sample jobs which assume that SYSPCH output is to be directed to tape. Figure 3-4 on page 3-14 shows a sample job for producing SYSPCH output; Figure 3-5 on page 3-15 shows a sample job for restoring libraries from this output.

Cataloging for conversion purposes is a step of the stand-alone installation procedure for VSE/Advanced Functions 2.1.

```
    // JOB PUNCH OUT PRIVATE LIBRARY
(1) // DLBL IJSYSCL,'COSTCTL.SYSTEM.PROJ1'
(1) // EXTENT SYSCLB,,1,0,19,551
(2) // ASSGN  SYSCLB,131
(3) // ASSGN SYSPCH,281
    // EXEC CSERV
(4) PUNCH ALL
    /*
     ... ... ...                DLBL and EXTENT statements if required
(2) // ASSGN  SYSRLB,131
(3) // ASSGN SYSPCH,281
    // EXEC RSERV
(4) PUNCH ALL
    /*
     ... ... ...                DLBL and EXTENT statements if required
(2) // ASSGN  SYSSLB,131
(3) // ASSGN SYSPCH,281
    // EXEC SSERV
(4) PUNCH ALL
    /*
    /&
```

---

(1) // DLBL IJSYSCL,...
// EXTENT SYSLB,...
They define the library that is to be punched out. If the required label information is stored in your system's label-information area, there is no need to supply these statements.

(2) // ASSGN SYSxLB,131
Assigns the system logical unit for the library to the disk drive on which you have mounted the library volume.

(3) // ASSGN SYSPCH,281
The example assumes that a tape drive is available for nine-track tapes with a density of 6250 bpi at the hardware address 281.

(4) PUNCH ALL
Causes all members (phases, object modules, or source books, whichever applies) to be written to tape as card-image records; use this statement for converting an entire library under DOS/VS Release 33 or an earlier DOS/VS release. To convert only specific members of a library (certain procedures, for example), supply instead a PUNCH statement listing the names of those members. Example:

```
PUNCH ASSEMB,COMPCOB,STINVBLL,STPAYRLL,STSALEST,...
```

Figure 3-4. Sample Job for Producing SYSPCH Output on Tape

```
    // JOB     LIBRARY CONVERT-RESTORE - FROM SYSPCH
(1) // OPTION  STDLABEL=ADD
    // DLBL    DEVLIBR,,99/365
    // EXTENT  ,DEVLIB,,,19,7657
    // OPTION  USRLABEL
    // EXEC    LIBR
(2) DEFINE  LIB=DEVLIBR
    /*
    // ASSGN   SYSIPT,281
    // OPTION  CATAL
    // LIBDEF  PHASE,CATALOG=DEVLIBR.PGMDEV1
       INCLUDE
(3) // EXEC    LNKEDT
(4) // EXEC    LIBR,PARM='ACCESS SUBLIB=DEVLIBR.PGMDEV1'
(5) // EXEC    LIBR,PARM='ACCESS SUBLIB=DEVLIBR.PGMDEV1'
    /&
```

---

(1) // OPTION STDLABEL=ADD
Stores the subsequent DLBL and EXTENT statements in the system's information area, accessible from any partition.

(2) DEFINE LIB=DEVLIBR
Formats the library as defined by the preceding DLBL and EXTENT statements.

(3) // EXEC LNKEDT
The linkage editor reads the phases from the tape mounted on drive 281 and catalogs them in the target sublibrary named in the preceding LIBDEF statement.

(4) // EXEC LIBR,PARM='ACCESS SUBLIB=DEVLIBR.PGMDEV1'
The statement invokes the librarian; it passes to the librarian an ACCESS statement that makes sublibrary DEVLIBR.PGMDEV1 accessible for cataloging members of type OBJ (object modules) that were copied onto the input tape from the library IJSYSRL.

(5) // EXEC LIBR,PARM='ACCESS SUBLIB=DEVLIBR.PGMDEV1'
Similar to (4). The ACCESS statement being passed makes the named sublibrary accessible for cataloging source-type members (source books) that were copied onto the input tape from the library IJSYSSL.

Figure 3-5. Sample Job for Restoring Libraries from SYSPCH Output on Tape

Format Conversion, Version 2 to Version 1

The example in Figure 3-6 shows how to convert individual library members from a Version-2 library format to a Version-1 format.

Under control of Version 2 of VSE/Advanced Functions:

```
    // JOB RECONV 1
(1) // DLBL   DEVLIBR,...
(1) // EXTENT ,DEVLIB,...
(2) // ASSGN  SYSPCH,181
    // EXEC   LIBR
(3) ACCESS SUBLIB=DEVLIBR.PGMDEV1
(4) PUNCH  NEWPR*.PHASE
(5) PUNCH  NEWPR*.OBJ,FORMAT=OLD,EOF=NO
(6) PUNCH  NEWPR*.E,FORMAT=OLD
    /*
    /&
```

```
(1) // DLBL  DEVLIBR,...
    // EXTENT  ,DEVLIB,...
    The statements are not needed if the required label information is
    stored in the label-information area.

(2) // ASSGN  SYSPCH,181
    The device at address 181 is assumed to be a tape drive.

(3) ACCESS SUBLIB=DEVLIBR.PGMDEV1
    Makes sublibrary PGMDEV1 of library DEVLIBR accessible.  The members
    to be converted are assumed to reside in this sublibrary.

(4) PUNCH  NEWPR*.PHASE
    Causes all PHASE-type members whose names begin with the characters
    NEWPR to be punched out.

(5) PUNCH  NEWPR*.OBJ,FORMAT=OLD,EOF=NO
    Causes all OBJ-type members whose names begin with the characters
    NEWPR to be punched out.  Specify EOF=NO because object modules and
    source-type members are both cataloged by the Version-1 librarian
    program MAINT.

(6) PUNCH  NEWPR*.E,FORMAT=OLD
    Causes all E-type (edited macro) whose names begin with the
    characters NEWPR to be punched out.
```

Figure 3-6 (Part 1 of 2). Library-Format Conversion – Version 2 to Version 1

Under control of Version 1 of VSE/Advanced Functions:

```
     // JOB RECONV 2
     // DLBL PRODL04,'COSTCTL.SYSTEM.PROJ1'
     // EXTENT SYSCLB,...
        ... ... ...  ]
        ... ... ...  ]  DLBL and EXTENT statements as needed.
        ... ... ...  ]
(1)  // ASSGN  SYSIPT,181
(2)  // LIBDEF CL,TO=PRODL04
(2)  // LIBDEF RL,TO=PRODL05
(2)  // LIBDEF SL,TO=PRODL06
     // OPTION CATAL
      INCLUDE
(3)  // EXEC LNKEDT
(4)  // EXEC MAINT
     /&
```

---

(1) // ASSGN SYSIPT,181
Assigns SYSIPT to the tape drive on which the operator mounted the volume with the library members converted to Version-1 library format.

(2) // LIBDEF xL,TO=PRODL0n
The statements define the target libraries (CL = core image - for phases; RL = relocatable library - for object modules; SL = source statement library - for source books).

(3) // EXEC LNKEDT
The job step catalogs the converted members of type PHASE into the core image library named PRODL04. The number of phases that can be cataloged by one linkage editor run depends on the size of your partition.

(4) // EXEC MAINT
The job step catalogs the converted members of type OBJ into the relocatable library PRODL05 and source-type members into the source statement library named PRODL06.

Figure 3-6 (Part 2 of 2). Library-Format Conversion – Version 2 to Version 1

Conversion of Existing Job Streams

Existing job streams, frequently cataloged as procedures, may be converted (but not tested) under control of your currently installed pre-version 2 system.

Version 2 of VSE/Advanced Function includes a job-stream conversion aid, the Job-Control Scanner program. The program examines job streams that have been copied onto tape and writes the results of its processing to the device assigned to SYSLST. For certain input, the program produces SYSPCH (tape) output in a form suitable as input for later editing under VSE/ICCF. For more information about the scanner program, refer to Appendix E.

For the use of unchanged existing Version-1 job streams under Version 2, the following rules apply.

- DLBL and EXTENT statements defining the addresses and sizes of pre-version 2 libraries have to be changed such that they define your converted (Version-2) libraries.

- LIBDEF statements used in job streams of the past need not be changed if you assemble and catalog a *library-migration table* as SVA eligible. The librarian uses this table for replacing your current library names by the corresponding new-format sublibrary names.

  The librarian accepts pre-version 2 LIBDEF statements and LIBDEF statements in the Version-2 format side by side without any restrictions, except when the ESERV program is used. ESERV requires a LIBDEF statement of the Version-2; else the program accesses the sublibrary IJSYSRS.SYSLIB by default.

  The assembly and cataloging of a library-migration table is further discussed in section "Assembly of and Cataloging a Library-Migration Table" on page 3-19.

- If private libraries were made accessible via job control ASSGN statements (or commands), these statements have to be replaced by LIBDEF statements of the new format. The LIBDEF statements must define the sublibraries that are to be accessed for the processing of a specific job.

  An example showing the necessary changes is given under "Job Streams with ASSGN Statements for Private Libraries" on page 3-20.

- Jobs that depend on a system library as the one to be used by default have to be changed by adding LIBDEF statements of the new format. This is very much the same as shown under "Job Streams with ASSGN Statements for Private Libraries" on page 3-20.

- An existing library service job is accepted if a compatible Version-2 librarian function exists. This is further discussed under "Processing of Existing Library Service Jobs" on page 3-22.

Assembly of and Cataloging a Library-Migration Table

Under Version 2 of VSE/Advanced Functions, you can use existing job streams with LIBDEF statements of the old format. To do this, you have to assemble and catalog a library-migration table. The librarian expects this table to reside in your system's SVA; it uses the table to relate old library names to new sublibrary names.

The macro INLMIGR is available for the assembly of this table. You code this macro to build one entry of the table; that is, one reference from an old library name to a new sublibrary name. Figure 3-7 on page 3-20 shows an example for the assembly and the cataloging of a library-migration table.

Old Library Name	New Sublibrary Name
PGMDEV1	DEVLIBR.PGMDEV1
PGMDEV2	DEVLIBR.PGMDEV1
PGMDEV3	DEVLIBR.PGMDEV1
PRODVL1	PRODLIB.PAYRLLPR
PRODVL2	PRODLIB.INVBLLPR
PRODVL3	PRODLIB.SALESTPR
SYSADMN	PRODLIB.SYSADMIN
IJSYSRS	IJSYSRS.SYSLIB

Assemble and catalog job:

```
     // JOB    CREATE LIBRARY-MIGRATION TABLE
     LIBDEF PHASE,CATALOG=PRODLIB.SYSADMIN
     // OPTION CATAL
     // EXEC   ASSEMBLY
     *         ┌───── Column 10
     *         │
     *         │
               INLMIGR OLD=PGMDEV1,LIB=DEVLIBR,SUBLIB=PGMDEV1
               INLMIGR OLD=PGMDEV2,LIB=DEVLIBR,SUBLIB=PGMDEV1
               INLMIGR OLD=PGMDEV3,LIB=DEVLIBR,SUBLIB=PGMDEV1
               INLMIGR OLD=PRODVL1,LIB=PRODLIB,SUBLIB=PAYRLLPR
               INLMIGR OLD=PRODVL2,LIB=PRODLIB,SUBLIB=INVBLLPR
               INLMIGR OLD=PRODVL3,LIB=PRODLIB,SUBLIB=SALESTPR
               INLMIGR OLD=SYSADMN,LIB=PRODLIB,SUBLIB=SYSADMIN
(1)            INLMIGR OLD=IJSYSRS,LIB=IJSYSRS,SUBLIB=SYSLIB
               END
     /*
     // EXEC LNKEDT
     LIBDEF PHASE,SEARCH=PRODLIB.SYSADMIN
     SET SDL
     INLPLMT,SVA
     /*
     /&
```

(1) INLMIGR OLD=IJSYSRS,LIB=IJSYSRS,SUBLIB=SYSLIB
An entry for a system library is required if an old-format LIBDEF statement refers to that system library explicitly.

Figure 3-7. Example - Assembly and Cataloging of a Library-Migration Table

Job Streams with ASSGN Statements for Private Libraries

Figure 3-8 on page 3-21 shows how existing job streams would have to be changed to make converted libraries accessible. The example given in Figure 3-8 assumes that the three private libraries (IJSYSCL, IJSYSRL, and IJSYSSL) were converted into one sublibrary.

If an existing job stream with ASSGN statements for private libraries includes a link-edit step, replace the ASSGN statements by LIBDEF statements as follows:

```
// LIBDEF OBJ,SEARCH=libname.sublibname [,TEMP|,PERM]
// LIBDEF PHASE,CATALOG=libname.sublibname [,TEMP|,PERM]
```

In this statement, supply the name of the target sublibrary in the format as shown.

A private library unassign request in an existing job stream, such as

```
ASSGN SYSCLB,UA
```

should be replaced by one of the following, whichever applies

```
// LIBDROP PHASE,SEARCH=...[,TEMP|,PERM]
// LIBDROP PHASE,CATALOG=...[,TEMP|,PERM]
```

Statements in Existing Job Streams

```
 ... ... ...
// DLBL IJSYSCL,'COSTCTL.SYSTEM.PROJ1A'
// EXTENT SYSCLB,,1,0,19,551
// ASSGN  SYSCLB,131
// DLBL IJSYSRL,'COSTCTL.SYSTEM.PROJ1B'
// EXTENT SYSRLB,,1,0,570,475
// ASSGN  SYSRLB,131
// DLBL IJSYSSL,'COSTCTL.SYSTEM.PROJ1C'
// EXTENT SYSSLB,,1,0,1045,1083
// ASSGN  SYSSLB,131
 ... ... ...
```

Corresponding Statements Under Version 2

```
     ... ... ...
    // DLBL  DEVLIBR,'COSTCTL.SYSTEM.PROJ1'
    // EXTENT ,DEVLIB,,,19,7657
(1) LIBDEF *,SEARCH=DEVLIBR.PGMDEV1,PERM
```

(1) LIBDEF *,SEARCH=DEVLIBR.PGMDEV1,PERM
Possibly you might want to set up library search chains (for the retrieval of library members) already at this point. You do this by defining the desired search-order chain in the SEARCH operand. For more information about chaining (or concatenation) of sublibraries, see the IBM publication VSE/Advanced Functions System Management Guide.

Figure 3-8. Changing Job Streams with ASSGN Statements for Private Libraries

Processing of Existing Library Service Jobs

Figure 3-9 on page 3-23 gives a list of pre-version 2 library service jobs which the new librarian can process. Any such jobs that may exist at your location need not be changed if:

- They include old format LIBDEF statements with FROM and TO operands specified as required.
- A library-migration table exists (see "Assembly of and Cataloging a Library-Migration Table" on page 3-19).

You may submit existing library service requests of the types listed in Figure 3-9 even if no LIBDEF statements are included. In that case, the librarian prompts you by messages to the system console for the applicable library and sublibrary names.

Existing library service requests not listed in Figure 3-9 have to be recoded.

Figure 3-10 on page 3-24 gives examples for converting existing jobs for library service requests. These examples assume that correct label information for the involved libraries exists.

This ends the discussion of migration to Version 2. The next chapter gives you an overview of the purpose and functions of MSHP.

Type of Job		Remarks
Program	Function	
CORGZ	MERGE with COPYx	The MERGE statement is not checked. A merge to NRS (new system residence) cannot be processed. If a member with the same name exists already, the member in the target library is replaced. x = one of the characters C, P, R, S
xSERV	DSPCH DSPLY PUNCH	x = one of the characters C, P, R, S
DSERV	DSPLY DSPLYS	
MAINT	CATALx	If a member with the same name exists already, the member in the target library is replaced. x = one of the characters P, R, S If x = P, see Note 1, below. If x = S, see Note 2, below.
	DELETx	x = one of the characters C, P, R, S
	RENAMx	x = one of the characters C, P, R, S

Legend: C = Core image library R = Relocatable library
P = Procedure library S = Source statement library

Note 1: The Version-2 librarian copies the /+ (end of data) statement into the target sublibrary.

Note 2: The Version-2 librarian accepts the BKEND statement, but ignores any operands that you may have specified.

Figure 3-9. Library Service Requests Processed Unchanged

Pre-Version 2 Statements	To be Replaced by
Library reorganization (CORGZ) with ALLOC:	

`// EXEC CORGZ`
`   ALLOC CL=nn(n),RL=nnn(n),...`
`   ...`
`   ...  ] Required COPY statements`
`   ...`
`/*`
	`// EXEC LIBR`
	`   DEFINE LIB=libname`
	`   DEFINE SUBLIB=libname.sublibname`
	`    ...`
	`    ...  ] Required librarian`
	`    ...    commands`
	`/*`

Library reorganization (CORGZ) with NEWVOL:	

`// EXEC CORGZ`
`   NEWVOL CL=nn(n),RL=nnn(n),...`
`   ...`
`   ...  ] Required COPY statements`
`   ...`
`/*`
	`// EXEC LIBR`
	`   DEFINE LIB=libname`
	`   DEFINE SUBLIB=libname.sublibname`
	`    ...`
	`    ...  ] Required librarian`
	`    ...    commands`
	`/*`

Library reorganization (CORGZ) with COPY ALL (assuming that two pre-version 2 libraries are involved):	

`// EXEC CORGZ`
`   COPY ALL`
`/*`
	`// EXEC LIBR`
	`   CONNECT -`
	`     S=liba.sublibb:libx.subliby`
	`   COPY *.*`
	`/*`

Library reorganization (CORGZ) with MERGE to NRS:	

`// EXEC CORGZ`
`   MERGE RES,NRS`
`/*`
	`// EXEC LIBR`
	`   CONNECT -`
	`     S=IJSYSRS.SYSLIB:lib.sublib`
	`   COPY *.*`
	`/*`

Library maintenance (MAINT) for source library update (assuming that one library is involved):	

`// EXEC MAINT`
`   UPDATE x.name1,x.name2,v.m,nn`
`   ...`
`   ...  ] Required update con-`
`   ...    trol statements`
`/*`
	`// EXEC LIBR`
	`   ACCESS SUBLIB=libname.sublibname`
	`   UPDATE name1.x SA=name2.x SEQ=nn`
	`   ...`
	`   ...  ] Update subcommands (like`
	`   ...    the update control state-`
	`/*        ments in the past)`

Figure 3-10. Examples for Converting Existing Library Service Jobs

Chapter 4. The Maintain System History Program (MSHP)

MSHP, the IBM installation and service tool for VSE systems, is a part of VSE/Advanced Functions. Any installation or service job using MSHP runs in a partition of 640K bytes or larger; this includes MSHP's partition GETVIS requirement of up to 92K bytes.

MSHP uses various other system programs to perform its installation and service functions. These programs (for example, the linkage editor and the librarian) run in the same partition as MSHP. MSHP invokes them and passes data to them, and they return control to MSHP when their function is complete.

MSHP records installation and service activities done under its control in a file on disk. This file, the system history file, is used by MSHP for maintaining system integrity. It is therefore essential that the file reflects the system's current change level at all times.

During installation and service of licensed programs, certain functions of MSHP use an auxiliary history file, normally as a work file.

The System History File

The file is maintained under the file name IJSYSHF. Unless you specify otherwise, MSHP uses the logical unit SYSREC, the volume that contains the system recorder file, to access the history file. However, you can place the history file on a volume other than SYSREC, and use any programmer logical unit to refer to the file.

The file label should be permanently defined (// OPTION STDLABEL). To place the file to be on the SYSREC volume, supply DLBL and EXTENT statements as follows:

```
// DLBL IJSYSHF,'A5666301.SYSTEM.HISTORY.FILE',99/365
// EXTENT SYSREC,vol-id,1,0,number1,number2
```

```
where vol-id = The six-character volume serial number of the con-
               taining disk volume
```

You may also define the file by using the MSHP DEFINE statement. If you use this method and request several MSHP functions in one run, your DEFINE statement must follow the *first* MSHP function request. This definition is valid only until the end of this MSHP run.

The key elements of the history file are briefly discussed below:

- The product-identification (or feature) records

  A record of this type contains the six-digit product code. For VSE/Advanced Functions 2.1, for example, this code is 301H07, where:

  301 is a number unique for VSE/Advanced Functions.
  H07 is the component level number (Version 2, Release 1).

  When you install other products later on, the code enables MSHP to verify that a required program is installed already.

- The component records

  A record of this type contains the component and release identification. The record is the "anchor" for all service related data such as installed PTFs and APAR fixes or the file identifier of the target sublibraries for service changes.

  If a component is removed from the system under MSHP control, MSHP invalidates all service related data stored for the component in the system history file.

The Auxiliary History File

MSHP needs this file for certain functions during the installation of a licensed program or of service changes.

The file, which MSHP uses primarily as a work file, is maintained on disk under the file name IJSYS02. Normally, MSHP creates the file on the volume assigned to the logical unit SYS002, but you can use any other programmer logical unit if this is desirable. If you use a different programmer logical unit, this logical unit must be assigned before you submit the EXEC statement invoking MSHP.

MSHP Control Statements

MSHP has two types of control statements: function control and detail control.

A *function control* statement defines the desired MSHP function.

A *detail control* statement provides descriptive data about the requested function. Detail control statements, if required for a function, must always follow immediately the applicable function control statement.

Using MSHP

The remaining chapters describing the various installation procedures show how you use MSHP.

Chapter 5. Installing VSE/Advanced Functions

On completion of the applicable installation procedure (stand-alone or on-line), VSE/Advanced Functions as shipped by IBM is installed together with an up-to-date history file.

Besides describing the procedure, this chapter expands on items that you should do or consider before you start installing VSE/Advanced Functions. It includes a section on the proper response to MNOTEs, assembler generated error messages. The chapter does not discuss required migration activities; they are discussed in Chapter 3.

Preparing the Installation Process

In preparing the process, it may be helpful to read the selected installation procedure from beginning to end and to set up required jobs in advance. By familiarizing yourself with the procedure, you can decide beforehand which steps you have to do and which you may safely omit, based on your system-planning decisions.

As mentioned in Chapter 1, you can initialize required disk volumes in advance. To be on the safe side, you can also produce a backup tape of your system-residence file and your system's history file for fallback purposes. Use the support available with your currently installed system as far as this is possible.

Initialize Disk Volumes: The Device Support Facilities program is available for this purpose. For information on how to use this program, see the IBM publication *Device Support Facilities.*

If migration to this release of VSE/Advanced Functions does not involve newly supported disk devices, use the program's on-line version.

If your migration involves newly supported disk devices, use the stand-alone version of the Device Support Facilities program as shipped with the distribution tape. To use this version of Device Support Facilities, perform an IPL from the tape.

When initializing CKD disk volumes on a VSE system, bear in mind that the VTOC (Volume Table of Contents) must be contained within one cylinder on these devices. Avoid having too many small files on one volume.

System Backup: Use your Version 1 system's BACKUP system utility for producing a backup tape. The section "Restore-Conversion Method" on page 3-9 gives a sample job.

MNOTEs During Assemblies

During an assembly (of your supervisor, for example, if this is necessary), the assembler may issue MNOTEs (macro error notes).

You may choose to ignore some by accepting the default values given in the MNOTEs. For others, you may have to modify your specification(s) and reassemble the supervisor. A general procedure for resolving MNOTEs follows:

1. Go to the DIAGNOSTICS section at the end of the assembly listing; the section includes references to the MNOTEs generated during assembly. An MNOTE reference is in the form:

   ```
   statement-no. IPK216  MNOTE GENERATED
   ```

2. Look up the statement in the listing and examine the MNOTE. A severity code precedes the message portion – the higher the code, the more severe the error.

3. Find out the reason for the MNOTE. The MNOTE indicates the affected operand and usually gives an indication of the detected discrepancy. Some errors to look for are:

   - Misspelled names or erroneous numbers.
   - Specifications outside the valid limits.
   - Operands or specifications that are incompatible with other operands or specifications in the same or in another macro.

4. Make necessary changes and reassemble the source code.

Procedures for Installing VSE/Advanced Functions

VSE/Advanced Functions 2.1 is to be installed stand-alone if you migrate from a Version-1 release of VSE/Advanced Functions or a release of DOS/VS or even DOS. This means that you begin with an initial program load from the tape drive on which you have mounted the distribution tape. No application can run on your computer system until the installation is complete. The procedure for stand-alone installation is described under "The Stand-Alone Installation Procedure" on page 5-3.

Once you have migrated to Version 2 of VSE/Advanced Functions and this system is operational, you can do an on-line installation (of a refresh release, for example). In that case, you install the new system under control of your currently operational system. The procedure for this kind of installation is described under "The On-line Installation Procedure" on page 5-29.

The installation procedures include a number of sample jobs. How to read sample jobs is discussed in section "Reading Sample Jobs and Prompting Examples" on page 1-6.

The Stand-Alone Installation Procedure

The procedure assumes that the required disk volumes have been initialized and are on-line. How to initialize a disk volume is described in the IBM publication *Device Support Facilities.*

Various steps of the procedure refer to illustrations that show examples of program prompts and responses. In these illustrations:

- Prompts and message displays by the system are shown as character strings in uppercase.
- Responses and command input are shown as character strings in lower case; in addition, they are indicated by an arrow (= = = >).
- Pressing END/ENTER is indicated by a square bullet (■).

Step 1. Restore the System Residence File

This step restores the PRODUCTION library of VSE/Advanced Functions from the distribution tape to disk. Following this step, you can start up the new system and perform the remaining steps. Restoring the GENERATION library from the distribution tape is an extra step (or job) under control of code contained in the PRODUCTION library. This is described under "Step 6. Restore the GENERATION Library" on page 5-15.

Proceed as follows:

1. Mount the IBM-supplied distribution tape on a tape drive.

2. Perform two successive IPLs from this tape drive.

   How to do an IPL is described in the *Operating Procedures* manual for your processing unit.

   The first IPL causes the stand-alone Device Support Facilities (DSF) program to be loaded into processor storage. By loading DSF, the system positions the distribution tape for loading one of the available stand-alone VSE programs. Loading of DSF is complete when the system enters the wait state.

   Perform the second IPL at this point.

3. When the system enters the wait state after your second IPL, press END/ENTER.

This causes the system to prompt you for device and control information as shown by the example in Figure 5-1 on page 5-4. Following this prompting sequence, the system restores the PRODUCTION library from the distribution tape to disk.

```
             ***** STAND ALONE PROGRAMS LOADED *****
             IF YOU WANT A LISTING, SPECIFY CUU OF PRINTER;
             ELSE, OR IF PRINTER IS NOT OPERATIONAL, PRESS END/ENTER.
(1) ===> 00e ■
             SPECIFY DEVICE TYPE OF PRINTER XXXXYY
(2) ===> PRT1 ■
             SPECIFY DATE  MM/DD/YY
    ===> 05/31/84 ■
             SELECT ONE OF THE FOLLOWING PROGRAMS OR TYPE END
             FASTCOPY, RESTORE, INITEM
    ===> restore ■
             SPECIFY ADDRESS OF INPUT DEVICE  CUU
    ===> 280 ■
             SPECIFY TYPE OF INPUT DEVICE  XXXXYY
(2) ===> 3420t9 ■
             SPECIFY ADDRESS OF SYSRES DISK  CUU  OR END/ENTER
    ===> 160 ■
             SPECIFY TYPE OF DISK  XXXXYY
(2) ===> 3330 ■
             L302A  ENTER YES TO RESTORE SYSRES FILE IJSYSRS OR NO TO SKIP TO
                    NEXT SYSRES
    ===> yes ■
             L315I  ORIGINAL FILE ID= A5666301.PRODUCTION.LIBRARY
             L316A  ENTER YES TO KEEP OR NO TO RESPECIFY THE SYSRES FILE ID
    ===> yes ■
(3)          L309I  ORIGINAL ALLOCATION=   1367 TRACKS =  71 CYLINDERS 18 TRACKS
```

---

(1) ===> 00e ■
If the printer has a universal character set buffer (UCB) and this buffer contains a wrong buffer image, list output may be hard to read or even unusable.

(2) ===> PRT1 ■ | 3420t9 ■ | 3330 ■
Define input or output device types. Specify the type codes of the devices you use. For a list of valid device-type codes, refer to the description of the IPL ADD command in the IBM publication _VSE/Advanced Functions System Control Statements_.

(3) L309I ORIGINAL ALLOCATION= 1367 TRACKS = 71 CYLINDERS 18 TRACKS
The system displays the amount of disk space needed to restore the system library. The displayed size considers a free space of 20 %.

Figure 5-1 (Part 1 of 2). Stand-Alone Restore Prompting Example

Stand-alone restore prompting example (continued)

```
(4)      L310A  ENTER YES TO KEEP OR NO TO RESPECIFY THE ALLOCATION
    ===> yes ■
(5)      L329A  ENTER YES TO RESTORE ALL SUBLIBRARIES OR NO FOR SELECTIVE
                RESTORE
    ===> yes ■
         L338I  SUMMARY OF RESTORE PARAMETERS:
         L318I  FILE NAME= IJSYSRS
         L319I  FILE ID= A5666301.PRODUCTION.LIBRARY
         L321I  ALLOCATION=  1367 TRACKS
         L344I  START= CYLINDER 0 TRACK 1 - END= CYLINDER  71 TRACK 18
         L327I  RESTORE ALL SUBLIBRARIES
         L322A  ENTER YES IF THE SPECIFICATION IS CORRECT OR NO TO RESPECIFY
    ===> yes ■
         L300I  FORMATTING OF LIBRARY IJSYSRS IN PROGRESS
         L306I  RESTORE OF LIBRARY IJSYSRS IN PROGRESS
         L325I  RESTORE OF SUBLIBRARY IJSYSRS.SYSLIB IN PROGRESS
(6)      L325I  RESTORE OF SUBLIBRARY IJSYSRS.PR$DS2 IN PROGRESS
(6)      L325I  RESTORE OF SUBLIBRARY IJSYSRS.PR$092 IN PROGRESS
(6)      L325I  RESTORE OF SUBLIBRARY IJSYSRS.PR$260 IN PROGRESS
         L326I  RESTORE COMPLETE FOR LIBRARY IJSYSRS
         *** END OF STAND ALONE PROCESSING ***
```

---

(4) L310A ENTER YES TO KEEP OR NO TO RESPECIFY THE ALLOCATION
The message prompts you to either accept or reject the displayed value. To accept, ensure the displayed size is large enough to copy, into this library, additional members that may have to be included. This example assumes that the size as displayed can be reduced.

(5) L329A ENTER YES TO RESTORE ALL SUBLIBRARIES OR NO FOR SELECTIVE RESTORE
You should respond to the message as shown. A response of "no" to this message is meaningful only if you restore from a backup copy in order to rebuild one of the PRODUCTION library's sublibraries.

(6) L325I RESTORE OF SUBLIBRARY IJSYSRS.PR$xxx IN PROGRESS
PR$DS2 = Device Support Facilities
PR$092 = VSE/Online Test Executive Program
PR$260 = Environmental Recording Editing and Printing Program

Figure 5-1 (Part 2 of 2). Stand-Alone Restore Prompting Example

Step 2. Start Up the New System

Use the ASI method; the restored system includes applicable start-up procedures. For the statements contained in these procedures, see the *Program Directory* shipped with the distribution tape.

Proceed as follows:

1. Initiate program load as described in the *Operating Procedures* manual for your processing unit.

2. Enter the load unit address.

The system now prompts you for required control information as shown in Figure 5-2 on page 5-7; it displays the control information that is supplied by means of the selected ASI procedures. Follow the example and deviate as required.

You may want to override some of the specifications in the ASI procedure to meet your specific requirements. You can do this by using the ASI stop facility. The example shows how to use this facility.

```
     ===> i 160 ■
(1)       WAIT
(2)       [I] ■
(3)       0I04I  IPLDEV=X'160',VOLSER=vol-id,CPUID=110502574331
          0I03D  ENTER SUPERVISOR PARAMETERS OR ASI PARAMETERS
(4)  ===> ipl=iplsupe,jcl=$$jcl1,stop=add ■
          0J01I  IPL=IPLSUPE ,JCL=$$JCL1  ,SUPVR=$$A$SUPE,P
          0I30I  DATE=05/03/83,CLOCK=11/29/53,ZONE=EAST/00/00
                 THE DATE VALUE FORMAT IS MM/DD/YY
```

---

(1) WAIT
The system enters the wait state to get an interrupt from the console (IPL) device.

(2) [I] ■
Hit the Interrupt key [I] first and then the END/ENTER key.
If you restore under VM/SP, submit a CP EXT command instead.

(3) 0I04I IPLDEV=X'160',VOLSER=vol-id,CPUID=110502574331
For vol-id, the system displays the six-character identifier of your system-residence volume.

(4) ===> ipl=iplsupe,jcl=$$jcl1,stop=add ■
In response to message 0I03D, specify the set of IPL and JCL procedures that load the supervisor you want to use. The sets of the available IPL and JCL procedures are:

```
ipl=iplsupe,jcl=$$jcl1 - To get the MODE=E supervisor.
ipl=iplsup3,jcl=$$jcl1 - To get the MODE=370 supervisor.
ipl=iplsupv,jcl=$$jcl1 - To get the MODE=VM supervisor.
```

Specifying stop=add causes the system to stop processing when it encounters the first of the specified commands.
Use the IPL command that best suits your requirements; for example, omit 'add' in the list if there is no need for any reconfiguration.

Figure 5-2 (Part 1 of 5). Example of ASI Start-Up of the Restored System – Stand-Alone

Example of ASI start-up of the restored distribution system (continued)

```
          ADD 01E,3277
          0J05D  ASI STOP. ENTER COMMANDS, HIT END/ENTER TO CONTINUE
(5) ===> add 080,3277 ■
(5) ===> add 230:233,3380 ■
    ===> ■
          ADD 00C,2540R
          ADD 00D,2540P
           ... ... ...
          ADD 280:283,3420T9
          DEF SYSREC=160
          0J10I IPL RESTART POINT BYPASSED
          DEF SYSCAT=UA
          DLA UNIT=160,NCYL=1,CYL=180,DSF=Y,NAME=VSE/AF
          0I52I LABEL AREA ON 160:        LOW        HIGH
                          CC HH:      180   0     180  18
          DPD UNIT=160,CYL=181,DSF=N
          0I52I PDS EXT 01 ON 160:        LOW        HIGH
                          CC HH:      181   0     194  18
                    PAGE NUMBER:            0       1535
                    VIO START PAGE NUMBER:          1504
          SVA
          0J24I DASD SHARING SUPPORT RESET
(6)       0J39I ACTUAL BUFSIZE IS 550
          0I26I $$BUCB3  AND -------- LOADED        CUU=00E
          0I20I IPL COMPLETE FOR VSE/AF 5666-301 V2 R1.0 ECLEV=0000
                SUPVR USERID=VSE.4300.SUPE
```

---

(5) ===> add 080,3277 ■
    ===> add 230:233,3380 ■

After having displayed ADD 01E,3277, a command included in the IPL procedure, the system gives you a chance to enter one or more additional ADD commands. The second command (add 230:233,3380) defines four IBM 3380s as being attached to the system with the hardware addresses 230 through 233.

By entering a null line (hitting END/ENTER) you cause processing to be continued with the displayed command, ADD 01E,3277 in this case.

(6) 0J39I ACTUAL BUFSIZE IS 550

The displayed size (550) applies to a processor storage size of 4M bytes.

Figure 5-2 (Part 2 of 5). Example of ASI Start-Up of the Restored System – Stand-Alone

Example of ASI start-up of the restored distribution system (continued)

```
(7)      BG 000 LOG
         BG 000 ASSGN SYSLST,PRINTER
         BG 000 1T20I  SYSLST HAS BEEN ASSIGNED TO X'00E'
         BG 000 ASSGN SYSPCH,PUNCH
         BG 000 1T20I  SYSPCH HAS BEEN ASSIGNED TO X'00D'
         BG 000 ASSGN SYSLNK,SYSRES
         BG 000 1T20I  SYSLNK HAS BEEN ASSIGNED TO X'160'
         BG 000 ASSGN SYS001,SYSRES
         BG 000 1T20I  SYS001 HAS BEEN ASSIGNED TO X'160'
         BG 000 ASSGN SYS002,SYSRES
            ... ... ...
         BG 000 1T20I  SYS004 HAS BEEN ASSIGNED TO X'160'
         BG 000 ASSGN SYS025,SYSRES
         BG 000 1T20I  SYS025 HAS BEEN ASSIGNED TO X'160'
         BG 000 ASSGN SYS026,SYSRES
         BG 000 1T20I  SYS026 HAS BEEN ASSIGNED TO X'160'
         BG 000 * AFTER END OF PROCEDURE:
         BG 000 * SET RF=CREATE (IF RECORDER FILE HAS TO BE CREATED)
         BG 000 * SET HC=CREATE (IF HARD COPY FILE HAS TO BE CREATED)
         BG 000 * EXEC PROC=LABEL... (IF LABELS HAVE TO BE ESTABLISHED)
         BG 000 EOP $0JCL1
         BG 000 1C10D  PLEASE ASSIGN SYSRDR.
         BG-000

(8) ===> 0 assgn sysin,c ■
         BG-000
```

(7) BG 000 LOG
The first statement of the JCL procedure processed by the system.

(8) ===> 0 assgn sysin,c ■
At this point, you may submit additional ASSGN commands to override the assignments made as a result of processing the JCL procedure.

Figure 5-2 (Part 3 of 5). Example of ASI Start-Up of the Restored System – Stand-Alone

Example of ASI start-up of the restored distribution system (continued)

```
(9) ===> 0 exec proc=labels30 ■
         BG 000 *  STANDARD LABELS FOR 3380 FOR ORIGINAL ALLOCATIONS
         BG 000 *  USE COL 73 -79 ONLY AS IDENTIFIER IN PROC OV OPERATION
         BG 000 // OPTION STDLABEL
         BG 000 // DLBL IJSYS04,'VSE/AF.WORK-FILE.4',0,SD
         BG 000 // EXTENT SYS004,,1,0,6422,152    8 CYLS, 338 - 345, ANY DRIVE
          ... ... ...
         BG 000 // OPTION USRLABEL
         BG 000 // OPTION PARSTD
         BG 000 EOP LABELS30
```

---

(9) ===> 0 exec proc=labels30 ■

Executes procedure LABELS30 to store standard label information. The distribution tape includes procedures for this purpose as follows (for a listing of their contents, refer to the Program Directory):

LABELFBA	For FBA disk devices
LABELS14	For IBM 2314/2319 disk devices
LABELS30	For IBM 3330/3333 disk devices
LABELS40	For IBM 3340/3344 disk devices (with 70M data modules)
LABELS50	For IBM 3350 disk devices
LABELS75	For IBM 3375 disk devices
LABELS80	For IBM 3380

To store the required label information, either specify the procedure that best meets your location's needs or run a job of your own. Use one of the supplied procedures if all or most of the included library and file definitions are acceptable. You can, later on, change this label information by deleting some and adding other. If you run your own job, ensure that it includes DLBL and EXTENT statements for your system's history file as shown below:

```
// DLBL IJSYSHF,'A5666301.SYSTEM.HISTORY.FILE',99/365
// EXTENT SYSREC,vol-id,1,0,number1,number2
```

In the // EXTENT statement, you may use a programmer logical unit instead of SYSREC if the history file resides on a volume other than the one with the system recorder file.
Refer to "The System and System-Work Files" on page 2-10 for IBM recommended extent sizes. For how to store label information, see the IBM publication VSE/Advanced Functions System Management Guide.

Figure 5-2 (Part 4 of 5). Example of ASI Start-Up of the Restored System – Stand-Alone

Example of ASI start-up of the restored distribution system (continued)

```
         BG-000
(10)===> 0 // job a ■
         BG 000 // JOB A
         DATE 07/06/84,CLOCK 11/33/14
         BG 000 1I93I  RECORDER FILE IS   1% FULL
         BG-000
    ===> 0 /& ■
         BG 000 EOJ A
         DATE 07/06/83,CLOCK 11/33/18,DURATION 00/00/03
         BG-000
```

```
(10)===> 0 // job a ■
    This is a dummy job.  The job formats the recorder and the hard-copy
    file.
```

Figure 5-2 (Part 5 of 5). Example of ASI Start-Up of the Restored System – Stand-Alone

Step 3. Catalog Your Own Print Control Buffer Phases

This step applies if the print trains (or belts) used on your location's printers do not match the default FCB and UCB images that are loaded automatically during IPL. You may catalog your own FCB (forms control buffer) or UCB (universal character set buffer) image phases, if there is a need.

For a list of default FCB and UCB image phases, refer to the IBM publication *VSE/Advanced Functions System Control Statements*; this list includes the names of the UCB-image object modules which IBM has included in the PRODUCTION library for possible link-editing. The publication describes, in addition, how to create FCB or UCB image phases and how to catalog them.

Your control statements for cataloging buffer-image phases (assuming that an IBM supplied UCB-image object module can be used) should be:

```
// JOB    CATALOG BUFFER IMAGE
// OPTION CATAL
LIBDEF *,SEARCH=IJSYSRS.SYSLIB,CATALOG=IJSYSRS.SYSLIB
  PHASE $$BUCBxx,*
  INCLUDE IJBxxxxx
   ...
   ...       PHASE and INCLUDE statements for additional
   ...       buffer-image phases.
   ...
// EXEC LNKEDT,PARM='MSHP'
/*
/&
```

On completion of this job step, you can load any of the newly cataloged buffer-image phases by entering an LFCB or LUCB command. Use a format as shown:

```
LFCB cuu,phasename
LUCB cuu,phasename,NOCHK
```

Step 4. Restore the System History File

Do this by running a job as shown in Figure 5-3. However, if you want to restore the file to a unit other than SYSREC, make sure a history-file extent has been defined already by permanently stored DLBL and EXTENT statements (OPTION STDLABEL). Else change the job by supplying DLBL and EXTENT statements for the file behind such an OPTION statement (see also reference (9) in Figure 5-2 earlier in this procedure).

On completion of this step, the functions of MSHP are available to you.

```
    // JOB RESTORE SYSTEM HISTORY
(1) // ASSGN SYS006,280
(2) // MTC REW,SYS006
(2) // MTC FSF,SYS006,2
    EXEC MSHP
    RESTORE HIST SYS
    /*
    /&
```

```
(1) // ASSGN SYS006,280
    The statement assumes that the distribution tape is still mounted on
    the tape drive used for restoring the PRODUCTION library (see also
    Figure 5-1 on page 5-4).

(2) // MTC REW,SYS006
    // MTC FSF,SYS006,2
    The MTC statements position the distribution tape to the beginning of
    the system history file.
```

Figure 5-3. Restoring the System History File – Stand-Alone

Step 5. Personalize the System History File

To personalize the restored system's history file (for ease of system identification on listings) use control statements as shown in Figure 5-4.

```
    // JOB PERSONAL
    // EXEC MSHP
(1) PERS 'customer name' -
(2)      ADDR='location' -
(3)      PHONE='extension' -
(4)      PROGR='programmer name' -
(5)      ENV='environment'
    /*
    /&
```

---

(1) PERS 'customer name'
For 'customer name', specify a company identification of no more than 20 characters.

(2) ADDR='location'
For 'location', specify your address; your specification may not be longer than 45 characters.

(3) PHONE='extension'
For 'extension', specify your telephone number. Your specification may not be longer than 17 characters.

(4) PROGR='programmer name'
For 'programmer name', specify the responsible programmer's name; your specification may not be longer than 24 characters.

(5) ENV='environment'
For 'environment', specify information related to your system. For example: the component level code(s) describing your software, the type code of your computer system's processing unit and system residence device. Your specification may not be longer than 62 characters.

Figure 5-4. Personalizing the History File — Stand-Alone

Step 6. Restore the GENERATION Library

Skip this step if you install only the PRODUCTION library.

To restore the GENERATION library, which you must do if you assemble a supervisor of your own, for example, submit a job similar to the one shown in Figure 5-5.

```
    // JOB  INSTALL GENERATION LIBRARY
(1) // DLBL GENLIB,'A5666301.GENERATION.LIBRARY'
(2) // EXTENT  ,WRKFL1,,,1,1700
(3) // ASSGN SYS006,280
(4) // MTC REW,SYS006
    // EXEC MSHP
(5) INSTALL SYSRES GENERATION INTO=GENLIB
    /*
    /&
```

---

(1) // DLBL GENLIB,'A5666301.GENERATION.LIBRARY'
The IBM set default file name (GENLIB) and file identifier are used. You may replace them by names of your own choosing, but to simplify future service applications, you should retain the default names. There is no need for you to define the library to the system's librarian. MSHP does this for you automatically.

(2) // EXTENT ,WRKFL1,,,1,1700
Restoring the library to a work volume is of advantage. Later, when the library is no longer required for the installation of VSE/Advanced Functions, you can create a backup tape of this library and take it offline. This makes the WRKFL1 volume available for reuse.

If disk space is not a problem, you may restore the GENERATION library to a permanently allocated area on disk and keep the library on-line.

This set of statements defines space for a library named GENLIB. There is no need for you to supply the statements; MSHP generates them automatically if you omit them.

(3) // ASSGN SYS006,280
Assign SYS006 to the tape drive on which you have mounted the distribution tape; in this example it is the drive at the address 280.

(4) // MTC REW,SYS006
Rewind the distribition tape.

(5) INSTALL SYSRES GENERATION INTO=GENLIB
This MSHP statement causes the GENERATION sublibrary to be read from the distribution tape and to be stored on the disk volume WRKFL1.

Figure 5-5. Restoring the GENERATION Library – Stand-Alone

Step 7. Tailor Your Own Supervisor

This step does not apply if you install only the PRODUCTION library.

The job for this step, which assembles a supervisor and catalogs it into the sublibrary IJSYSRS.SYSLIB, is to be submitted from the SYSRDR/SYSIPT device. A sample job is shown in Figure 5-6 on page 5-17; it causes the following to be recorded in the system's history file:

- The tailoring of a supervisor named $$A$SUP1.
- The macros (such as FOPT, IOTAB) that were included by the assembler.
- All data between the statements EXECUTE ... and /$.

Make it a point to submit the job only after having verified your source code. You may have to repeat this step if the assembler detects an error. An error in the source code causes the assembler to write an appropriate MNOTE reference at the end of the assembler output listing. The section "MNOTEs During Assemblies" on page 5-2 discusses this topic in more detail.

```
    // JOB $$A$SUP1
(1) // ASSGN SYS001,cuu
(1) // ASSGN SYS002,cuu
(1) // ASSGN SYS003,cuu
(1) // ASSGN SYS004,cuu
(1) // ASSGN SYSLNK,cuu
    // ASSGN SYSPCH,IGN
    // OPTION CATAL
    // EXEC MSHP
(2) TAILOR 5666-301-06-H07 PHASE=$$A$SUP1 KEEPDATA
(3) RESOLVES 'SUPERVISOR 1 GENERATION'
    EXECUTE (ASSEMBLY LNKEDT) XREF
    *        10 — 16 ────────── Column ──────────────────────── 72
    *         |      |                                            |
    *         |      |                                            |
              TITLE 'VSE SUPERVISOR SOURCE CODE FOR $$A$SUP1'
              SUPVR ID=1,                                         C
                    MICR=1419,                                    C
                     ... ... ...
              FOPT  DASDSHR=YES,                                  C
                    TTIME=BG,                                     C
                     ... ... ...
              IOTAB IODEV=254,                                    C
                    NPGR=600
              END
    /$
    /*
    /&
```

---

(1) // ASSGN SYSnnn,cuu
Assignments for work files (on disk) as follows:

Logical Unit	For Use by	File Name in EXTENT Statement
SYS001	Assembler and linkage editor	IJSYS01
SYS002	Assembler	IJSYS02
SYS003	Assembler	IJSYS03
SYS004	MSHP	IJSYS04
SYSLNK	Linkage editor	IJSYSLN

(2) TAILOR 5666-301-06-H07 PHASE=$$A$SUP1 KEEPDATA
KEEPDATA, an optional specification, causes the source code to be stored in the system's history file. MSHP uses that code if re-tailoring is to be done later on.

(3) RESOLVES 'SUPERVISOR 1 GENERATION'
An example of an optional comment statement.

Figure 5-6. Assembly of a Supervisor – Stand-Alone

Step 8. Assemble and Catalog IOCS Modules

Perform this step only if the release you get:

1. Includes the support of a new unit-record device (a printer, for example).

2. Your computer system's configuration includes this new device.

3. Your VSE system does not include a compatible I/O module (for a PRT1 printer, for example).

You may assemble and catalog IOCS modules at any time after completion of the installation. Ignore this step if there is no need for the assembly and cataloging of IOCS modules under control of MSHP.

To catalog an IOCS module under control of MSHP, you use the program's TAILOR function. Run a job similar to the one shown in Figure 5-7 on page 5-19. Submit the job from the SYSRDR/SYSIPT device.

Make it a point to submit the job only after having verified your source code. You may have to repeat this step if the assembler detects an error. An error in the source code causes the assembler to write an appropriate MNOTE reference at the end of the assembler output listing. The section "MNOTEs During Assemblies" on page 5-2 discusses this topic in more detail.

```
    // JOB TAILIOCS
    // OPTION DECK,LIST,LOG,NOXREF
(1) // ASSGN SYS001,cuu,VOL=vol-id
(1) // ASSGN SYS002,cuu,VOL=vol-id
(1) // ASSGN SYS003,cuu,VOL=vol-id
(1) // ASSGN SYS004,cuu,VOL=vol-id
(1) // ASSGN SYSLNK,cuu
    // ASSGN SYSPCH,IGN
    // EXEC  MSHP
(2) TAILOR 5666-301-02 MOD=IJCFAOI7 KEEPDATA
    EXECUTE (ASSEMBLY LIBR) NOX
    *       10 — 16 ———————— Column ———————————————— 72
    *        |    |                                   |
    *        |    |                                   |
             PRINT NOGEN
             CDMOD CTLCHR=ASA,                        C
                   IOAREA2=YES,                       C
                   DEVICE=3505,                       C
                   SEPASMB=YES
             END
    /$
(2) TAILOR 5666-301-02 MOD=IJFFZZZZ KEEPDATA
    EXECUTE (ASSEMBLY LIBR) NOX
             PRINT NOGEN
             CDMOD RECFORM=UNDEF,                     C
                   DEVICE=3881,                       C
                   SEPASMB=YES
             END
    /$
    /*
    /&
```

(1) // ASSGN SYS00n,cuu,VOL=vol-id
These are the same work-file assignments as for the preceding step (tailoring your own supervisor).

(2) TAILOR 5666-301-02 MOD=IJxxxxxx KEEPDATA
Specifying KEEPDATA, which is optional, causes the source code (of IJCFAOI7 and IJFFZZZZ, respectively) to be stored in the system's history file. MSHP uses that code if retailoring is to be done later on.

Figure 5-7. Assembly of IOCS Modules – Stand-Alone

Step 9. Build the Dump Analysis-Routine File

Information/Analysis needs this file for analysing a stand-alone dump. To build the file, an extent of one track on a CKD disk or one block on an FBA disk is to be defined. Submit a job similar to the one shown in Figure 5-8.

```
(1)  // ASSGN   SYS004,SYSIPT
     // ASSGN   SYS005,SYSRES
     // OPTION  STDLABEL=ADD
     // DLBL    UOUT,'INFOANA.ROUTINE',,SD
(2)  // EXTENT  SYS005,,1,0,number1,1
     // OPTION  USRLABEL
     // EXEC    OBJMAINT
     ./ CARD    DLM=$$
     ./ COPY
     ANEXIT     IJBXDBUG
     ANEXIT     IJBXSDA
     $$
     /*
     /&
```

(1) `// ASSGN   SYS004,SYSIPT`
OBJMAINT uses SYS004 as logical unit for input. The statement makes the system-input device available as input.

(2) `// EXTENT  SYS005,,1,0,number1,1`
For number1, code the start track (or block) reserved for the extent by IBM in the label for file BLNXTRN. To get this number, refer to the IBM supplied label-information procedures. These procedures are listed in the Program Directory.

Figure 5-8. Building the SA-Dump Analysis Routine File — Stand-Alone

Step 10. Migrate your Current (Pre-Version 2) Libraries

Not until these libraries have been migrated is your computer system ready for productive operation under Version 2 of VSE/Advanced Functions. This step is, in fact, a multiple-step process. For more information about the migration of these libraries and for sample jobs, see Chapter 3.

Step 11. Create a Backup Copy

At this point of the procedure, you may want to create a backup tape of your new system residence file and, optionally, of the migrated private libraries. To create a backup copy, proceed as follows:

1. Mount the tape to be used on an available tape drive.

2. Run a job similar to the one shown in Figure 5-9.

You can use the created backup tape for stand-alone restore of the system residence file and an on-line restore of private libraries as required. For more information about restoring libraries, refer to the IBM publication *VSE/Advanced Functions System Management Guide.*

```
    // JOB BACKUP SYSTEM
    // EXEC LIBR
(1)  BACKUP LIB=IJSYSRS TAPE=180 RESTORE=STANDALONE -
            INCLUDE=HISTORY
(2)  BACKUP LIB=SLECNTL TAPE=180 -
            ID='SALES-CNTL-APPL'
     ...   ...   ...
    /*
    /&
```

```
(1) BACKUP LIB=IJSYSRS TAPE=180 RESTORE=STANDALONE -
           INCLUDE=HISTORY
    This causes the system residence file, including the system history
    file, to be dumped onto the tape on the drive at the hardware address
    180 - the available tape drive in this example.

    By specifying 'RESTORE=STANDALONE,' you cause an IPL and a PRODUCTION
    library restore routine to be written onto the backup tape.

(2) BACKUP LIB=SLECNTL TAPE=180 -
           ID='SALES-CNTL-APPL'
    This causes a private library (sales-control applications, in this
    example) to be dumped onto the same tape.
```

Figure 5-9. Creating a Backup Copy Using the Librarian – Stand-Alone

Step 12. Obtain a Listing of the History File

Make it a habit to have MSHP print a listing of the history file's contents whenever you have done an installation under MSHP control. Keep this listing on file for quick reference. MSHP writes this output to the device assigned to SYSLST. Sample job:

```
// JOB RETRACE
// EXEC MSHP
RETRACE
/*
/&
```

Step 13. Obtain Library-Directory and Label-Information Listings

To have up-to-date system records readily available as hard copy, submit a job stream as shown in Figure 5-10.

```
    // JOB SYSTEM STATUS LISTS
    // EXEC LIBR
(1) LISTDIR LIB=(IJSYSRS SLESTAT STCKINV) UNIT=SYSLST
    /*
(2) // EXEC LSERV
    /*
(3) // ASSGN SYS004,SYSRES
(4) // ASSGN SYS005,SYSLST
(5) // EXEC LVTOC
    /&
```

```
(1) LISTDIR LIB=(IJSYSRS SLESTAT STCKINV) UNIT=SYSLST
    Requests a sorted listing of the directories of the libraries IJSYSRS
    (the system library), SLESTAT, and STCKINV.  Replace these library
    names by the ones used at your location.

(2) // EXEC LSERV
    Requests a listing of the contents of the label information area.

(3) // ASSGN SYS004,SYSRES
    LVTOC uses SYS004 as logical unit for input.  The statement makes the
    system-residence volume's VTOC available as input.

(4) // ASSGN SYS005,SYSLST
    Makes the SYSLST device available to the LVTOC utility.

(5) // EXEC LVTOC
    Requests a listing of the contents of the VTOC on the input volume.

    To get a listing of VTOCs on additional volumes, code and submit ad-
    ditional LVTOC requests.
```

Figure 5-10. Obtaining Library-Directory and Label-Information Listings

Step 14. Loading Phases into the SVA

This is an optional, performance-tuning step. You can do this kind of system tuning only if you installed also the GENERATION library of VSE/Advanced Functions.

The system uses predefined load lists for loading system-required phases into the SVA during system start-up. You might consider specifying phases of your own or additional system phases to be so loaded by including their names into a load list.

The system as shipped includes a procedure that causes a selected set of frequently used B- and C-transient phases to be loaded into the SVA. For any of these phases to be executed, the system moves that phase to an area within the supervisor. This service of moving phases from the SVA is referred to as the fast-fetch facility.

SVA Load List for User-Written Phases: To be included into a cataloged load list, a phase must be cataloged in the sublibrary IJSYSRS.SYSLIB. Having these phases loaded by including a SET SDL command in a partition start-up (ASI) procedure may therefore be the better approach.

To create a load list of your own, proceed as follows:

1. Submit the job below. This gives you a listing of the names of the phases that the system loads into the SVA during start-up. Check this listing to make sure you do not specify the names of phases that are already included in one of the IBM-supplied load lists. Control statements to be used:

   ```
   // JOB PRINT LOAD-LIST CONTENTS
   // EXEC LIBR
   ACCESS  SUBLIB = IJSYSRS.SYSLIB
   LIST    $SVA*.PHASE
   /*
   /&
   ```

2. Assemble and catalog your own load list.

   To do this, use a job similar to the one shown in Figure 5-11 on page 5-26. The sample job assumes that the required assembler work-file extents are permanently defined and that permanent assignments for them exist.

```
    // JOB    BUILD LOAD LIST
    LIBDEF PHASE,CATALOG=IJSYSRS.SYSLIB,SEARCH=GENLIB.G1$301
    // OPTION CATAL
    // EXEC PGM=ASSEMBLY
             SPACE
    SIPL     TITLE '$SVA0000 PRIVATE SVA LOAD LIST'
    *       10 — 16 ───────────── Column ──────────────────72
    *        |     |                                         |
    *        |     |                                         |
             SPACE
(1)          SVALLIST $SVA0000,(phase01),(phase02),(phase03),    C
                   (phase04),(phase05),(phase06),(phase07),      C
                   ...  ...  ...                                 C
                   (phasenn)
             END
    /*
    // EXEC LNKEDT
    /&
```

---

(1) SVALLIST $SVA0000,(phase01),(phase02),...
The source code for creating a load list consists of just this macro. Code the macro as shown; supply, for the second through nth operands the names of the phases that you want to be loaded into the SVA automatically during system start-up.

In theory, the number of phases that you can specify is unlimited. However, the more phases your load list includes, the larger will be your system's SVA.

Figure 5-11. Creating SVA Load List $SVA0000

The Fast-Fetch Facility: The facility is supported by the cataloged procedure FASTFTCH, which is a member of sublibrary IJSYSRS.SYSLIB. The procedure builds SDL entries for a selected set of frequently used B- and C-transient phases. You may, if this is desirable, build a procedure of your own, having the system load a different set of phases. To display the contents of the procedure, use the following job:

```
// JOB DISPLAY FASTFETCH
// EXEC LIBR
ACCESS SUBLIB = IJSYSRS.SYSLIB
LIST FASTFTCH.PROC
/*
/&
```

How to code a procedure is described in the IBM publication *VSE/Advanced Functions System Management Guide.*

Loading ISAM Phases into the SVA: To improve the performance of programs that access ISAM files, consider loading the following ISAM phases into the SVA:

Phase Name	Function of the Phase
IJHAVBBF	HOLD=YES and CORDATA=YES
IJHAVBBS	HOLD=YES, but CORDATA=YES omitted
IJHAVBCF	CORDATA=YES, but HOLD=YES omitted
IJHAVBCS	Neither HOLD=YES nor CORDATA=YES
IJHAVGBF	Two I/O areas, HOLD=YES, and CORDATA=YES
IJHAVGBS	Two I/O areas, HOLD=YES, but CORDATA=YES omitted
IJHAVGCF	Two I/O areas, CORDATA=YES, but HOLD=YES omitted
IJHAVGCS	Two I/O areas and neither HOLD=YES nor CORDATA=YES
IJHZXGZS	IOROUT=LOAD

Include into one of your JCL start-up procedures a SET SDL command followed by phasename, SVA statements for the phases that you want to reside in the SVA. For more details about loading phases into the SVA, see the IBM publication *VSE/Advanced Functions System Management Guide.*

Step 15. Create a Backup Copy

At this point of the procedure, your system is ready for operation, and you should consider creating (possibly once more) a backup copy of the system. How to do this is described under "Step 11. Create a Backup Copy" on page 5-22.

After completion of this step, you can take up normal system operation. You can, for example, install additional IBM licensed programs to complement your newly installed VSE/Advanced Functions or replace an early release of a licensed program with its current release.

The On-line Installation Procedure

As mentioned earlier in this chapter, you can use this procedure only if you have migrated to Version 2 of VSE/Advanced Functions previously and your Version 2 system is operational.

Run the installation jobs in a partition of 640K bytes or larger. MSHP uses other system programs (such as the librarian and the linkage editor) as required.

Where applicable, the sample jobs include statements for running the installation jobs under VSE/POWER. If your background partition is not under VSE/POWER control when you run these jobs, the system ignores those statements.

The procedure assumes that the required disk volumes have been initialized and are on-line.

Step 1. Restore the System Residence File

This step restores, from tape to disk, the PRODUCTION library of VSE/Advanced Functions and, optionally, its GENERATION library. You restore the library (or libraries) under control of MSHP:

1. Mount the distribution tape on an available tape drive.
2. Run a job similar to the one shown in Figure 5-12 on page 5-30.

When this step is complete, then:

- The libraries are restored to disk.
- The history file supplied on the distribution tape has been restored and merged into the currently installed system's history file. In other words, your system's history file reflects the status of the new system merged with the old system. MSHP deletes the entries of any components that are replaced by components of the newly installed system.

The remaining steps of the procedure are concerned with making the newly restored VSE/Advanced Functions operational while your system does productive work. Following are explanations for Figure 5-12.

```
    * $$ JOB  JNM=AFREST,CLASS=0
    // JOB    INSTALL NEW SYSTEM
(1) // ASSGN  SYS006,cuu
(2) // DLBL   IJSYSR1,'A5666301.PRODUCTION.LIBRARY',99/365
(2) // EXTENT ,SYSRES,,,1,1653
(3) // DLBL   GENLIB,'A5666301.GENERATION.LIBRARY'
(3) // EXTENT ,WRKFL1,,,1,1700
(4) // EXEC   LIBR
(4) DEFINE LIB=IJSYSR1
(4) DEFINE LIB=GENLIB
(4) /*
    // EXEC   MSHP
(5) INSTALL SYSRES FROMTAPE ID='301H07' -
                   PRODUCTION INTO=IJSYSR1 -
                   GENERATION INTO=GENLIB
    /*
    /&
    * $$ EOJ
```

The numbers within parentheses refer to explanations given in the text preceding this illustration – references (1) and (2) – or following the illustration – references (3) through (5).

(1) // ASSGN SYS006,cuu
For cuu supply channel and unit address of the tape drive on which you mounted the distribution tape.

(2) // DLBL IJSYSR1,'A5666301.PRODUCTION.LIBRARY',99/365
// EXTENT ,SYSRES,,,1,1653
The DLBL and EXTENT statements needed to define the address and the space of the new PRODUCTION library. SYSRES (in the extent statement) is the disk pack's volume serial number.

The library as shipped includes a free space of 20 % of the total library.

The number-of-track specification in the EXTENT statement assumes that an IBM 3380 is used as system residence device. Look up the Program Directory for the value that applies to your location; you find this value in the list of label-information procedures for the various disk devices. The value in this example was chosen arbitrarily.

Figure 5-12 (Part 1 of 2). Restoring the Distribution Tape to Disk – On-line

(3)
```
// DLBL   GENLIB,'A5666301.GENERATION.LIBRARY'
// EXTENT ,WRKFL1,,,1,1700
```
The same as (2) above, but for the GENERATION library.
Using a work disk facilitates taking the library off-line when the necessary generation activities are complete. If the library has not been modified by these activities, you can use the distribution tape as backup.
The sample job assumes that the GENERATION library of the currently operational VSE/Advanced Functions was originally installed on the same volume and at the same address. Ensure that the library is off-line when you run this installation job; this avoids that the library is overwritten by the newly restored GENERATION library.

(4)
```
// EXEC LIBR
DEFINE LIB=IJSYSR1
DEFINE LIB=GENLIB
/*
```
This set of statements formats the libraries named IJSYSR1 and GENLIB. There is no need for you to supply the statements; MSHP generates them automatically if you omit them.

(5)
```
INSTALL SYSRES FROMTAPE ID='HD301H07' -
               PRODUCTION INTO=IJSYSR1 -
               GENERATION INTO=GENLIB
```
The ID shown here applies to VSE/Advanced Functions 2.1; it is release-dependent. Verify this ID by looking up the installation procedure given in the Program Directory.

Figure 5-12 (Part 2 of 2). Restoring the Distribution Tape to Disk – On-line

Step 2. Copy Location-Specific Library Members

Certain location-specific data need to be stored in the active system residence file. Such data, primarily ASI procedures, is to be included into the newly restored system residence file before you can make it operational.

Figure 5-13 is a sample job for copying members of type PROC to a newly restored system sublibrary under control of the currently operational system. Supply COPY commands for members of other types if any such members have to be copied. Candidates are, for example: location specific FCB- or UCB-image phases that you want to be loaded during IPL; an SVA load list (see also "Step 14. Loading Phases into the SVA" on page 5-25, a step of the stand-alone installation procedure).

```
     * $$ JOB  JNM=CPYMEMB,CLASS=0
     // JOB    COPY ASI PROCEDURES
(1)  // DLBL   IJSYSR1,'A5666301.PRODUCTION.LIBRARY',99/365
(1)  // EXTENT SYS012
(1)  // ASSGN  SYS012,cuu
     // EXEC   LIBR
     CONNECT   S=IJSYSRS.SYSLIB : IJSYSR1.SYSLIB
(2)  COPY      IPLSUP3.PROC REPLACE=YES
(2)  COPY      $0JCL1.PROC REPLACE=YES
(2)  COPY      $1JCL1.PROC REPLACE=YES
      ... ... ...
     /*
     /&
     * $$ EOJ
```

```
(1) // DLBL   IJSYSR1,'A5666301.PRODUCTION.LIBRARY',99/365
    // EXTENT SYS012
    // ASSGN  SYS012,cuu
```
The three statements define your system's PRODUCTION library as the target library. For cuu, supply the channel and unit number of the new system-residence device.

(2) `COPY ... REPLACE=YES`
Specify REPLACE=YES if you have used the IBM-defined default procedure names for cataloging your location's ASI procedures.

Figure 5-13. Copy Library Members to the New System Residence File – On-line

Step 3. Start Up the New System

Up to this point, your installation activities have taken place under control of your currently installed system. To start up the newly restored system, you have to:

1. Shut down the system currently in control

   You may want to do this at the end of a day or at the end of a certain shift. For details, see the IBM manual *VSE/Advanced Functions Operation.*

2. Perform system start-up as usual

   Supply as load address the channel and unit number of the disk drive whose volume contains the newly installed VSE/Advanced Functions.

   The new system uses the label information area referred to in the DLA command of the IPL ASI procedure you use. If this procedure requests a supervisor generated at your location, the system most likely prompts you for the name of a supervisor (because the requested supervisor is not cataloged in the system sublibrary of the IPLed system). Respond to the system prompt (message 0I03D) with the name of the IBM-supplied supervisor which best meets your location's needs. Some of your jobs may not perform as before or even fail until you have assembled and cataloged a suitable supervisor under the newly installed system.

   How to assemble and catalog a supervisor is described under "Step 5. Tailor Your Own Supervisor" on page 5-35.

Step 4. Personalize the System History File

Any information that remains unchanged need not be redefined; if none of the identifying information has changed, then omit this step.

To personalize the restored system's history file (for ease of system identification on listings) use control statements as shown in Figure 5-14.

```
     * $$ JOB  JNM=PERSHIST,CLASS=0
    // JOB PERSONAL HISTORY
    // EXEC MSHP
(1) PERS 'customer name' -
(2)      ADDR='location' -
(3)      PHONE='extension' -
(4)      PROGR='programmer name' -
(5)      ENV='environment'
    /*
    /&
    * $$ EOJ
```

```
(1) PERS 'customer name'
    For 'customer name', specify a company identification of no more than
    20 characters.

(2) ADDR='location'
    For 'location', specify your address; your specification may not be
    longer than 45 characters.

(3) PHONE='extension'
    For 'extension', specify your telephone number.  Your specification may
    not be longer than 17 characters.

(4) PROGR='programmer name'
    For 'programmer name', specify the responsible programmer's name; your
    specification may not be longer than 24 characters.

(5) ENV='environment'
    For environment, specify information related to your system.  For
    example: the component level code(s) describing your software, the type
    code of your computer system's processing unit and system residence
    device.  Your specification may not be longer than 62 characters.
```

Figure 5-14. Personalizing the History File — On-line

Step 5. Tailor Your Own Supervisor

This step does not apply if you install only the PRODUCTION library.

The job for this step, which assembles a supervisor and catalogs it into the sublibrary IJSYSRS.SYSLIB, is to be submitted from the SYSRDR/SYSIPT device. A sample job is shown in Figure 5-15 on page 5-36; it causes the following to be recorded in the system's history file:

- The tailoring of phase $$A$SUP1.
- The macros (such as FOPT, IOTAB) that were included by the assembler.
- All data between the statements EXECUTE ... and /$.

Make it a point to submit the job only after having verified your source code. You may have to repeat this step if the assembler detects an error. An error in the source code causes the assembler to write an appropriate MNOTE reference at the end of the assembler output listing. The section "MNOTEs During Assemblies" on page 5-2 discusses this topic in more detail.

Following are explanations to Figure 5-15:

(1) // ASSGN SYSxxx,cuu
Assignments for work files (on disk) as follows:

Logical Unit	For Use by	File Name in EXTENT Statement
SYS001	Assembler and linkage editor	IJSYS01
SYS002	Assembler	IJSYS02
SYS003	Assembler	IJSYS03
SYS004	MSHP	IJSYS04
SYSLNK	Linkage editor	IJSYSLN

(2) TAILOR 5666-301-06-H07 PHASE=$$A$SUP1 KEEPDATA
KEEPDATA, an optional specification, causes the source code to be stored in the system's history file. MSHP uses that code if retailoring is to be done later on.

(3) RESOLVES 'SUPERVISOR 1 GENERATION'
An example of an optional comment statement.

```
     * $$ JOB  JNM=TAILSUPV,CLASS=0
     // JOB $$A$SUP1
(1)  // ASSGN SYS001,cuu
(1)  // ASSGN SYS002,cuu
(1)  // ASSGN SYS003,cuu
(1)  // ASSGN SYS004,cuu
(1)  // ASSGN SYSLNK,cuu
     // ASSGN SYSPCH,IGN
     // OPTION CATAL
     // EXEC MSHP
(2)  TAILOR 5666-301-06-H07 PHASE=$$A$SUP1 KEEPDATA
(3)  RESOLVES 'SUPERVISOR 1 GENERATION'
     EXECUTE (ASSEMBLY LNKEDT) XREF
     *        10 — 16 ——————— Column ———————————————— 72
     *         |     |                                 |
     *         |     |                                 |
               TITLE 'VSE SUPERVISOR SOURCE CODE FOR $$A$SUP1'
               SUPVR ID=1,                             C
                     MICR=1419,                        C
                      ... ... ...
               FOPT  DASDSHR=YES,                      C
                     TTIME=BG,                         C
                      ... ... ...
               IOTAB IODEV=254,                        C
                     NPGR=600
               END
     /$
     /*
     /&
     * $$ EOJ
```

The numbers within parentheses refer to explanations given in the text preceding this illustration.

Figure 5-15. Assembly of a Supervisor – On-line

Step 6. Assemble and Catalog IOCS Modules

Perform this step only if:

1. The release you get includes the support of a new unit-record device (a printer, for example);

2. Your computer system's configuration includes this new device;

3. Your VSE system does not include a compatible I/O module (for a PRT1 printer, for example).

You may assemble and catalog IOCS modules at any time after completion of the installation. Ignore this step if there is no need for the assembly and cataloging of IOCS modules under control of MSHP.

To catalog an IOCS module under control of MSHP, you use the program's TAILOR function. Run a job similar to the one shown in Figure 5-16 on page 5-38. Submit the job from the SYSRDR/SYSIPT device.

Make it a point to submit the job only after having verified your source code. You may have to repeat this step if the assembler detects an error. An error in the source code causes the assembler to write an appropriate MNOTE reference at the end of the assembler output listing. The section "MNOTEs During Assemblies" on page 5-2 discusses this topic in more detail.

```
    * $$ JOB  JNM=TAILIOM,CLASS=0
    // JOB TAILIOCS
    // OPTION DECK,LIST,LOG,NOXREF
(1) // ASSGN  SYS001,cuu,VOL=vol-id
(1) // ASSGN  SYS002,cuu,VOL=vol-id
(1) // ASSGN  SYS003,cuu,VOL=vol-id
(1) // ASSGN  SYS004,cuu,VOL=vol-id
(1) // ASSGN SYSLNK,cuu
    // ASSGN SYSPCH,IGN
    // EXEC   MSHP
(2) TAILOR 5666-301-02 MOD=IJCFAOI7 KEEPDATA
    EXECUTE (ASSEMBLY LIBR) NOX
    *       10 — 16 ———————— Column ——————————————————— 72
    *        |     |                                     |
    *        |     |                                     |
             PRINT NOGEN
             CDMOD CTLCHR=ASA,                           C
                   IOAREA2=YES,                          C
                   DEVICE=3505,                          C
                   SEPASMB=YES
             END
    /$
(3) TAILOR 5666-301-02 MOD=IJFFZZZZ KEEPDATA
    EXECUTE (ASSEMBLY LIBR) NOX
             PRINT NOGEN
             CDMOD RECFORM=UNDEF,                        C
                   DEVICE=3881,                          C
                   SEPASMB=YES
             END
    /$
    /*
    /&
    * $$ EOJ
```

(1) // ASSGN SYS00n,cuu,VOL=vol-id
These are the same work-file assignments as for the preceding step (tailoring your own supervisor).

(2) TAILOR 5666-301-02 MOD=IJxxxxxx KEEPDATA
Specifying KEEPDATA, which is optional, causes the source code (of IJCFAOI7 and IJFFZZZZ, respectively) to be stored in the system's history file. MSHP uses that code if retailoring is to be done later on.

Figure 5-16. Assembly of IOCS Modules – On-line

Step 7. Obtain a Listing of the History File

Make it a habit to have MSHP print a listing of the contents of the history file whenever you have done an installation under control of MSHP. Keep this listing on file for quick reference. MSHP writes this output to the device assigned to SYSLST. Sample job:

```
* $$ JOB  JNM=LSTHIST,CLASS=0
// JOB RETRACE
// EXEC MSHP
RETRACE
/*
/&
* $$ EOJ
```

Step 8. Copy Private Data from the Old System Residence Volume

Do this step if your old system-residence volume contains files that should reside also on your new system residence volume. Your SA-dump analysis-routine file, for example, is a candidate for being copied to the new system residence volume.

Use the IBM licensed program *Fast Copy Data Set* (FCOPY) for this purpose or any other suitable file-copy program. The FCOPY program copies data from one disk volume to another of the same type. For more information about the program, see the IBM publication *VSE/Fast Copy Data Set Installation Reference.*

Step 9. Obtain Library-Directory and Label-Information Listings

To have up-to-date system records readily available as hard copy, submit a job as shown in Figure 5-17.

```
    * $$ JOB JNM=SYSLIST,CLASS=0
    // JOB SYSTEM STATUS LISTS
    // EXEC LIBR
(1) LISTDIR LIB=(IJSYSRS SLESTAT STCKINV) UNIT=SYSLST
    /*
(2) // EXEC LSERV
    /*
(3) // ASSGN SYS004,SYSRES
(4) // ASSGN SYS005,SYSLST
(5) // EXEC LVTOC
    /&
    * $$ EOJ
```

```
(1) LISTDIR LIB=(IJSYSRS SLESTAT STCKINV) UNIT=SYSLST
    Requests a sorted listing of the directories of the named libraries:
    IJSYSRS, the system library; SLESTAT; and STCKINV.  Replace these
    names by the ones used at your location.

(2) // EXEC LSERV
    Requests a listing of the contents of the label information area.

(3) // ASSGN SYS004,SYSRES
    LVTOC uses SYS004 as logical unit for input.  The statement makes the
    system-residence volume's VTOC available as input.

(4) // ASSGN SYS005,SYSLST
    This statement makes the SYSLST device available to the LVTOC utility.

(5) // EXEC LVTOC
    Requests a listing of the contents of the VTOC on the input volume.
    To get a listing of VTOCs on additional volumes, code and submit ad-
    ditional LVTOC run requests.
```

Figure 5-17. Obtaining Library-Directory and Label-Information Listings – On-line

Step 10. Create a Backup Copy

At this point of the procedure, it may be advisable to create a backup tape of your new system residence file. To create a backup copy, proceed as follows:

1. Mount the tape to be used on an available tape drive.

2. Run a job similar to the one shown in Figure 5-18.

You can use the created backup tape for stand-alone restore of the system residence file, should the need arise.

On-line installation of VSE/Advanced Functions is now complete, and you can resume normal operation under control of the newly installed release of VSE/Advanced Functions.

```
     * $$ JOB  JNM=BACKUP,CLASS=0
     // JOB BACKUP SYSTEM
     // EXEC LIBR
(1)  BACKUP LIB=IJSYSR1 TAPE=180 RESTORE=STANDALONE -
            INCLUDE=HISTORY
     /*
     /&
     * $$ EOJ
```

```
(1) BACKUP LIB=IJSYSRS TAPE=180 RESTORE=STANDALONE -
           INCLUDE=HISTORY
    This causes the system residence file, including the system history
    file, to be dumped onto the tape on the drive at the hardware address
    180 - the available tape drive in this example.

    By specifying RESTORE=STANDALONE, you cause an IPL routine and a
    PRODUCTION library restore routine to be written onto the backup tape.
```

Figure 5-18. Creating a Backup Copy of the System Residence File – On-line

Chapter 6. Installing Licensed Programs and Features

An IBM licensed program (which, in this context, refers also to a feature of a licensed program), is normally distributed on magnetic tape as a distribution library (DLIB). On this tape, the code of the product is stored as a library consisting of:

- A PRODUCTION sublibrary, which contains the code of the program ready for loading and running without modification, and
- Optionally, a GENERATION sublibrary, which contains the code of the program in a format suitable for modification.

A licensed program may be available on a tape in so-called SYSIN format. To install a program in SYSIN format, use the separate installation instructions that are shipped with the distribution tape. This section deals with installing a product distributed as a DLIB tape.

A DLIB tape always includes a history file containing service and installation data about the product. MSHP merges the product's history-file data into the existing system history file.

Consider installing each of your licensed programs in a separate sublibrary. This enables you to run small backup jobs on a program-product level under control of MSHP. The backup tape you get includes the product's history file. MSHP builds this file by extracting the product-related entries from your system's history file.

To install a licensed program, proceed as follows:

1. Submit the installation job

   This is a job similar to the one shown in Figure 6-1 on page 6-2. The sample job includes VSE/POWER JECL statements, which you may want to use if you run your job in a partition under control of VSE/POWER. If you use a partition outside control of VSE/POWER, the system ignores these statements.

   MSHP makes use of the librarian and, if necessary, also of the linkage editor; therefore, run the job in a partition of a size of 640K or larger.

2. Produce a product-level backup tape

   Submit a job similar to the one shown in Figure 6-2 on page 6-5. Restoring a licensed program from a backup tape is, in fact, a reinstallation of the product.

3. Verify the installation

   To do this, obtain a printout of the history file by issuing a RETRACE PRODUCT request.

Given below are the explanations for Figure 6-1:

```
     * $$ JOB JNM=INSTALL,CLASS=0
     // JOB    INSTALL LICENSED PROGRAM
(1)  // ASSGN  SYS006,cuu
     // OPTION  STDLABEL=ADD
(2)  // DLBL   PRPNLIB,'PROGRAM.LIBRARY.PRODUCT'
(2)  // EXTENT ,PRPDLB,,,1,1899
(3)  // DLBL   PRGNLIB,'PROGRAM.LIBRARY.GENER'
(3)  // EXTENT ,PRPLB2,,,1,1899
     // OPTION  USRLABEL
(4)  // EXEC   LIBR
(4)  DEFINE LIB=PRPNLIB
(4)  DEFINE LIB=PRGNLIB
     /*
     // EXEC   MSHP
(5)  INSTALL PRODUCT FROMTAPE ID='tapefile-id' -
             PRODUCTION INTO=PRPNLIB.SLIB1 -
             GENERATION INTO=PRGNLIB.SLIB2
(6)  RETRACE PRODUCT
     /*
     /&
     * $$ EOJ
```

```
    The numbers within parentheses refer to explanations given in the
    text preceding this illustration.

(1) // ASSGN  SYS006,cuu
    For cuu, supply the channel and unit address of the tape drive
    on which you mounted the distribution tape.
```

Figure 6-1 (Part 1 of 3). Installing a Licensed Program

(2) // DLBL   PRPNLIB,'PROGRAM.LIBRARY.PRODUCT'
    // EXTENT ,PRPDLB,,,1,1899
    The statements define the disk residence of the library PRPNLIB. These statements (and also the ones that call the librarian for formatting the library) not required if the library has been created previously and they are stored in your system's label information area.

    For PRPNLIB and 'PROGRAM.LIBRARY.PRODUCT' in the DLBL statement and for PRPDLB and 1899 in the EXTENT statement, use location-specific values.

    The preceding OPTION statement ensures that the subsequent label-information statements up to the second OPTION statements are stored permanently in the system's label-information area.

(3) // DLBL   PRGNLIB,'PROGRAM.LIBRARY.GENER'
    // EXTENT ,PRPLB2,,,1,1899
    The same as (2) above, but for the library PRPGNLIB

(4) // EXEC   LIBR
    DEFINE LIB=PRPNLIB
    DEFINE LIB=PRGNLIB
    This set of statements is not required if the target libraries have been defined previously. For PRPNLIB and PRPGNLIB, use location-specific names.

(5) INSTALL PRODUCT FROMTAPE ID='tapefile-id' -
    For 'tapefile-id,' specify the identifier which IBM has used when the distribution tape was built. Check the installation sample job in the <u>Program Directory</u> for this identifier.

    PRODUCTION INTO=PRPNLIB.SLIB1 -
    GENERATION INTO=PRGNLIB.SLIB2
    Installation of a licensed program's GENERATION library requires that you install also the PRODUCTION library, either at the same time as shown or earlier. A licensed program's PRODUCTION and GENERATION libraries must be installed into separate sublibraries.

Figure 6-1 (Part 2 of 3). Installing a Licensed Program

To have MSHP install:

- Only the PRODUCTION library of a licensed program, omit the GENERATION ... line.
- The product's PRODUCTION and GENERATION libraries as separate sublibraries of one library, specify INTO=libname and omit the GENERATION ... and PRODUCTION ... lines. For libname, specify the name of the target library.

If the product's history file indicates a prerequisite (such as prior installation of the base product for a feature), MSHP verifies that this prerequisite is met. MSHP merges the distribution libraries with the existing base-product sublibraries if this is indicated by the product identifier in the product's history file.

(6) RETRACE PRODUCT
If you install several licensed programs at a time, submit this statement in the installation job for the last one. The output listing gives you, besides other information, the IBM-assigned product identifier. You need this identifier for a product-level backup.

Figure 6-1 (Part 3 of 3). Installing a Licensed Program

```
     * $$ JOB JNM=PRBCKUP,CLASS=0
     // JOB    LICENSED PROGRAM BACKUP
(1)  // ASSGN  SYS006,cuu
     // EXEC   MSHP
(2)  BACKUP PRODUCT=product-id ID='tapefile-id' PRODUCTION
     /*
     /&
     * $$ EOJ
```

---

(1) // ASSGN SYS006,cuu
For cuu, supply the address of the tape drive on which you mounted the output (product-backup) tape.

(2) BACKUP PRODUCT=product-id ID='tapefile-id' PRODUCTION

Specify for:

product-id - The installed product's identifier. Look up your RETRACE PRODUCT output listing for this identifier.

'tapefile-id' - An identifier of your own choosing. MSHP records this identifier on the backup tape as a file identifier. If a restore run should become necessary, MSHP can position the tape by way of this identifier.

In the statement, specify:

PRODUCTION
If a backup of only the product's PRODUCTION library is required.

GENERATION
If a backup of only the product's GENERATION library is required.

Neither of the two
If the licensed program has only a PRODUCTION library or, if it has also a GENERATION library, you want a backup of both libraries.

Figure 6-2. Sample Job for a Product-Level Backup

Chapter 7. Installing Service Changes

IBM distributes service changes for VSE/Advanced Functions and related licensed programs as either of the following:

One or a number of program temporary fixes (PTFs) on tape.
An authorized program analysis report (APAR) fix.

A PTF contains one or more phases, modules, or macros, each of them replacing an existing phase, module, or macro, respectively. An APAR is an update to a phase, a module, or a macro. How to handle PTFs and APARs is summarized below.

- Handling of PTFs

  Normally, PTFs are shipped on a service tape. This tape contains (in the sequence as listed):

  A history file or a null file;
  A preventive service document;
  A dummy file;
  An EXCLUDE list or a null file;
  PTF cover letters or a null file;
  PTFs;
  Two dummy files.

  Note: MSHP can install PTFs from a service tape only if the PTFs are stored on the tape as the sixth file with a block size of 10320 bytes. Service tapes which you received from IBM before you migrated to Version 2 may be unsuitable for processing on your Version-2 system.

  When you receive a service tape, then:

  1. Have MSHP produce a listing of the service document and give this printout a careful reading. This helps you in planning and performing the task of installing required PTFs.

  2. Install those PTFs which correct system problems, if you have any at your location, and which avoid potential problems in the future. You may install all PTFs supplied by IBM if this is desirable. This approach is referred to as preventive service.

     The service tape as shipped by IBM may not include the PTFs whose installation on your system is a prerequisite for one or more of the PTFs you want to install. In this case, MSHP does not install the affected PTF and informs you by a message.

If you know that prerequisite PTFs exist on another service tape and that these PTFs are not yet installed, have MSHP scan one or more additional tape volumes for the prerequisite PTFs and retrieve them for installation.

You may receive, from IBM, also a single PTF. You get this PTF in the form of an executable job which invokes MSHP.

- Handling of APAR or Local Fixes

  Normally, such fixes are not distributed in machine readable form; you install them using the CORRECT function of MSHP.

- Revokable or Irrevocable Installation

  The REVOKABLE option of MSHP produces so called backout jobs of PTFs that are being installed, if this option is specified. Use the option only if you install just one or few PTFs for the purpose of solving a specific problem.

  MSHP writes backout jobs onto the tape mounted on the drive to which SYS004 is assigned. You can use this tape as input to MSHP should there be need for a backout of a PTF. Installing a backout PTF amounts to a re-installation of the library member(s) replaced by the installation of the PTF.

  Note: Do not install the backout job for a PTF that is a pre- or co-requisite for other PTFs or has comparable APAR/local fix dependencies.

  This chapter gives sample jobs for both the installation of PTFs and the installation of backout jobs. It gives sample jobs for installing and removing an APAR or a local fix.

- Preparing the System

  Before the installation of service changes, you should:

  – Produce a backup on tape

    Ensure that you have an up-to-date backup of the sublibrary (or licensed program) on which you want to install a service change. Perform a librarian BACKUP run (see "Step 11. Create a Backup Copy" on page 5-22 for a sample job) to get a backup of VSE/Advanced Functions; perform an MSHP BACKUP run (see Figure 6-2 on page 6-5 for a sample job) to get a backup of an individual licensed program.

- Restore the generation library

  For installation of service changes, MSHP requires the product's generation (sub)library, if there is one, to be on-line. To restore the generation library of VSE/Advanced Functions, proceed as described under "Step 6. Restore the GENERATION Library" on page 5-15. The library may be taken off-line again when the service run is complete.

There is no need to define the libraries or sublibraries that hold the products which are to be serviced. MSHP establishes required search chains based on the information recorded in the history file; it uses the services of the librarian to actually delete replaced members and to catalog new, replacement members.

Service-Related Activities: The chapter includes a discussion of activities that IBM recommends after successful completion of a service run; it gives sample jobs for history-related activities for which there may be a need occasionally at your location.

Sample Jobs: Where applicable, the sample jobs in this chapter include statements for running the installation jobs under VSE/POWER. If your service partition is not under VSE/POWER control when you run these jobs, the system ignores those statements.

Handling of PTFs

This activity includes the installation of PTFs and the backout of installed PTFs, if this is necessary.

Installing PTFs from a Service Tape

For the installation of PTFs, IBM recommends that you proceed as follows:

1. List the service document

   Submit a job similar to the one shown in Figure 7-1 on page 7-5. The job produces, on the device assigned to SYSLST, a printout of service documentation as requested by the MSHP LIST statement.

2. Examine the service document

   The information contained in the printed document helps you decide which of the PTFs on the tape are to be installed.

   You may want to install only those PTFs which correct a problem or prevent a potential problem. In that case, you should prepare a list of the PTFs you want to be included (by way of INCLUDE statements) or excluded (by way of EXCLUDE statements), whichever is the shorter list.

Write INCLUDE statements in the form

```
INCLUDE PTF=(UDnnnnn,UDnnnnn,...)
```

Write EXCLUDE statements in the form

```
EXCLUDE PTF=(UDnnnnn,UDnnnnn,...)
```

To install all PTFs on the service tape as a preventive service, do not supply any INCLUDE or EXCLUDE statements.

3. Set up and run the installation job

   To install the required PTFs, set up and submit a job similar to the one shown in Figure 7-2 on page 7-6. The sample assumes that:

   - Label information for the affected libraries is stored in the system's label information area.
   - The affected sublibraries are on-line.

```
     * $$ JOB JNM=LSTSVCE,CLASS=0
     // JOB   LIST SERVICE INFORMATION
(1)  // ASSGN SYS006,cuu
     // EXEC  MSHP
(2)  LIST SERVICETAPE -
          DOCUMENT COVER SEPARATE
(3)  PTF=(UD12345,UD45678,...)

     /*
     /&
     * $$ EOJ
```

---

(1) // ASSGN SYS006,cuu
Assign SYS006 to the tape drive on which you mounted the service tape.

(2) LIST SERVICETAPE DOCUMENT COVER SEPARATE
Specifying:

DOCUMENT
Produces a printout of the service document stored on the service tape.

COVER
Produces a printout of PTF cover letters (see also reference (3) below).

SEPARATE
Causes a new page to be started for each PTF cover letter that is to be printed.

(3) PTF=(UD12345,UD45678,...)
This detail control statement causes MSHP to print only the cover letters of those PTFs that are specified. If you omit this statement, MSHP prints all PTF cover letters.

Figure 7-1. Sample Job for Listing the Service Document

```
     * $$ JOB JNM=INSTSVE,CLASS=0
     // JOB   INSTALL SERVICE
(1)  // ASSGN SYS002,cuu
(2)  // ASSGN SYS006,cuu
     // EXEC  MSHP
(3)  INSTALL SERVICE TAPES=2
(4)  INCLUDE PTF=(UD12345,UD45678,...)
     /*
     /&
     * $$ EOJ
```

---

(1) // ASSGN SYS002,cuu
Assign SYS002 to a work disk for use by MSHP.

(2) // ASSGN SYS006,cuu
Assign SYS006 to the tape drive on which you mounted the service tape.

(3) INSTALL SERVICE TAPES=2
You may supply the statement with REVOKABLE specified (for example: INSTALL SERVICE REVOKABLE). Do this only if you install just one or few PTFs and you are sure that none of the PTFs being installed is a requirement for other PTFs or has APAR/local-fix dependencies. An attempt to revoke a PTF with such dependencies may result in a down-leveled (inoperative) system.

Specifying REVOKABLE requires that you mount an extra tape to which MSHP can write the created backout job(s). Assign SYS004 to the tape drive you use.

TAPES=2 indicates to MSHP that it is to scan and process two service tapes. You can specify up to nine service tapes. The tapes are to be mounted one after the other, in response to MSHP's mount request, on the tape drive assigned to SYS006.
When MSHP has scanned the last tape, it issues another mount request. Now mount the tapes for processing, one after the other, in response to the mount requests from MSHP.

(4) INCLUDE PTF=(UD12345,UD45678,...)
PTFs defined here are installed by MSHP. All other PTFs stored on the service tape are not installed. If you were using an EXCLUDE statement, MSHP would install all PTFs not defined in the statement.

Figure 7-2. Sample Job for Installing PTFs

Restarting a PTF Installation Run

The installation of PTFs may require modules to be link-edited into phases. Before this link-editing under MSHP control starts, MSHP takes a checkpoint. Should this link-editing fail, then MSHP terminates PTF installation, but allows you to set up the installation job again at the recorded checkpoint. To restart the installation process at this checkpoint, submit a job similar to the one shown below:

```
* $$ JOB JNM=RSTRTSV,CLASS=0
// JOB   INSTALL SERVICE
// EXEC  MSHP
INSTALL SERVICE RESTART
/*
/&
* $$ EOJ
```

For a restart, MSHP needs no input other than the INSTALL statement as shown in the above sample job.

Installing a Backout PTF

To install a backout PTF, which amounts to recataloging the library member(s) replaced by the corresponding PTF, you proceed in nearly the same way as for the installation of a PTF from the service tape:

1. Mount the MSHP created backout tape.

2. Submit a job similar to the one shown in Figure 7-3 on page 7-8.

For the MSHP job to complete successfully, it is necessary that:

- Label information for the affected library is stored in the system's label information area.

- The affected sublibrary is on-line.

```
    * $$ JOB JNM=INSTBKO,CLASS=0
    // JOB    INSTALL BACKOUT PTF
(1) // ASSGN  SYS002,cuu
(2) // ASSGN  SYS006,cuu
    // EXEC   MSHP
    INSTALL BACKOUT
(3) INCLUDE PTF=UD12345
    /*
    /&
    * $$ EOJ
```

```
(1) // ASSGN  SYS002,cuu
    Assign SYS002 to a work disk for use by MSHP.

(2) // ASSGN  SYS006,cuu
    Assign SYS006 to the tape drive on which you mounted the backout tape.

(3) INCLUDE PTF=UD12345
    MSHP installs the backout PTF that corresponds to the PTF whose number
    you specify.  Any other backout PTF stored on the backout tape is not
    installed.  If you use an EXCLUDE statement, MSHP installs any backout
    PTF whose number is not specified in the statement.
```

Figure 7-3. Sample Job for Installing a Backout PTF

Handling of APAR and Local Fixes

An APAR or a local fix, a correction of software to resolve a problem, may be developed locally or may come from IBM. To install such a fix on your system, use the CORRECT function of MSHP. MSHP, in turn, uses system programs as required to implement the fix; it supplies input to those programs based on your specifications. MSHP records the installation of a fix in the system history file.

Later on, weeks or months after you installed the fix, you may have to install a PTF that overlays this fix. If this occurs, remove the fix by using MSHP's UNDO function before you install the PTF. You can do this if you installed the fix with REVOKABLE specified either explicitly or by default. Reinstall the fix after having installed the PTF, should this be necessary.

If you install a fix revokable, MSHP does the following in addition:

- For phases and modules – It records the old and the new data.
- For macros – It writes the affected macros to SYSPCH before actually altering them.

MSHP handles a CORRECT request for a macro as a fix for an edited macro, except when a member type is specified in the AFFECTS statement. In that case, MSHP handles the request as a fix for an unedited macro.

MSHP can be requested to *expand* a *phase* or a *module* if this phase or module consists of only one CSECT. This allows you to add code to the end of the affected phase or module. You do this by specifying, in the AFFECT statement, the number of bytes by which the phase or module is to be expanded. Figure 7-4 shows an AFFECT statement used to expand a phase by 100 bytes.

This section gives sample jobs for the installation of fixes to the various types of library members. These jobs assume that permanently stored label information exists for the affected libraries. For a list of component names – IBM assigned numbers such as 5666-273-01-A45, which is used in the first sample job – refer to your MSHP RETRACE COMPONENTS listing. In this listing, this component name would be printed as 5666-27345; 45, the last two digits of the PRODUCT identifier, is the level code.

Handling a Fix for a Phase

The first sample job shown in Figure 7-4 on page 7-10 corrects phase IPW$$OT. The job causes MSHP to expand the phase by 100 bytes. The second sample job shown in Figure 7-4 removes this fix again.

Installing the Fix:

```
     * $$ JOB JNM=CORPHASE,CLASS=0
     // JOB CORRECT PHASE
     // EXEC MSHP
(1)  CORRECT 5666-273-01-A45  :  DY21001
     AFFECTS PHASES=IPW$$OT EXPAND=100
(2)  ALTER F0 9200B0F8:92F180F8
(3)  ALTER 6FA 00000000:4700C426
     RESOLVES 'ERROR ON TAPE OPEN'
     /*
     /&
     * $$ EOJ
```

Removing the Fix:

```
     * $$ JOB JNM=UNDO,CLASS=0
     // JOB UNDO FIX
     // OPTION CATAL
     // EXEC MSHP
(4)  UNDO 5666-273-01-A45  :  DY21001
     /*
     /&
    * $$ EOJ
```

---

(1) CORRECT 5666-273-01-A45 : DY21001
5666-273-01 is the component identifier and A45 the level indicator; DY21001 is the APAR number assigned to the fix.

(2) ALTER F0 9200B0F8:92F180F8
F0 is the (hexadecimal) displacement into the phase of the data that is to be altered. This example alters the operands of an MVI (92) instruction from 00B0F8 to F180F8.

(3) ALTER 6FA 00000000:4700C426
The statement inserts the specified BC (47) instruction into the expansion area.

(4) UNDO 5666-273-01-A45 : DY21001
Causes MSHP to restore the original instruction and to record the APAR (in the history file) as having been revoked.

Figure 7-4. Sample Job for Installing and Removing an APAR/Local Fix

Handling a Fix for a Module

The sample job shown in Figure 7-5 corrects module IJWCCD2 of the Clear Disk utility.

For the change to take effect, this and other modules have to be linked by the linkage editor, as phase CLRDK, into the system sublibrary IJSYSRS.SYSLIB. The sample job for installing the fix assumes that permanent assignments exist, assigning SYS001 through SYS004 to work files on disk.

The sample job shown in Figure 7-6 on page 7-12 removes this fix again.

```
    * $$ JOB JNM=CORRELO,CLASS=0
    // JOB    CORRECT RELOCATABLE MODULE
    // EXEC   MSHP
(1) CORRECT 5666-301-02-H07  :  DY19227
(2) AFFECTS MODULE=IJWCCD2 ESDID=1
(3) ALTER 1048 47F0F000:47F0F800
    RESOLVES 'CLEAR DISK ERROR'
    INVOLVES LINK = IJWCCD
    /*
    /&
    * $$ EOJ
```

```
(1) CORRECT 5666-301-02-H07  :  DY19227
    5666-301-07 is the component identifier and H07 the level indicator;
    DY19227 is the APAR number assigned to the fix.

(2) AFFECTS MODULE=IJWCCD2 ESDID=1
    ESDID=1 tells MSHP that the change applies to code identified by an
    ESD-ID of 1.

(3) ALTER 1048 47F0F000:47F0F800
    1048 is the hexadecimal displacement into the module.  The statement
    alters the operand of a BC instruction (47) from F000 to F800.
```

Figure 7-5. Sample Job for Installing a Fix for a Relocatable Module

```
    * $$ JOB JNM=UNDO,CLASS=0
    // JOB    UNDO FIX
    // EXEC   MSHP
(1) UNDO 5666-301-02-H07  :  DY19227
    /*
    /&
    * $$ EOJ
```

```
(1) UNDO 5666-301-02-H07  :  DY19227
    Causes MSHP to restore the original instruction and to record the APAR
    (in the history file) as having been revoked.
```

Figure 7-6. Sample Job for Removing a Fix from a Relocatable Module

Handling a Fix for an Edited Macro

Installing the Fix: The sample job shown in Figure 7-7 on page 7-13 corrects macros in a sublibrary named PRDSLA. This job assumes that permanent assignments have been made for the required system work files on disk (SYS001 through SYS004) and for SYSPCH (to a tape drive).

Since the REVOKABLE option is effective (by default), MSHP writes revoke jobs for the macros to SYSPCH before actually altering them.

Note: MSHP does not support the sequence-number restart function of ESERV. Therefore, to update a macro that requests statement-sequence numbers to be restarted, use ESERV and record the update in your system's history file. Use MSHP's ARCHIVE function for recording the update; the sample job given under "Archiving an Update in the History File" on page 7-16 shows how to do this.

```
     * $$ JOB JNM=CORSRCE,CLASS=0,DISP=H
     // JOB CORRECT SOURCE MACRO
     // EXEC MSHP
(1)  CORRECT 5666-301-02-H07 : DY17291
     AFFECTS MACROS=CDLOAD
(2)  VERIFY 007100
              AIF    (K'&PHASE LE 8).FOUR
(3)  INSERT 7100
            AGO   STOP
(5)  /$
(3)  INSERT 9100
     STOP    ANOP
(5)  /$
(1)  CORRECT 5666-301-02-H07 : DY18456
     AFFECTS MACROS=SETL
(4)  REPLACE : 300000+21
     RETURN ANOP
(5)  /$
     AFFECTS MACROS=SECHECK
(6)  DELETE 071000 : 072000
     /*
     /&
     * $$ EOJ
```

(1) CORRECT 5666-301-02-H07 : DY1nnnn
DY17291 and DY18456 are APAR numbers by which MSHP records the changes.

(2) VERIFY 007100
Tells MSHP to verify that the text on the next input line is identical to the data stored at the specified line (7100 in this example). This verification must be successful for the operation to continue.

(3) INSERT nnnn
Requests MSHP to insert the next input line immediately behind the specified line (7100 and 9100, respectively, in the example).

(4) REPLACE : 300000+21
The colon preceding the line number indicates that only one line of code is to be replaced. If a macro includes statements without line numbers, use the relative addressing technique shown here. The statement requests MSHP to replace the 21st line after line 300000 by the source code on the next line (RETURN ANOP in this example).

(5) /$
Indicates end of data input for the current MSHP function.

(6) DELETE 071000 : 072000
Requests MSHP to delete the statements from line 71000 to 72000.

Figure 7-7. Sample Job for Installing a Fix for an Edited Macro

Removing the Fix: To remove a fix implemented for an edited macro:

1. Add VSE/POWER JECL statements as shown in Figure 7-8 on page 7-14, if this is possible.

2. Start the revoke run with the MSHP produced job on the device assigned to SYSIN (if on tape) or SYSRDR and SYSIPT (if in card image).

The sample job given in Figure 7-8 revokes the corrections implemented with the sample job shown in Figure 7-7 on page 7-13.

```
* $$ JOB JNM=UNDO,CLASS=0,DISP=H

 ----------- MSHP generated jobs -----------

   // JOB DY17291  (UNDO) 05/06/84
   // EXEC MSHP
   UNDO 5666-301-02-H07 : DY17291
   DATA
    ...  ...  ...
   /$
   /*
   /&
   // JOB DY18456  (UNDO) 05/06/84
   // EXEC MSHP
   UNDO 5666-301-02-H07 : DY18456
   DATA
    ...  ...  ...
   /$
   /*
   /&

* $$ EOJ
```

Figure 7-8. Sample Job for Removing a Fix from an Edited Macro

Handling a Fix for an Unedited Macro

Figure 7-9 shows a sample job for implementing a fix for an unedited macro in the system source statement library.

MSHP implements the fix with the REVOKABLE option effective (by default). Therefore, before it alters this macro, MSHP writes a revoke job for the macro to the device assigned to SYSPCH.

The sample job assumes that permanent assignments have been made for the required system work files on disk (SYS001 through SYS004) and for SYSPCH (to a tape drive).

To remove a fix for an unedited macro, you proceed the same way as for revoking a fix for an edited macro, which is described under "Removing the Fix" on page 7-14.

```
    * $$ JOB JNM=CORSORCA,CLASS=0
    // JOB CORRECT UNEDITED SOURCE MACRO
(1) // PAUSE ASSGN SYSPCH TO TAPE FOR BACKOUT CREATION
    // EXEC MSHP
    CORRECT 5746-XX1-00-A50 :  PP73336
(2) AFFECTS MACROS=DLZCKOPT TYPE=A
    DELETE : 000400
    INSERT 450
           LCLB@B(9),@NGP
    @B(9)  SETB  (@PIO(@P))
    /$
    /*
    /&
    * $$ EOJ
```

```
(1) // PAUSE ASSGN SYSPCH TO TAPE FOR BACKOUT CREATION
    Gives the operator a chance to enter an ASSGN command for SYSPCH.

(2) AFFECTS MACROS=DLZCKOPT TYPE=A
    By specifying a macro type (TYPE=A in this example) you indicate that
    an unedited macro is to be corrected.

    A specification of TYPE=F indicates that a macro of the F-book format
    is to be corrected.  MSHP uses the NCP assembler rather than the DOS/VSE
    assembler in this case.
```

Figure 7-9. Sample Job for Installing a Fix for an Unedited Macro

Service-Run Complete Activities

When you have finished installing a PTF or an APAR/local fix, then:

1. Create a new backup tape for each of the sublibraries that have been changed. Submit a job similar to the one as indicated below, whichever applies:

    - If the system sublibrary IJSYSRS.SYSLIB was serviced – see "Step 10. Create a Backup Copy" on page 5-41.

    - If a licensed program was serviced – see Figure 6-2 on page 6-5.

2. Obtain a listing of the history file. Submit a job similar to the one shown under "Step 7. Obtain a Listing of the History File" on page 5-39.

History-File Related Service Activities

This section describes how to:

- Archive the update of a library member in the system history file.
- Handle a history-file-full situation.
- Record the (new) residence of a previously installed system component.

Archiving an Update in the History File

The need for this may arise if, because of unusual circumstances, you change a member in the system sublibrary without using MSHP for this purpose. At a later point in time, this change may have become an APAR fix which MSHP requires for the installation of a PTF. MSHP cannot install this PTF until you have recorded your change as the installation of the required APAR fix.

To record the change in your system's history file, run a job similar to the one shown below:

```
* $$ JOB JNM=ARCHUPD,CLASS=0
// JOB   ARCHIVE UPDATE
// EXEC  MSHP
ARCHIVE  5666-301-06-H07 APAR=DY12345
AFFECTS  MODULE=IJWCCDZ
/*
/&
* $$ EOJ
```

The above sample job records, in the system history file, a change by APAR DY12345; it records this change for module IJWCCDZ of the component 5666-301-06 on level H07.

Handling a History-File Full Situation

It may happen that your system's history file becomes full during installation of service changes; MSHP indicates this by a message. To recover, run a job as shown in Figure 7-10. Run the job in your system's background partition.

```
    * $$ JOB JNM=CRTEHST,CLASS=0
    // JOB CREATE NEW HISTORY FILE
(1) // ASSGN SYS002,cuu
    // EXEC  MSHP
    CREATE HISTORY AUXILIARY
(2) DEFINE HISTORY AUX EXT=19:54
    COPY HISTORY SYSTEM AUXILIARY
    /*
(3) ***** Making the New History File Accessible *****
    // OPTION STDLABEL=DELETE
    IJSYSHF
    /*
    // OPTION STDLABEL=ADD
    // DLBL IJSYSHF,'A5666301.SYSTEM.HISTORY',99/365
    // EXTENT ,vol-id,,,19,54
    /*
    /&
    * $$ EOJ
```

---

(1) // ASSGN SYS002,cuu
For cuu give the address of the disk drive whose volume contains the full system history file.

(2) DEFINE HISTORY AUX EXT=19:54
Defines an area, on the same volume, to where MSHP is to copy the contents of the full history file. Make sure you define this area (54 tracks in this example) significantly larger than the history file that has become full.

(3) ***** Making the New History File Accessible *****
The remaining statements up to (but not including) /& make the new history file accessible by MSHP. Omit these statements if, at your location, you do not have permanently stored label information for the system history file. You may then have to change cataloged procedures instead.

Figure 7-10. Sample Job for Handling a History-File Full Situation

Recording the Residence of a Licensed Program

If you move a licensed program (or a system component) to another sublibrary, this change is to be recorded in the system history file. To have MSHP record the changed residence, use a job similar to the one shown below:

```
* $$ JOB JNM=RECRES,CLASS=0
// JOB    RECORD NEW RESIDENCE
// EXEC   MSHP
RESIDENCE PRODUCT=301F93 -
          PRODUCTION=DSFSLIB.PROD -
          GENERATION=DSFSLIB.GENE
/*
/&
* $$ EOJ
```

The above sample job assumes that the Device Support Facilities program, an SCP prerequisite of VSE/Advanced Functions, has been moved to a different sublibrary.

This ends the discussion of service installation and service-related activities.

Appendix A. IBM Supplied Supervisors

Following is a table of generation options used for the supervisors supplied by IBM as part of VSE/Advanced Functions. The macros used for specifying these options must be coded in this sequence: SUPVR, FOPT, and IOTAB.

Supervisor $$A$SUP3 - For operation in 370 mode:

```
SUPVR   ID=3          FOPT   DASDSHR=YES          IOTAB   IODEV=254
        MICR=1419            FASTTR=YES                   NPGR=1000
        MODE=370             RPS=YES
        NPARTS=12            TRKHLD=12
                             TTIME=NO
                             USERID=VSE.370.SUP3
```

Supervisor $$A$SUPE - For operation in ECPS:VSE mode:

```
SUPVR   ID=E          FOPT   DASDSHR=YES          IOTAB   IODEV=254
        MICR=1419            FASTTR=YES                   NPGR=1000
        MODE=E               RPS=YES
        NPARTS=12            TRKHLD=12
                             TTIME=NO
                             USERID=VSE.4300.SUPE
```

Supervisor $$A$SUPV - For operation in VM mode:

```
SUPVR   ID=V          FOPT   DASDSHR=YES          IOTAB   IODEV=254
        MODE=VM              FASTTR=NO                    NPGR=1000
        NPARTS=12            RPS=YES
                             TRKHLD=12
                             TTIME=NO
                             USERID=VSE.VM.SUPV
```

Appendix B. The Supervisor Generation Macros

This appendix gives a description of the supervisor generation macros and shows how their operands are to be coded. The rules for using macros in assembler-language source programs apply also to these macros, except that the name field must remain blank. For a detailed discussion of these rules, refer to *OS/VS-DOS/VSE-VM/370, Assembler Language.*

Figure B-1 on page B-2 gives an overview of the available generation macros and their operands. It also shows which operands have been deleted since the availability of VSE/Advanced Functions 1.2. and why they have been deleted.

Coding Sequence: The macros and their operands have to be coded for the assembly of a supervisor in the following sequence: SUPVR, followed by FOPT, followed by IOTAB. This appendix discusses the macros and their operands in this sequence. The operands of a generation macro may be specified in any order.

Conventions for Format Descriptions: The conventions for showing the format of the generation-macro operands are:

- The default value for a specification, if applicable, is underscored. The assembler uses this default value if you omit the operand in question or specify an incorrect value.
- Braces ({ }) indicate that you must select one of the enclosed values. The values from which you can choose are separated by a vertical line (|).
- Uppercase characters and equal signs (=) must be coded as shown.
- Lowercase characters represent values which you must (or may) supply.

The Supervisor Generation Macros

Existing generation macros and their operand keywords:

SUPVR Macro	FOPT Macro	IOTAB Macro
ID	DASDSHR	IODEV
MICR	FASTTR	NPGR
MODE	RPS	
NPARTS	TRKHLD	
	TTIME	
	USERID	

Operands deleted since the availability of VSE/Advanced Functions 1.2 (if you code any of these operands, the assembler issues an MNOTE):

Macro	Operand Keyword	Reason for Deletion
SUPVR	NTASKS	The supervisor includes support for 208 or for 31 x p subtasks, whichever is the smaller; p is the number of partitions defined in NPARTS=n.
	TP	The telecommunication support is always included.
	VM	Specify MODE=VM instead.
FOPT	ASYNOC	Asynchronous operator communication is now standard support.
	CBF	Console buffering support is now standard. It is activated automatically if you submit an ADD command for a console printer during system start-up.
	DASDFP	Replaced by the DASDFP operand of the IPL command SYS.
	DOC	No longer required.
	ERRQ	No longer required.
	JA	Replaced by the JA operand of the IPL command SYS.
	JALIOCS	No longer required.
	LCONCAT	No longer required.

Figure B-1 (Part 1 of 2). The Supervisor Generation Macros and their Operands

Macro	Operand Keyword	Reason for Deletion
FOPT	MSECS	Replaced by the MSECS command.
	SEC	Replaced by the SEC operand of the IPL command SYS.
	SLD	No longer required.
	SYNCH	Dropped. There is no need for this support.
	XECB	No longer required.
IOTAB	BGPGR	Replaced by a new operand, NPGR, which causes a pool of programmer logical units to be reserved. Subsets of this pool may be assigned to partitions by way of the job control command NPGR.
	FnPGR	See BGPGR, above.
	CHANQ	Replaced by the CHANQ operand of the IPL command SYS.
	JIB	No longer required.
	NRES	No longer required.
	BUFSIZE	Replaced by the BUFSIZE operand of the IPL command SYS.

Figure B-1 (Part 2 of 2). The Supervisor Generation Macros and their Operands

SUPVR Generation Macro

The SUPVR (supervisor) generation macro defines support for functions such as hardware-operating mode for the system; it defines the scope of the multiprogramming support.

The operands for the macro are:

`ID={1|c}`

For c, specify an identifying character unique to your system if you plan to use two or more supervisors on the system. As an identifying character, use any alphameric character (from A through Z and from 1 through 9).

The assembler adds the specified character to the string $$A$SUP and uses this new string as the name of your supervisor. For example: if you specify ID=A, the assembler assigns $$A$SUPA to your supervisor.

`MICR={NO|1419|1419D}`

Specify MICR=1419 if the supervisor is to support magnetic ink character readers or optical reader/sorters. Specify 1419D to get support for a dual address adapter, which applies only to IBM 1419s or 1275s.

You need no MICR support for devices of type IBM 3886 and 3890.

`MODE={370|E|VM}`

The operand defines the operating mode in which your system is to operate. Specify:

`MODE=370`

For operation in 370 mode. Use this specification or omit the operand if your supervisor is to be used stand-alone (not under VM/SP) on a supported processor:

Of the System/370 architecture.
With ECPS:VSE and used in 370 mode.

You may load this supervisor as a virtual machine under VM/SP (to take advantage of the virtual addressability extension support, for example). If you do this, certain functions (page management, for instance) are performed by both VM/SP and the virtual machine under VM/SP.

`MODE=E`

For operation on a 43xx processor with ECPS:VSE, operating in ECPS:VSE mode.

`MODE=VM`

For use of the supervisor as a virtual machine under VM/SP. Specifying MODE=VM forces MICR=NO. You cannot run programs REAL under a supervisor with MODE=VM.

By specifying this operand, you get a supervisor that avoids a duplication of services provided by VM/SP (such as page-in and page-out control). In addition, your supervisor includes support for *VM/VTAM Communications Network Application.* Through this support, any virtual machine running under the same VM/SP can have access to the facilities of SNA.

You may use a MODE = VM supervisor under the VM/Extended Architecture (VM/XA) Migration Aid – an operating system that helps you migrate to MVS/XA, should this be desirable. However, in a VM/XA Migration Aid environment, some of the VSE functions may not be available. There is, for example, no support for access to the facilities of SNA in this environment.

NPARTS={<u>5</u>|n}
Specify the number (n) of partitions to be available for multiprogramming. The minimum value for n is 2, the maximum is 12.

FOPT Generation Macro

Use the FOPT (Optional Function) generation macro to specify optional functions that you want to be included in your supervisor.

The operands for the macro are:

DASDSHR={<u>NO</u>|YES}
Specify DASDSHR = YES if sharing of disks by two or more (up to 31) systems is desired. The support provides for an across-system locking mechanism needed to ensure data integrity when disk devices are shared by two or more systems.

FASTTR={<u>NO</u>|YES}
Specify FASTTR = YES to get support for fast CCW translation in 370 mode and fast-function support in ECPS:VSE mode. The operand does not apply if your SUPVR macro specifies MODE = VM.

RPS={<u>NO</u>|YES}
Specify RPS = YES to get support for rotational position sensing if your disk devices have the required hardware support. This hardware support is optional on an IBM 3340; it is not available on an IBM disk device of type 23xx.

RPS support permits a disk device to disconnect from the channel while the access arm of the device is being positioned.

TRKHLD={<u>NO</u>|n}
Specify the number (n) of hold requests that the system should allow to be active at any one time if the hold function is to be supported.

When processing a request with a hold for an update to a file on disk, the system prevents any other task using the track hold function from accessing the same data.

The maximum number of hold requests you can specify to be active at a time is 255. If your specification is invalid (non-numeric or greater than 255), the generated support allows 10 hold requests to be active concurrently.

Certain program products may require the track hold support to be included in your supervisor.

The hold function is not supported for the IBM 3540 diskette I/O unit.

`TTIME={NO|BG|Fn}`
: Specify task timer support by identifying the partition which is to own the task timer.

`USERID=identifier`
: Specify an identifier for a supervisor if you want this identifier to be printed as part of the IPL COMPLETE message.

You may specify an identifier of up to 16 characters in length. If you specify a longer identifier, the assembler truncates your specification to 16 characters at the right end. If you specify less than 16 characters, the assembler pads the identifier with blanks on the right.

Do not specify the identifier within quotes; imbedded blanks are not allowed.

IOTAB Generation Macro

The IOTAB (Input/Output Tables) generation macro defines the space needed by the system for device tables.

The operands for the macro are:

`IODEV={25|n}`
: Specify the number (n) of I/O devices attached to your system's processing unit. The maximum you can specify is 254, the minimum value is 4. Each unit that you define to the system by an IPL ADD command must be accounted for in your specification for the IODEV= operand.

`NPGR=n`
: Specify the number (n) of programmer logical units you need to have available for all of your partitions at any point in time. The highest and lowest values you can specify for n are:

```
Highest      255 x p
Lowest        30 x p (this value is also the default)

where p = The number of partitions as defined in NPARTS=n
          for SUPVR.
```

Appendix C. Machine Requirements and Device Support

Machine Requirements

One of the IBM processing units listed below:

- Any System/370 CPU[1] of Models 135 through 158 with a processor storage of at least 512K bytes. For acceptable performance, however, IBM recommends a processor storage of one megabyte or more.
- IBM 3031
- IBM 3033 operating as a single processor.
- IBM 4300 series processor.
- IBM 9370 series processor.
- Any IBM processing unit operating under control of VM/SP.

Supported I/O devices as follows:

- One IBM console.
- Either an IBM card reader or an IBM diskette I/O unit.
- One IBM printer.
- One IBM magnetic tape unit.
- One or more IBM disk drives with a total capacity of at least 120M bytes of disk storage. None of the disk drives may be an IBM 2311.

  For disk data to be shared, the following is required in addition:

[1] The processing unit must have: the clock comparator and the CPU timer (which are standard on all of the supported models, except Models 135 and 145); the floating point feature (which is standard on all supported models, except Models 135, 145, and 155).

- If the system operates stand-alone (not under VM/SP), this data must reside on an IBM disk drive whose control unit has a channel switch or a string switch.
- If the system operates under control of VM/SP, the disk drive with the lock file must be defined to VM/SP with virtual RESERVE/RELEASE support.

Processor and Device Support

Note: IBM may announce processors or I/O devices as supported by VSE/Advanced Functions after the publication's current edition has become available.

Supported devices are listed by type and number; model information is given only if this is of significance.

IBM-supplied devices that link to a channel-attached communication control unit are supported by VSE/Advanced Functions through this control unit. Some of the listed devices may be accessible for data transfer only if you write your own channel program.

Processors

The maximum processor storage size supported by VSE/SP is 16 megabytes.

As a virtual machine under VM/SP, VSE/Advanced Functions can run also on processors others than those listed below:

IBM 3031
IBM 3033 (See Note 1.)
IBM 3135 (See Notes 2 and 3.)
IBM 3138
IBM 3145 (See Notes 3 and 4.)
IBM 3148
IBM 3155-II (See Note 3.)
IBM 3158
IBM 4321
IBM 4331
IBM 4341
IBM 4361
IBM 4381
IBM 9373
IBM 9375
IBM 9377

Notes:

1. *Only for operation in single processor mode.*
2. *Use of VSE/Advanced Functions on this processor is not recommended.*
3. *Requires the optional floating point feature.*
4. *Requires the optional CPU timer and clock comparator.*

System Consoles

The interactive interface is designed to use a display console that has a screen size of 24 lines, 80 characters per line.

IBM 3210 (See Note 1 below.)
IBM 3215 (See Note 1 below.)
IBM 3277
IBM 3278, Models 2 through 5
IBM 3278, Model 2A
IBM 3279, Models 2A, 2B, 3A, 3B
IBM 3279, Models 2C (See Note 2 below.)

Notes:

1. *A printer-keyboard type system console.*
2. *Only the default colors (blue, green, and white) are supported.*

Local Display Stations

IBM 3178, Models C10 and C20
IBM 3277, All models with 24-line screens
IBM 3278, Models 2 - 5
IBM 3279, Models S2A, S2B, S3A, S3B, S3G, 2X, and 3X
IBM 3290 (supported as an IBM 3277)
IBM 8775, Models 1 and 2

Disk Devices

IBM 2311 (See Notes 1 and 2.)
IBM 2314/2319 (See Note 2.)
IBM 3330/3333
IBM 3340/3344 (See Note 4.)
IBM 3310 (See Notes 2 and 3.)
IBM 3350 (See Note 3.)
IBM 3370 (See Note 3.)
IBM 3375 (See Note 5.)
IBM 3380 (See Note 5 and 6.)
IBM 9332-400 (See Note 5.)
IBM 9335 (See Note 5.)

Notes:

1. *Supported only as I/O device for user data.*
2. *Not supported for sharing of data across computing systems.*
3. *ISAM is available only if the device is used in simulation or emulation mode, simulating or emulating a device supported by ISAM.*
4. *The IBM 3344 is supported as an IBM 3340; one head/disk assembly of the IBM 3344 simulates four 3348 Model 70 data modules on an IBM 3340.*
5. *ISAM is not available for the device.*
6. *Not supported are: the Dynamic Path Selection of the 3380 Model AA4 and the Speed Matching Buffer feature of the 3880 Storage Control.*

Magnetic Tape Units

IBM 2401
IBM 2415
IBM 2420
IBM 3410/3411
IBM 3420
IBM 3430
IBM 3480
IBM 8809
IBM 9347

Punched Card Devices

IBM 1442 Read/Punch, Model N1
IBM 1442 Punch, Model N2
IBM 2501 Reader
IBM 2520 Read/Punch, Model B1
IBM 2520 Punch, Models B2 and B3
IBM 2540 Read/Punch
IBM 2560 Multifunction Card Machine
IBM 3505 Reader
IBM 3525 Punch
IBM 5424 Multifunction Card Unit
IBM 5425 Multifunction Card Unit

System Printers

IBM 1403
IBM 1443
IBM 3200 (supported like an IBM 3800; see also Note below)
IBM 3203, Models 4 and 5
IBM 3211
IBM 3262, Models 1, 5, and 11
IBM 3289, Model 4
IBM 3800 (See Note 1 below.)
IBM 3820 (See Note 2 below.)
IBM 4245
IBM 4248 in PRT1 Mode (See Note 3 below.)
IBM 4248 in Native Mode (See Note 3 below.)

Notes:

1. *The logical input/output support must be ordered separately. For ordering details, contact your IBM representative.*

2. *The IBM 3820 printer requires ACF/VTAM to be installed on the system. Programs writing to this printer must be defined to ACF/VTAM as VTAM application programs. For more planning information, see the "IBM 3820 Page Printer and Advanced Function Printing Software: Introduction and Planning Guide".*

3. *The Stand-Alone Device Support Facilities program does not support the IBM 4248 printer.*

Terminal Printers

IBM 3213 (See Note below.)
IBM 3268
IBM 3284
IBM 3286
IBM 3287
IBM 3288
IBM 3289 all models except Model 4
IBM 5210
IBM 4224
IBM 4234
IBM 4245 Model D

Note: For use with the console of a 3158 processor.

Optical and Magnetic Character Reader Equipment

IBM 1255 (See Note 1.)
IBM 1259 (See Note 1.)
IBM 1270 (See Notes 1 and 2.)
IBM 1275 (See Notes 1 and 2.)
IBM 1287
IBM 1288
IBM 1419 (See Note 3.)
IBM 3881
IBM 3886
IBM 3890 (See Note 4.)

Notes:

1. *Supported like a 1419.*
2. *Not available in the United States.*
3. *The dual-address adapter is supported for the IBM 1419; but to use the adapter, you must restore the GENERATION library of VSE/Advanced Functions.*
4. *The logical input/output support must be ordered separately. To order this support, contact your IBM representative.*

Miscellaneous Input/Output Devices

IBM 3540 diskette I/O unit (see Note 1)
Feature 3401 Diskette Drive for IBM 4331 (see Note 2)
IBM 7443 Service Record File (see Note 3)
IBM 7770 Audio Response Unit

Notes:

1. *Supported as a unit-record input/output device.*
2. *Supported like an IBM 3540.*
3. *Used on the IBM 3031 service support console.*

Communication Control Units

Support is available for the full range of devices or subsystems that can be attached to an integrated communication adapter of an IBM 4300 processor or to one of the control units listed below. The support is available primarily with IBM ACF/VTAM and IBM CICS/VS; refer to the *General Information* manuals of these programs for a list of supported terminals. The control units are:

IBM 2701
IBM 2702
IBM 2703
IBM 3272
IBM 3274, Models 1A, 1B, and 1D
IBM 3704
IBM 3705
IBM 3720 (See Note below.)
IBM 3725
IBM 3791L – A local communication controller

Note: To use the IBM 3720, you require the additional IBM programs listed below (release level as shown or later):

ACF/System Support Programs, Version 3.2, Program No. 5666-322;

Either of the following:

- *ACF/Network Control Program, Version 4, Program No. 5668-854;*
- *ACF/Network Control Program, Version 4, Subset, Program No. 5668-754.*

Display Stations

IBM Personal Computer (See Note below.)
IBM 3178 (See Note below.)
IBM 3277
IBM 3278 (See Note below.)
IBM 3279 (See Note below.)
IBM 8775

Note: Supported as an IBM 3277.

Appendix D. Documentation for VSE/Advanced Functions

This appendix should help you and your staff in quickly finding the publication that answers your problem. You may copy this appendix for your own use.

The appendix lists, by major user tasks, the IBM publications that document VSE/Advanced Functions. Each of these user tasks (as seen by IBM) is briefly discussed. If a publication includes information for more than one major user task, it is listed for each applicable task.

The appendix includes also a Version-1 to Version-2 publications mapping chart. The chart shows which Version-2 publication you should look up for information that you used to find in a given Version-1 manual.

Publications by Major User Tasks

Administration-Task

The word "administration" is used here as a collective term for "resource definition" and "customization".

- *Resource definition* – The task of defining to the system the characteristics of required resources such as processor and virtual storage, disk space, libraries and library-search chains, or user profiles.

  The task includes working with some of the product's end-use programs (see also the section "End-Use Task" on page D-4).

- *Customization* – The task of enhancing or extending IBM provided support. For VSE/Advanced Functions, this is primarily the effort of coding, testing, and cataloging user-written exit routines.

Applicable publications:

> *IBM VSE/Advanced Functions System Management Guide*, SC33-6191
>
> Describes how to define required resources to VSE/Advanced Functions and how to make resources of your own, user-written programs, a part of your system. Discusses the coding of user-exit routines. Includes information about defining user and resource profiles for the protection of and controlled access to system resources.

You may have to consult cross-task publications as listed under "Cross-Task Publications" on page D-6.

Application-Programming Task

The task of designing, coding, compiling (assembly), and testing of application programs.

Applicable publications:

> *IBM VSE/Advanced Functions Data Management Concepts*, GC33-6192
>
> Describes the concepts of data handling under VSE/Advanced Functions. Summarizes the input/output functions provided by the available access methods. Briefly discusses the data organization of files that are created using those access methods; discusses the concepts and purpose of label processing.
>
> *IBM Guide to the DOS/VSE Assembler*, GC33-4024
>
> Provides guidance for using the assembler that is shipped by IBM as part of VSE/Advanced Functions.
>
> *IBM OS/VS-DOS/VSE-VM/370 Assembler Language*, GC33-4010
>
> Provides reference information needed for coding application programs in assembler language.
>
> *IBM VSE/Advanced Functions Application Programming: Macro User's Guide*, SC33-6196
>
> Contains guidance information for using IBM supplied macros in application programs written to run under VSE/Advanced Functions.
>
> *IBM VSE/Advanced Functions Application Programming: Macro Reference*, SC33-6197
>
> Contains reference information about IBM supplied macros.

You may have to consult cross-task publications as listed under "Cross-Task Publications" on page D-6.

Diagnosis Task

The task of isolating the cause of a problem and, if the cause is an IBM program, identifying this program. The task includes the user involvement in comparing the problem to similar known problems and in reporting a new problem.

Applicable publications:

IBM VSE/System Package Guide for Solving Problems, SC33-6311

Describes how to isolate a programming error either to IBM code or to user-written code. This description is given on a system level rather than on the limited VSE/Advanced Functions level.

For an error in IBM code, the publication tells how to collect data for reporting the problem to IBM and how to write and submit an APAR.

IBM VSE/Advanced Functions Service Aids, SC33-6195

A companion manual to *IBM VSE/System Package Guide for Solving Problems*; describes how to collect error information, using the service aids available with VSE/Advanced Functions.

IBM Environmental Recording Editing and Printing (EREP) Program, User's Guide and Reference, GC28-1378

Describes how to use the program for retrieving data recorded by the system in the system's recorder file.

The following publications, primarily intended for involved IBM service personnel, are available as optional material:

IBM VSE/Advanced Functions Diagnosis Reference:

Supervisor, LY33-9107
Error Recovery and Recording Transients, LY33-9108
Logical Transients and $IJBSxxx Phases, LY33-9109
Initial Program Load and Job Control, LY33-9110
Librarian, LY33-9111
Linkage Editor, LY33-9112
Maintain System History Program, LY33-9113
System Utilities, LY33-9114
Serviceability Aids, LY33-9115
LIOCS Volume 1 General Information and Imperative Macros, LY33-9116
LIOCS Volume 2 SAM, LY33-9117
LIOCS Volume 3 DAM and ISAM, LY33-9118
LIOCS Volume 4 SAM for DASD, LY33-9119

Also available, as optional material, are:

Device Support Facilities Logic, SY35-0030
DOS/VSE Assembler Logic, SY33-8567

In performing the diagnosis task, you may have to consult cross-task publications as listed under "Cross-Task Publications" on page D-6.

End-Use Task

End use means using an IBM program to achieve the purpose for which this program is provided. VSE/Advanced Functions includes utility programs that fall into this category. All but one of them are described as part of the documentation for the major tasks that make use of the utilities, *Administration* and *Operation*:

Utilities documented for the *administration task* (in the publication *IBM VSE/Advanced Functions System Management Guide*, SC33-6191) are:

Format Emulated Extent (INITEM)
Copy and Restore Diskette (CRDR)
Copy File (OBJMAINT)
Fast Copy Disk (FASTCOPY)
Initialize Tape (INTTP)

Utilities documented for the *operation task* (in the publication *IBM VSE/Advanced Functions Operation*, SC33-6194) are:

List Console Communication (LISTLOG)
Printer Train Cleaning (CLEANER)
Print Hard-Copy File (PRINTLOG)
List VTOC (LVTOC)

Applicable publication:

IBM Device Support Facilities, GC35-0033

Gives the information needed for disk initialization or disk surface analysis, and for alternate track (or block) assignment.

You may have to consult cross-task publications as listed under "Cross-Task Publications" on page D-6.

Installation Task

The effort of getting a program product ready for use and of installing IBM supplied changes to the product.

Applicable publications:

IBM VSE/Advanced Functions Planning and Installation, SC33-6193

The publication you are reading.

IBM VSE/Advanced Functions Maintain System History Program Reference, SC33-6199

Describes the control statements that can be used with the Maintain System History program, IBM's installation and service tool for VSE systems.

You may have to consult cross-task publications as listed under "Cross-Task Publications" on page D-6.

Operation Task

The task includes the starting and stopping of the system or of individual programs; it includes the monitoring of system operation and reacting to unusual situations. The task includes working with some of the product's end-use programs.

Applicable publications:

IBM VSE/Advanced Functions Operation, SC33-6194

The publication discusses the tasks normally performed by an operator; it provides the information needed to:

- Start up a VSE system.
- Control that system's operation from the console.
- Initiate device-oriented system services (such as loading a print buffer or performing a print-chain or print-train cleaning run).
- End (or shut down) the system.

IBM VSE/System Package Messages and Codes, SC33-6181

The publication explains the messages that are issued by the system. It includes instructions for actions to be taken at the console, if this is appropriate.

Planning Task

The task of making decisions about the options the program product offers. The decisions normally are written directions and procedures that are to be followed during installation of the product or later on during a customization effort.

Applicable publications:

IBM VSE/Advanced Functions Planning and Installation, SC33-6193

The publication you are reading.

IBM VSE/Advanced Functions System Management Guide, SC33-6191

You may have to refer to chapter 1 of the publication for information about concepts of system operation.

Cross-Task Publications

IBM VSE/Advanced Functions System Control Statements, SC33-6198

IBM considers this publication as a cross-task publication for all major user tasks discussed above, except planning and Operation. The publication describes the control statements and commands that are to be submitted for system operation. Not described are:

- Control statements and commands used with end-use programs. The statements and commands of those programs are described in the manuals for the applicable major user task: administration, end-use, and operation.
- Control statements used with MSHP, IBM's installation and service tool. These statements are described in the publication *IBM VSE/Advanced Functions Maintain System History Program Reference*, SC33-6199.

IBM VSE/System Package Messages and Codes. This manual is available under two different form-numbers:

SC33-6181	For VSE/Advanced Functions installed from a tape shipped by IBM **before** August 1, 1987;
SC33-6310	For VSE/Advanced Functions installed from a tape shipped by IBM **on or after** August 1, 1987.

IBM considers this as a cross-task publication for all major user tasks discussed in this appendix, except planning. The manual explains the messages and codes that are issued by program products which make up VSE/SP; VSE/Advanced Functions is always part of VSE/SP.

Version 1 to Version 2 Publications Mapping Chart

If not indicated otherwise, the publications listed in the chart below are part of the VSE/Advanced Functions library of publications. The chart can be helpful if, for example, you have a reference to a Version 1 publication but need the corresponding Version 2 publication.

Version-1 Publication	Corresponding Version-2 Publication
System Management Guide, SC33-6094	System Management Guide, SC33-6191
Data Management Concepts, GC24-5209	Data Management Concepts, GC33-6192
Planning and Installation, SC33-6096	Planning and Installation, SC33-6193
Operation, SC33-6097	Operation, SC33-6194
Diagnosis: Service Aids, SC33-6099	Service Aids, SC33-6195
Application Programming: Macro User's Guide, SC24-5210	Application Programming: Macro User's Guide, SC33-6196
Application Programming: Macro Reference, SC24-5211	Application Programming: Macro Reference, SC33-6197
System Control Statements, SC33-6095	System Control Statements, SC33-6198
Maintain System History Program, Reference, 33-6101	Maintain System History Program, Reference, SC33-6199
Reference Summary, GX33-9907	Reference Summary, GX33-9910
System Utilities, SC33-6100	
Administration-oriented: Format Emulated Extent (INITEM) Copy and Restore Diskette (CRDR) Copy File (OBJMAINT) Fast Copy Disk (FASTCOPY) Initialize Tape (INTTP)	System Management Guide, SC33-6191
Operation-Oriented: List Console Communication (LISTLOG) Printer Train Cleaning (CLEANER) Print Hard-Copy File (PRINTLOG) List VTOC (LVTOC)	Operation, SC33-6194

Figure D-1 (Part 1 of 2). Example for Preparing Scanner Input under VM/SP

Version-1 Publication	Corresponding Version-2 Publication
Messages and Codes, SC33-6098	VSE/System Package, Messages and Codes, SC33-6181 or SC33-6310
Diagnosis: Guide, SC33-6112	VSE/System Package Guide for Solving Problems, SC33-6311
Disk, Diskette, and Tape Labels, SC24-5213: Conceptual information	Data Management Concepts, GC33-6192
Label-layout information	Application Programming: Macro Reference, SC33-6197
Data Security Under the VSE System, GC33-6077: Planning Information	Planning and Installation, SC33-6193
Implementation Information	System Management Guide, SC33-6191

Figure D-1 (Part 2 of 2). Example for Preparing Scanner Input under VM/SP

Appendix E. Job Control Scanner Program

This appendix describes how to use the Job-Control Scanner program. This program, subsequently referred to as "scanner," assists in migrating from Version 1 of VSE/Advanced Functions to Version 2.

This appendix is the only documentation available for the scanner program, and includes examples and message explanations. If you wish, you may copy this appendix for use at your location.

Program Description

The scanner processes Version-1 job streams that have been copied onto a tape. The job streams on the tape may be one of the following:

One or more CMS files (applies to a virtual machine environment).
One or more VSE/ICCF libraries.
One or more VSE Version-1 source statement or procedure libraries.
An undefined source.

Based on its input, the scanner produces output as follows:

- List output on the SYSLST device

  The format of this output depends on the type of input. However, whichever input type is used, this output flags the job control statements that have to be changed or replaced; it flags, in addition, the statements which should be changed eventually.

- Tape output on the SYSPCH device

  The scanner makes this output available if the job streams on the input tape originated from a Version-1 procedure or source statement library.

Notes:

1. *The scanner processes the input line by line; it does not check for any kind of non-matching specifications.*

2. *Processing may fail for a member copied from a Version-1 library and containing more than 2000 statements.*

   If the scanner finds no migration hit for the first 2000 statements, the program's buffer space is exhausted. As a result, the scanner discontinues *processing the member and continues scanning the next member. The scanner informs you by a message.*

Operating Environment

The program needs a partition of:

- At least 240K bytes if job streams copied from a Version-1 procedure or source statement library are to be processed.
- Minimum size (128K bytes) if any other input is to be processed.

The possible types of input are discussed under "Preparing the Input" below.

You may, if this is desirable, use the scanner under Version 1 of VSE. To do this, proceed as follows:

1. Punch out the program's object code to a tape whose drive is assigned to SYSPCH. This code is cataloged in the system sublibrary by this member name and type:

   ```
   JCLSCAN.OBJ
   ```

   Use the librarian PUNCH function with the FORMAT=OLD specification for this punch-out run.

2. Link the punched out phase to a core image library of your Version-1 system.

Preparing the Input

The scanner reads its input from the tape mounted on the drive to which the logical unit SYS007 is assigned. It expects the input records to have a length of either 80 or 81 bytes. The scanner reads the tape from beginning until it finds a tapemark on the tape.

How to prepare the input is discussed by input type:

- CMS files (under VM/SP) – see "CMS Files" on page E-3.
- VSE Version-1 source statement or procedure libraries – see "Members from Procedure or Source Statement Libraries" on page E-3.
- VSE/ICCF libraries – see "VSE/ICCF Library Members" on page E-3.
- Undefined source – see "Undefined Source Input" on page E-3.

CMS Files

This input is referred to as *type C*.

You prepare the scanner-input tape by way of a MOVEFILE command. Figure E-1 on page E-4 shows a sequence of commands that creates a suitable input tape. The sequence assumes that the operator has mounted a scratch tape on the device known to VM/SP as TAP1.

Members from Procedure or Source Statement Libraries

This input is referred to as *type D*.

You prepare the scanner-input tape by using the PSERV program to copy procedures and the SSERV program to copy source books. To ensure that a tapemark follows the last member on the tape, issue an MTC WTM command or a CLOSE command.

For the scanner, a CATALP or CATALS statement marks the beginning of a member.

For details about setting up a PSERV or SSERV job, see the publication *IBM VSE/Advanced Functions System Management Guide*, SC33-6094.

VSE/ICCF Library Members

This input is referred to as *type I*.

You prepare the scanner-input tape by using the DTSUTIL program with a PUNCH PUNCTL command.

For the scanner, the statements ADD MEMBER and END OF MEMBER mark the beginning and end of a member.

For details about using the DTSUTIL program, see the publication *IBM VSE/Interactive Computing and Control Facility Installation and Operations Reference*, SC33-6067.

Undefined Source Input

This input is referred to as *type U*.

The scanner expects a tape with a file containing sequences of job control statements. Make sure to write a tapemark at the end of the file.

```
(1)    CH RDR CL J HOLD
(2)    SP PU * CL J
(3)    SP RDR CL J CONT
         ... ... ...       ┐ Punch the members (CMS
         ... ... ...       │  files) that are to be pro-
         ... ... ...       ┘ cessed by the scanner
(4)    FI INMOVE READER
(4)    FI OUTMOVE TAP1 (LRECL 80 BLOCK 80 RECFM F DEN 6250
(5)    MOVEFILE
(6)    TAPE WTM
(6)    TAPE REW
(7)    SP PU OFF CL A
(7)    SP RDR NOCONT CL A
```

---

(1) CH RDR CL J HOLD
Causes any files currently in your reader to be placed in the hold status. Do this as a precaution to avoid loss of data.

(2) SP PU * CL J
Causes punch files to be spooled into the own reader, class J.

(3) SP RDR CL J CONT
Ensures that all CMS files that are to be punched to the reader will be continuous, that is, all contained in one file.

(4) FI INMOVE READER
FI OUTMOVE TAP1 (LRECL 80 BLOCK 80 RECFM F DEN 6250
They define the source and the target for the write-to-tape operation.

(5) MOVEFILE
Initiates writing the spooled file to tape.

(6) TAPE xxx
TAPE WTM causes a tapemark to be written at the end of the last member (CMS file) that is copied onto tape.
TAPE REW causes the tape to be rewound.

(7) SP PU OFF CL A
SP RDR NOCONT CL A
They restore the original spool controls see references (2) and (3) above.

Figure E-1. Example for Preparing Scanner Input under VM/SP

Description of Output

Output to SYSLST

The scanner always produces output to SYSLST. This output is essentially the same for input types C, D, and I. The scanner structures this output by members and prints only the affected statements. Each one of these statements is followed by an action-flag message. See Figure E-2 for an example of SYSLST output for input of type D.

The SYSLST output for input of type U is a list of *all* statements in the same sequence as they are stored in the input file. The scanner prints an action-flag message behind any statement that is affected by migration to Version 2. The output differs from another scanner SYSLST output primarily by the absence of a structure by members. See Figure E-3 on page E-6 for an example of output for input of type U.

```
// JOB   SCANTST
// ASSGN SYS007,280
// EXEC  JCLSCAN,PARM='D,TRM6,155'
/&
```

Output on SYSLST:

```
LIBRARY 155     MEMBER STRTPR2
// ASSGN SYSSLB,300
*** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED **************
// ASSGN SYSCLB,310
*** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED **************
// LIBDEF CL,FROM=IJSYSRS
*** STATEMENT WILL BE MIGRATED BY THE SYSTEM, BUT SHOULD BE MODIFIED ***
// LIBLIST CL
*** STATEMENT WILL BE MIGRATED BY THE SYSTEM, BUT SHOULD BE MODIFIED ***
// EXEC BACKUP
*** PROGRAM NO LONGER SUPPORTED. REPLACE ALSO SYSIPT DATA **************
// EXEC MAINT
*** CHECK SYSIPT DATA **************************************************
// EXEC MSHP
*** CHECK SYSIPT DATA **************************************************
// OPTION LINK
*** CHECK IF MULTIPHASE OR OVERLAY PROGRAM IS BEING LINKED *************
END OF MEMBER STRTPR2
     E N D   O F   S C A N   P R O G R A M
```

Figure E-2. Example of a Scanner SYSLST Output for Input of Type D

```
  // JOB   SCANTST
  // ASSGN SYS007,280
  // EXEC  JCLSCAN,PARM='U'
  /&
```

Output on SYSLST:

```
// JOB STRTPR2
// ASSGN SYSSLB,300
*** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED **************
// ASSGN SYSCLB,310
*** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED **************
// LIBDEF CL,FROM=IJSYSRS
*** STATEMENT WILL BE MIGRATED BY THE SYSTEM, BUT SHOULD BE MODIFIED ***
// LIBLIST CL
*** STATEMENT WILL BE MIGRATED BY THE SYSTEM, BUT SHOULD BE MODIFIED ***
DVCDN 180
// ASSGN SYS005,300
// ASSGN SYS006,280
// ASSGN SYS007,300
// ASSGN SYS008,300
// ASSGN SYS009,300
// EXEC BACKUP
*** PROGRAM NO LONGER SUPPORTED. REPLACE ALSO SYSIPT DATA **************
// EXEC MAINT
*** CHECK SYSIPT DATA **************************************************
   DELETE MODUL1
// EXEC MSHP
*** CHECK SYSIPT DATA **************************************************
 RETRACE
// OPTION LINK
*** CHECK IF MULTIPHASE OR OVERLAY PROGRAM IS BEING LINKED *************
/*
/&
     E N D   O F   L I S T I N G
     E N D   O F   S C A N   P R O G R A M
```

Figure E-3. Example of a Scanner SYSLST Output for Input of Type U

Output to SYSPCH

The scanner provides output to tape (assigned to SYSPCH) if the input is of type D. This output is suitable for being loaded into a VSE/ICCF library using the VSE/ICCF DTSUTIL program. You can then make changes as required using the editor of VSE/ICCF. The output of the scanner includes for this purpose:

- An ADD MEMBER statement before each member.
- An END OF MEMBER statement behind each member.

These statements are needed by the VSE/ICCF DTSUTIL program in order to load the scanner output.

- The applicable action-flag message behind each statement that is affected by migration to Version 2. This is the same as for a scanner output to SYSLST.

CATALP and CATALS statements contained in the scanner input are included unchanged in this output. Following your update effort, you can submit the job(s) for recataloging in one of your system's VSE libraries.

For ease of job submission to VSE/POWER, the scanner has inserted job delimiting JECL place holders.

Figure E-4 on page E-8 shows an example of records in a SYSPCH output of the scanner.

Note: *If you do not want the scanner to provide output on the assigned SYSPCH tape, run the job with the statement*

```
// ASSGN SYSPCH,IGN
// JOB   SCANTST
// ASSGN SYS007,280
// ASSGN SYSPCH,281
// EXEC  JCLSCAN,PARM='D,TRM6,155'
/&
```

Output on SYSPCH:

```
 ADD MEMBER 155  STRTPR2    TRM6
 ..$$ JOB JNM=...
 // JOB ...
 // EXEC LIBR
 ACCESS SUBLIB=...
   CATALP STRTPR2
 // JOB STRTPR2
 // ASSGN SYSSLB,300
 *** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED **************
 // ASSGN SYSCLB,310
 *** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED **************
 // LIBDEF CL,FROM=IJSYSRS
 *** STATEMENT WILL BE MIGRATED BY THE SYSTEM, BUT SHOULD BE MODIFIED ***
 // LIBLIST CL
 *** STATEMENT WILL BE MIGRATED BY THE SYSTEM, BUT SHOULD BE MODIFIED ***
 DVCDN 180
 // ASSGN SYS005,300
 // ASSGN SYS006,280
 // ASSGN SYS007,300
 // ASSGN SYS008,300
 // ASSGN SYS009,300
 // EXEC BACKUP
 *** PROGRAM NO LONGER SUPPORTED. REPLACE ALSO SYSIPT DATA **************
 // EXEC MAINT
 *** CHECK SYSIPT DATA **************************************************
    DELETE MODUL1
 // EXEC MSHP
 *** CHECK SYSIPT DATA **************************************************
  RETRACE
 // OPTION LINK
 *** CHECK IF MULTIPHASE OR OVERLAY PROGRAM IS BEING LINKED *************
 /*
 /&
 ../*
 ../&
 ..$$ EOJ
END OF MEMBER
```

Figure E-4. Example of Records in a Scanner SYSPCH Output

Starting a Scan Run

To do this, submit a job similar to the example shown and explained in Figure E-5. The example assumes that permanent assignments exist for: SYSRDR, SYSIPT, and SYSLST.

```
(1)  * $$ JOB JNM=SCNJOB1,CLASS=9,DISP=D
     // JOB   SCANJCL
(2)  // ASSGN SYS007,280
(3)  // ASSGN SYSPCH,281
(4)  // EXEC  JCLSCAN,PARM='D,TRM6,155'
     /*
     /&
(1)  * $$ EOJ
```

The numbers within parentheses refer to explanations given following this sample job.

Figure E-5. Sample Job for Starting a Scan Run

Explanations for Figure E-5:

(1) * $$ ...
VSE/POWER statements. The system ignores these statements if you have the scanner run executed outside VSE/POWER control.

(2) // ASSGN SYS007,280
Assign SYS007 to the drive on which your operator is to mount the input tape.

(3) // ASSGN SYSPCH,281
Instruct your operator to mount a scratch tape on the drive that you specify. *However*, if you do not need the scanner's SYSPCH output, supply instead:

```
// ASSGN SYSPCH,IGN
```

and, as scanner control data, a dummy user identifier and a dummy VSE/ICCF library number. Example:

```
D,XZYZ,999
```

(4) // EXEC JCLSCAN,PARM='D,TRM6,155'
Supplies, by way of the PARM operand, the control data needed by the scanner.

If you use the scanner under Version 1 of VSE/Advanced Functions, this control data is to be submitted as SYSIPT data. In this case, your coding is like this:

```
// EXEC  JCLSCAN
D,TRM6,155
```

The scanner requires its control data in a format as follows:

{t|D,user-id,library-no}

where:

t = One of the following:

C A tape with input of type C (containing CMS files) is to be processed.

I A tape with input of type I (containing members of one or more VSE/ICCF libraries) is to be processed.

U A tape with input of type U (containing one or more job streams from an undefined source) is to be processed.

D A tape with input of type D (containing members from a Version-1 procedure or source statement library) is to be processed. If you specify this input type, you must specify also a user identifier and a library number as described below.

user-id
For user-id, specify an identifier of one to four characters. The scanner inserts this identifier, TRM6 in the example, into the ADD MEMBER statements of the SYSPCH output for the run.

library-no
For library-no, supply the three-digit number of the VSE/ICCF library into which the scanner's SYSPCH output is to be loaded for later editing.

Messages

The scanner issues two kinds of messages:

- Action-flag messages

  A message of this type follows an individual job control statement of the job stream(s) being scanned. It indicates the action to be taken in order to safely migrate the affected job stream to Version 2.

- Error messages

  A message of this type indicates an error condition found by the scanner.

All messages are directed to the device assigned to SYSLST.

Action-Flag Messages

***** STATEMENT WILL BE MIGRATED BY THE SYSTEM BUT SHOULD BE MODIFIED**

Explanation: A statement so flagged is to be changed if no library-migration table exists. For details, refer to "Conversion of Existing Job Streams" on page 3-18. IBM recommends that you change the statement at your earliest convenience.

***** STATEMENT WILL BE MIGRATED BY THE SYSTEM BUT SHOULD BE REPLACED**

Explanation: A statement so flagged is to be changed if it requests a function not available with the Version-2 librarian. For details, refer to "Processing of Existing Library Service Jobs" on page 3-22. IBM recommends that you replace the statement at your earliest convenience.

***** CHECK IF MULTIPHASE OR OVERLAY PROGRAM IS BEING LINKED**

Explanation: A // OPTION LINK statement was found in the job stream, which indicates immediate program execution. Version 2 of VSE no longer supports the link-and-go function for a multiphase or an overlay program.

Bypass: Link your phases permanently (// OPTION CATAL) into a sublibrary available for program testing.

***** STATEMENT IS NO LONGER SUPPORTED AND MUST BE REPLACED**

Explanation: Self-explanatory.

***** NO INCOMPATIBLE STATEMENTS**

Explanation: In the scanned member, the scanner found no statements that should be changed or replaced.

***** FOLLOWING MEMBER DOES NOT CONTAIN INCOMPATIBLE STATEMENTS**

Explanation: In the member named after this message, the scanner found no statements that should be changed or replaced.

***** PROGRAM NO LONGER SUPPORTED. REPLACE ALSO SYSIPT DATA**

Explanation The flagged statement is to be replaced, including the program control statements that follow the flagged statement. For a bypass, refer to "Migrators from Version 1 of VSE/Advanced Functions" on page 3-1; the section indicates possible bypasses for programs no longer supported.

***** CHECK SYSIPT DATA**

Explanation: The SYSIPT data for the involved // EXEC request may have to be changed. For details, refer to "Processing of Existing Library Service Jobs" on page 3-22.

Error Messages

1G01I GETVIS REQUEST UNSUCCESSFUL. PROGRAM TERMINATED

Explanation: The scanner is to process input of type D, but the available GETVIS storage is less than 160K bytes.

System Action: The program terminates.

Programmer Response: Rerun the job in a partition of 240K bytes or larger and with SIZE=80K specified in your // EXEC statement.

1G02I INVALID PARAMETER SPECIFIED. PROGRAM TERMINATED

Explanation: The // EXEC statement invoking the scanner is invalid for one of the following reasons:

- Wrong length of the specified parameter value(s).
- A delimiter other than a comma in the parameter specification.
- An invalid input type.

System Action: The program terminates.

Programmer Response: Correct the control information in the PARM operand of the // EXEC statement for the scanner and rerun the job.

1G03I **INVALID LIBRARY NUMBER. PROGRAM TERMINATED**

Explanation: The // EXEC statement for the scanner specifies input type D, but an alphameric string was given as library number in the PARM operand.

System Action: The program terminates.

Programmer Response: Correct the control information in the PARM operand of the // EXEC statement for the scanner and rerun the job.

1G04I **NO VALID ADD STATEMENT. PROGRAM TERMINATED**

Explanation: The scanner was called to process input of type I. However, in the input stream, there was no valid ADD MEMBER control statement at the place where the scanner expected to find such a statement. Either the scanner input was incorrectly copied to tape or a wrong tape was mounted.

System Action: The program terminates.

Programmer Response: Ensure that the scanner input is copied to tape by using the DTSUTIL program and specifying the PUNCTL operand in the PUNCH command.

1G05I **NO VALID CATALS OR CATALP STATEMENT. PROGRAM TERMINATED**

Explanation: The scanner is processing input of type D. However, the scanner either:

- Did not find a CATALS or CATALP statement, or
- Found such a statement without a member name or with a member name which is too long.

Possibly, the scanner input was incorrectly copied to tape.

System Action: The program terminates.

Programmer Response: Ensure that the scanner input is copied to tape by using the SSERV program for a source statement library and the PSERV program for a procedure library.

1G06I **STORAGE EXHAUSTED. SCAN FOR THIS MEMBER TERMINATED**

Explanation: The scanner is processing input of type D. The currently processed member contains more than 2000 statements, and the scanner found no migration hit for the first 2000 statements.

System Action: The scanner skips the remainder of the currently processed member and continues processing the next member.

Programmer Response: Either of the following:

- Manually check the 2001st and subsequent statements of the member up to the member's end.
- Insert, before the 2000th statement, a statement that causes a migration hit; then resubmit the job.

1G07I **INPUT FILE IS EMPTY**

Explanation: Most likely, a wrong tape was mounted.

System Action: The program terminates.

Programmer Response: Resubmit the job after having ensured that the input tape contains the correct input.

1G08I **NO CONTROL INFORMATION RECEIVED. SPECIFY PARAMETER OR SYSIPT DATA**

Explanation: Self-explanatory.

System Action: The program terminates.

Programmer Response: Resubmit the job with control information supplied either as a specification in the PARM operand of the // EXEC statement or as SYSIPT data.

1G09I **INVALID RECORD LENGTH OF INPUT FILE**

Explanation: The file on the input tape contains records whose length is neither 80 nor 81 bytes. Possibly, a wrong tape was mounted.

System Action: The program terminates.

Programmer Response: Resubmit the job after having ensured that the input tape contains the correct input.

1G10I **NO VALID :READ STATEMENT. PROGRAM TERMINATED**

Explanation: The scanner was called to process input of type C. However, in the input stream, there was no valid :READ statement at the place where the scanner expected to find such a statement. Possibly, a wrong tape was mounted.

System Action: The program terminates.

Programmer Response: Resubmit the job after having ensured that the input tape contains the correct input.

Note: IBM 3380 (see Note 5 on page C-3 and 6 on page C-3)

List of Abbreviations

ACF/VTAM. Advanced Communications Function/Virtual Telecommunication Access Method

APAR. Authorized Program Analysis Report

ASI. Automated system initialization

BG. Background partition

bpi. Bits per inch

CCB. Command control block

CCW. Channel command word

CICS. Customer Information Control System

CKD. Count-key-data

cuu. A variable. To mean: channel and unit number

DAM. Direct access method

DASD. Direct access storage device

DOS. Disk Operating System

DOS/VS. Disk Operating System/Virtual Storage

ECPS:VSE. Extended Control Program Support: VSE

FBA. Fixed block architecture

Fn. Foreground partition (where n = number of partition in hexadecimal).

I/O. Input/output

ICCF. Interactive Communication Control Facility

IPL. Initial program load

IOCS. Input/output control system

ISAM. Indexed sequential access method

JCL. Job control language

K. Kilo – to mean: 1024 (bytes)

LIOCS. Logical IOCS

M. Mega – to mean: 1024K (bytes)

MSHP. Maintain System History program

MVS/XA. MVS/Extended Architecture

N/A. Not applicable

NCP. Network control program

OCCF. Operator Communication Control Facility

PTF. Program temporary fix

RPS. Rotational position sensing

SA (dump). Stand-alone dump

SAM. Sequential access method

SCP. System control programs (as opposed to licensed programs)

SDL. System directory list

SNA. System network architecture

SVA. Shared virtual area

SYSRES. System residence

VAE. Virtual addressability extension

VM. Virtual machine

VM/SP. VM/System Product

volid. A variable. To mean: volume serial number

VSAM. Virtual Storage Access Method

Abbreviations

VSE.	Virtual storage extended
VSE/SP.	VSE/System Package
VSE/ICCF.	VSE/Interactive Communication Control Facility
VSE/OCCF.	VSE/Operator Communication Control Facility
VSE/VSAM.	VSE/Virtual Storage Access Method
VTAM.	Virtual Telecommunication Access Method

Index

IBM Virtual Storage Extended/Advanced Functions
Planning and Installation
Order No. SC33-6193-02

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*
Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: ____________

____

____

____

What is your occupation? ____________

Number of latest Newsletter associated with this publication: ____________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

SC33-6193-02

Reader's Comment Form

Fold and Tape **Please Do Not Staple** Fold and Tape

No Postage necessary if mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

International Business Machines Corp.
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold and Tape **Please Do Not Staple** Fold and Tape

IBM®

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

IBM Virtual Storage Extended/Advanced Functions
Planning and Installation
Order No. SC33-6193-02

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*
Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: ____________________

____

____

____

What is your occupation? ____________________

Number of latest Newsletter associated with this publication: ____________________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Cut or Fold Along Line

SC33-6193-02

Reader's Comment Form

Fold and Tape **Please Do Not Staple** Fold and Tape

No Postage necessary if mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

International Business Machines Corp.
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold and Tape **Please Do Not Staple** Fold and Tape

IBM®

IBM Virtual Storage Extended/Advanced Functions
Planning and Installation
Order No. SC33-6193-02

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*
Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: ______________________

____

____

____

What is your occupation? ______________________

Number of latest Newsletter associated with this publication: ______________________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

SC33-6193-02

Reader's Comment Form

Fold and Tape | **Please Do Not Staple** | Fold and Tape

No Postage necessary if mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

International Business Machines Corp.
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold and Tape | **Please Do Not Staple** | Fold and Tape

IBM®

SC33-6193-02

IBM Virtual Storage Extended/
Planning and Installation
(File No. S370/4300-34)
Printed in U.S.A.