GC28-0673-5
File No. S370-37

Systems

OS/VS System Modification Program (SMP) System Programmer's Guide

IBM

Sixth Edition (October, 1978)

This is a major revision of GC28-0673-4 and Technical Newsletter GN28-2918. See the Summary of Amendments for a list of changes that have been made to this manual. A vertical line to the left of the text or illustration indicates a technical change made in this edition.

This edition with Technical Newsletter GN28-2992 applies to the System Modification Program (SMP) for Release 6.7 of OS/VS1 and Release 3.8 of OS/VS2 MVS and to all subseqent releases of OS/VS unless otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM **System/370 Bibliography**, GC20-0001, for the editions that are applicable and current.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. 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.

Preface

The System Modification Program (SMP) is a service aid that is used to install system modifications (SYSMODs) on an OS/VS1 or OS/VS2 MVS operating system and associated distribution libraries.

This publication describes how to use SMP to install or remove system modifications, how to create and initialize SMP data sets, and how to correct and prevent installation errors. It also includes descriptions of the different types of system modifications, descriptions of SMP processing, and examples of system modifications.

This publication is intended for IBM personnel who create system modifications and system programmers who install and create system modifications. The reader should be experienced in using or maintaining VS operating systems.

Each page of SMP output includes an indicator denoting the SMP level being executed. The indicator is in the form xx.yy where:

> xx is the release level of SMP, increased by 1 for each subsequent release.
>
> yy is the PTF level within the release level, increased by 1 for each SMP PTF released that applies to the xx SMP level.
>
> This publication corresponds to level 04.10.

Note that this publication does not obsolete the <u>OS/VS System Modification Program (SMP) System Programmer's Guide</u>, GT28-0673-4 for SMP Release 3 users.

The publication contains ten chapters, three appendices and a glossary as follows:

Chapter 1: Introduction - provides an overview of SMP.

Chapter 2: System Modifications - provides an explanation of the types of system modifications supported by SMP. The information provided in this chapter is necessary to use and understand the SMP modification control statements described in Chapter 8.

Chapter 3: SMP Processing - provides a detailed explanation of the processing that takes place during RECEIVE, REJECT, APPLY, RESTORE, ACCEPT, JCLIN and UCLIN processing.

Chapter 4: SMP Installation and Use - provides the information necessary to install and execute SMP. Examples of commonly used SMP procedures are included.

Chapter 5: SMP Diagnostic Techniques - helps the reader to prevent, recognize, and recover from error conditions that might occur during SMP processing.

Chapter 6: SMP Reports - explains the reports that might be produced during SMP processing.

Chapter 7: SMP Control Statements - provides detailed descriptions of the SMP control statements, in alphabetic order by control statement.

Chapter 8: SMP Modification Control Statements - provides detailed descriptions of the modification control statements, in alphabetic order by modification control statement.

Chapter 9: SMP Data Sets - describes the data sets used by SMP.

Chapter 10: SMP Messages - lists the SMP messages, in alphanumeric order.

Appendix A: Rules for Coding SMP Statements - provides rules for coding SMP control statements and modification control statements.

Appendix B: Syntax Notation Conventions - provides the syntax notation conventions used to define SMP control statements and modification control statements.

Appendix C: PTF Compatibility Feature - describes a feature that enables SMP to process PTFs that are defined using syntax and rules on the modification control statements supported by previous versions of SMP.

Glossary: provides definitions of SMP terms and abbreviations.

Associated Publications

- OS/VS System Modification Program (SMP) Logic, SY28-0685

The following publications might be required when you use SMP:

- OS/VS Linkage Editor and Loader, GC26-3813
- OS/VS and DOS/VS Assembler Language, GC33-4010
- OS/VS MVS Utilities, GC26-3902
- OS/VS1 Utilities, GC26-3901
- OS/VS1 JCL Reference, GC24-5099
- OS/VS2 MVS JCL, GC28-0692
- OS/VS1 Service Aids, GC28-0665
- OS/VS2 System Programming Library: Service Aids, GC28-0674
- OS/VS1 Data Management Services Guide, GC26-3874
- OS/VS2 System Programming Library: Data Management, GC26-3830
- OS/VS2 DADSM Logic, SY26-3858
- OS/VS1 DADSM Logic, SY26-3837
- OS/VS2 MVS Data Management Services Guide, GC26-3875

During the installation of SMP, the following publications can provide needed data:

- OS/VS1 System Generation, GC26-3791
- OS/VS2 System Programming Library: System Generation Reference, GC26-3792

- OS/VS2 System Programming Library: Initialization and Tuning Guide, GC28-0681
- OS/VS2 MVS Release 3.8 Guide, GC28-0707

Table of Contents

Figures

Summary of Amendments for GC28-0673-5

This document describes an enhanced System Modification Program (SMP). It allows IBM developers and users to better control the function and service level of their system. It also provides enhanced planning functions and automatic maintenance facilities.

Incompatibilities

- The SU process as supported by the INSTALL macro and Release 3 of SMP is not supported by this enhancement.
- SMP control information applicable to Release 3 of SMP requires modification to be applicable to Release 4.

Support of Function Installation

- SMP recognizes when a function is being installed. Facilities are provided for the support of a hierarchy of function.
- SMP provides facilities for the management of function. Specifically, SMP allows the element content of a function package to change and parts or all of the function to be replaced in a system.
- The service level of the system is maintained whenever a function is installed. SMP ensures that the service level of other functions and of the installed function is at the proper level.
- A facility is provided that allows SMP to ensure that, upon function installation, the system is automatically brought up to the correct service level with respect to the functions installed.

- SMP allows the installation of a function even if the function had been previously installed, and ensures that the proper service level is maintained.

User Processes

- You are allowed to receive function and service for that function without requiring the application of the function and service to his system. This provides an enhanced pre-sysgen planning capability that does not interfere with the ability to service existing systems.

- The user is able to do dry runs of APPLY and ACCEPT processing and thereby determine the effects of applying and accepting new service or function.

- SMP allows function and service to be received regardless of the state of the control data set (CDS). For installations with more than one version of an operating system, this allows one RECEIVE operation to be valid for all system versions.

- SMP allows a user to specify a permanent parameter list in the SMP data sets to override some default operations.

- SMP controls the preparation of the user distribution libraries by the use of RECEIVE and ACCEPT NOAPPLY.

- A facility is provided that allows the user to have SMP merge a user modification into a source module or macro.

- The JCLIN information, which includes the description of load module target system structure, is automatically maintained by SMP. You have the ability to package JCL input data inline with the associated modification.

Service Installation

- SMP Release 4 only supports a system modification (SYSMOD) construction that uses FMID operands.

- SMP allows the user to receive all potentially applicable service into the PTS. SMP then automatically groups together related service whenever the environment of the target system or distribution libraries changes.

- SMP allows multiple modifications to an element during APPLY processing. This allows you to apply as many modifications as desired to a single element without accepting any modifications into your distribution libraries.

- SMP allows multiple replacements and updates to elements to be processed concurrently during APPLY and ACCEPT processing.

- SMP distinguishes between APAR fixes, PTFs, user modifications, and function modifications.

- SMP provides facilities to merge source updates to the same source module and macro updates to the same macro at APPLY/ACCEPT time.

Miscellaneous

- The name of a modification may be any seven character alphanumeric string, the first character of which should be alphabetic. SMP is insensitive to the content of the system modification name, but the alphabetic first character is required by some system utilities used by SMP.

- Each element has associated with it a function modification identifier (FMID). This identifier represents the function package to which this element belongs. Future modifications to an element must specify a relationship to the function using the FMID modification identifier.

- A new keyword (VERSION) is provided to allow a system modification package to indicate superiority to other functions.

Reliability, Availability and Serviceability (RAS)

SMP has a positive impact on system RAS because it automates the function and service installation process and thereby improves your ability to keep your system at the highest IBM-provided service level. This has the potential of reducing reported system failures and thereby improving the serviceability of IBM products. This improvement comes from two factors:

- First, by integrating function and service installation with the SMP process, SMP removes user dependency on the INSTALL macro and its level.

- Secondly, SMP Release 4 permits a significantly improved service strategy for both the user and IBM. Release 4 provides for the staging of any function and its service into the SMP PTF Temporary Store Data Set (SMPPTS). SMP allows any valid combination of modifications to be taken from this data set and applied to the target system libraries and accepted into the distribution libraries (DLIBs).

These two functions allow the user to maintain in the PTS an accumulation of all potentially applicable IBM function and service. The user may then, either periodically or because of a system failure, APPLY all or selected applicable service to the system libraries.

The user is also able to plan function installation and ensure that all applicable service to the function is being accumulated on the PTS in preparation for application.

Data Sets

New data sets have been added to SMP. The following are the ddnames used by SMP:

- SMPACRQ
- SMPCRQ
- SMPLIST
- SMPRPT
- SMPSCDS
- SMPTLIB
- SMPWRK1
- SMPWRK2

• SMPWRK3	SYSUT4
• SMPWRK4	SMPADDIN
• SMPWRK5	SMPPUNCH

Data sets have been deleted from SMP. The following are the ddnames of the deleted data sets:

- SMPREPIN
- SMPUCS

SMP Control Statements

The following SMP control statements have been deleted:

- CONVERT - No conversion is required to SMP Release 4 data sets.
- PRINT -You can print elements using IEBPTPCH or a comparable utility.
- PUNCH -You can punch elements using IEBPTPCH or a comparable utility.
- PTPCH -You can print or punch elements using IEBPTPCH or a comparable utility.
- RTNCODE - You can set the PTS SYSTEM entry RC parameters in place of RTNCODE.

The following SMP control statement has been added:

- UNLOAD -CDS or ACDS data is punched in UCLIN format.

SMP Control Statement Keywords

The following new SMP control statement keywords have been added:

- ACCEPT - APARS, ASSEM, BYPASS, DIS, USERMODS, RETRY

  1) The APARS and/or USERMODS keyword must be specified in order for ++APAR and/or ++USERMOD system modifications to be accepted into the distribution libraries.

  2) The ASSEM keyword is for SYSMODs that contain both source and object text for the same modules; it is used when the source text is to be assembled to replace the object text.

  3) BYPASS allows you to bypass termination conditions resulting from SYSMOD processing.

  4) DIS allows you to specify a mode for processing the ACDS directory.

  5) RETRY causes SMP to retry following dataset out of space conditions.

- APPLY - ASSEM, BYPASS, DIS, NOJCLIN, RETRY

  1) ASSEM and BYPASS are same as for ACCEPT above.

  2) DIS allows you to specify a mode for processing the CDS directory.

  3) NOJCLIN specifies that all or selected SYSMODs with ++JCLIN modification control statements are not to have the JCLIN data processed.

  4) RETRY causes SMP to retry following dataset out of space conditions.

- JCLIN - DIS

  DIS allows you to specify a mode for processing the CDS directory.

- LIST - ACRQ, CRQ, PTS, SCDS

  Support is included for new or redefined data sets. The CDS is no longer the default for the LIST control statement. Additional options are available on the LIST control statement, including the XREF option which can be specified when listing the ACDS or CDS to produce macro or module cross references or SYSMOD histories.

- RECEIVE - BYPASS

  BYPASS allows you to bypass the function modification identifier check during RECEIVE processing.

- REJECT - PURGE

  PURGE allows you to remove SYSMODS from the PTS which have been accepted.

- RESETRC

  A new control statement that resets the return code values previously returned by SMP functions.

- RESTORE - BYPASS, DIS, RETRY

  1) BYPASS specifies checking functions to be bypassed in the processing of the SYSMODs.

  2) DIS allows you to specify a mode for processing the CDS directory.

  3) RETRY causes SMP to retry following dataset out of space conditions.

- UCLIN - ACRQ, CRQ, DIS, SCDS

1) ACRQ, CRQ and SCDS provide support for new SMP data sets.

2) DIS allows you to specify a mode for processing the CDS or ACDS directory.

3) The CDS is no longer the default data set. A data set name must be specified.

- UCL Statements - SYSMOD

  SYSMOD replaces the PTF keyword.

The following SMP control statement keywords have been eliminated:

- ACCEPT - ERROR, FORCE, LIB, NOLIB, NOREQ, REPLACE

  1) The ERROR, FORCE, LIB, NOLIB, NOREQ and REPLACE keywords are no longer supported. Specification of any of these keywords causes a syntax error.

  2) SYSMODs with the ERROR status indicator set can be processed by specifying their SYSMOD-IDs in the SELECT or GROUP operand list.

  3) The FORCE keyword is replaced by the new keyword, BYPASS, which more accurately describes the resulting SMP action.

  4) The LIB keyword is eliminated because users can update their own permanent libraries rather than the distribution libraries by specifying their own data sets on the DD statements that would normally specify the distribution libraries.

  5) NOLIB has been eliminated because the CDS SYSMOD entries do not require that an ACCEPT indicator be set. This support was for the user who maintained two or more target systems with the same distribution libraries used for RESTORE processing.

  6) NOREQ is replaced by the BYPASS(REQ) option.

  7) REPLACE is unnecessary because of support for user modifications.

- APPLY - ERROR, FORCE, NOASM, NOREQ, REPLACE

  1) ERROR, FORCE, NOREQ, and REPLACE are the same as ACCEPT.

  2) NOASM is unnecessary because assemblies are always required for source module updates unless the module is replaced in the same SYSMOD. Usage of the NOASM keyword results in a syntax error.

- LIST - PDS

The LIST PDS option is no longer valid; the LIST PTS option has been defined. The UCS is no longer used and the MTS and STS data set member names can be listed using IEHLIST or a comparable utility program.

- RECEIVE - FORCE, NOMERGE, PRINT, PUNCH, PTPCH

1) The PRINT, PUNCH, PTPCH, FORCE and NOMERGE keywords are no longer supported. Specification of any of these keywords causes a syntax error.

2) The FORCE keyword has been replaced. A SYSMOD that would not be received because of FMID validation failure can be received by specifying BYPASS(FMID).

3) The NOMERGE keyword is no longer used. SYSMODs are now stored as single entities instead of element replacements and updates, and there is no need to merge SYSMODs that have elements in common.

4) The PRINT, PUNCH and PTPCH keywords are eliminated because of PTS restructuring. Individual updates and replacements within a SYSMOD are not stored as separate members on the PTS. To print or punch the SYSMODs in the PTS data set, use IEBPTPCH or any comparable utility program.

- REJECT - GROUP

The GROUP keyword has been eliminated and specification of this keyword causes a syntax error.

- RESTORE - ERROR, FORCE, NOREJECT, NOREQ

1) ERROR, FORCE and NOREQ are the same as in ACCEPT.

2) A function equivalent to NOREJECT is available through the setting of the REJECT indicator in the PTS SYSTEM entry. If the REJECT indicator is off, a successfully restored SYSMOD is not deleted from the PTS.

- UCLIN - UCS

  The SMPUCS data set is no longer used and specification of the UCS keyword causes a syntax error.

- UCL Statements - SRCUPD, UPDTE, ZAP

  The SRCUPD, UPDTE, and ZAP keywords applied to the PTS data set, which has been redefined.

The following SMP control statement keywords have assumed new meanings:

- APPLY, ACCEPT - GROUP

  The GROUP keyword specifies one or more SYSMODS to be placed into the target system libraries or the distribution libraries. Any requisite and prerequisite SYSMODs are automatically included in the processing, including any requisites and prerequisites of those SYSMODs.

- RESTORE - GROUP

  The GROUP keyword specifies one or more SYSMODs to be removed from the target system libraries, including any other SYSMODs not specified that reference any of the specified SYSMODs as requisites or prerequisites.

SMP Modification Control Statements

The following SMP modification control statements have been added:

- ++APAR - identifies a temporary corrective fix
- ++FUNCTION - identifies new or replacement function
- ++IF - identifies conditional actions
- ++JCLIN - used to include JCL input data within a SYSMOD
- ++MACUPD - identifies a macro update and is interchangable with ++UPDTE
- ++USERMOD - identifies a user modification to IBM software

The SMP REPIN modification control statements are no longer supported.

The following modification control statement has been redefined:

- ++PTF - identifies only IBM supplied service.

SMP Modification Control Statement Keywords

The following new SMP modification control statement keywords have been added:

- ++MAC - DELETE, DISTMOD, DISTSRC, RELFILE, RMID, VERSION
- ++MACUPD/++UPDTE - DISTMOD, DISTSRC, VERSION
- ++MOD - DELETE, LMOD, RELFILE, RMID, VERSION
- ++PTF - FILES
- ++SRC - DELETE, DISTMOD, RELFILE, RMID, VERSION
- ++SRCUPD - DISTMOD, VERSION
- ++VER - DELETE, FMID, VERSION

Messages

The following message activity has occurred:

- Deleted: HMA200, HMA208, HMA209, HMA210, HMA211, HMA212, HMA215, HMA217, HMA220, HMA222, HMA223, HMA225, HMA232, HMA241, HMA242, HMA243, HMA244, HMA245, HMA254, HMA260, HMA265, HMA270, HMA271, HMA272, HMA275, HMA280, HMA286, HMA289, HMA291, HMA293 - HMA301, HMA307 - HMA318, HMA320 - HMA323

- Added: HMA204, HMA261, HMA327, HMA338 - HMA343, HMA345 - HMA370, HMA372 - HMA374, HMA376 - HMA398, HMA400 - HMA402, HMA404 - HMA407, HMA408 - HMA415, HMA418 - HMA432,HMA434 - HMA440

- Redefined: HMA201, HMA203, HMA206, HMA207, HMA214, HMA216, HMA219, HMA224, HMA226, HMA227, HMA231, HMA237, HMA238, HMA239, HMA240, HMA246, HMA247, HMA249, HMA253, HMA256, HMA257, HMA259, HMA262, HMA263, HMA268, HMA274, HMA276, HMA277, HMA281, HMA283, HMA284, HMA302, HMA304, HMA319, HMA324, HMA325, HMA355, HMA359, HMA406, HMA422,

EXEC Card Parameters

The following parameters can be specified in the PARM operand field of the EXEC JCL card:

DATE=U or IPL or REPLY or yyddd

DATE specifies the date to be used for listings and date fields in the created or updated PTS, CDS and ACDS SYSMOD entries. The default is 'U' or 'IPL', which means the date maintained by the operating system. 'yyddd' is the Julian date.

ASM, COMPRESS, COPY, LKED, UPDTE and ZAP are no longer specifiable. See 'The UCL SYS Statement' in Chapter 7 for specification of these program names.

The SIZE parameter is no longer supported. The SIZE parameter for the linkage editor is now contained in the PTS SYSTEM entry and is specified by the UCL SYS LKEDPARM statement.

The NODIS parameter is no longer supported. Directories in storage for the APPLY, ACCEPT, RESTORE, JCLIN and UCLIN functions can be circumvented by specifying DIS(NO) on their

SMP Modification Control Statements

The following SMP modification control statements have been added:

- ++APAR - identifies a temporary corrective fix
- ++FUNCTION - identifies new or replacement function
- ++IF - identifies conditional actions
- ++JCLIN - used to include JCL input data within a SYSMOD
- ++MACUPD - identifies a macro update and is interchangable with ++UPDTE
- ++USERMOD - identifies a user modification to IBM software

The SMP REPIN modification control statements are no longer supported.

The following modification control statement has been redefined:

- ++PTF - identifies only IBM supplied service.

SMP Modification Control Statement Keywords

The following new SMP modification control statement keywords have been added:

- ++MAC - DELETE, DISTMOD, DISTSRC, RELFILE, RMID, VERSION
- ++MACUPD/++UPDTE - DISTMOD, DISTSRC, VERSION
- ++MOD - DELETE, LMOD, RELFILE, RMID, VERSION
- ++PTF - FILES
- ++SRC - DELETE, DISTMOD, RELFILE, RMID, VERSION
- ++SRCUPD - DISTMOD, VERSION
- ++VER - DELETE, FMID, VERSION

Messages

The following message activity has occurred:

- Deleted: HMA200, HMA208, HMA209, HMA210, HMA211, HMA212, HMA215, HMA217, HMA220, HMA222, HMA223, HMA225, HMA232, HMA241, HMA242, HMA243, HMA244, HMA245, HMA254, HMA260, HMA265, HMA270, HMA271, HMA272, HMA275, HMA280, HMA286, HMA289, HMA291, HMA293 - HMA301, HMA307 - HMA318, HMA320 - HMA323
- Added: HMA204, HMA261, HMA327, HMA338 - HMA343, HMA345 - HMA370, HMA372 - HMA374, HMA376 - HMA398, HMA400 - HMA402, HMA404 - HMA406, HMA408 - HMA415, HMA418 - HMA432
- Redefined: HMA201, HMA203, HMA206, HMA207, HMA214, HMA216, HMA219, HMA224, HMA226, HMA227, HMA231, HMA237, HMA238, HMA239, HMA240, HMA246, HMA247, HMA249, HMA253, HMA256, HMA257, HMA259, HMA262, HMA263, HMA268, HMA274, HMA276, HMA277, HMA281, HMA283, HMA284, HMA302, HMA304, HMA319, HMA324, HMA325

EXEC Card Parameters

The following parameters can be specified in the PARM operand field of the EXEC JCL card:

DATE=U or IPL or REPLY or yyddd

DATE specifies the date to be used for listings and date fields in the created or updated PTS, CDS and ACDS SYSMOD entries. The default is 'U' or 'IPL', which means the date maintained by the operating system. 'yyddd' is the Julian date.

ASM, COMPRESS, COPY, LKED, UPDTE and ZAP are no longer specifiable. See 'The UCL SYS Statement' in Chapter 7 for specification of these program names.

The SIZE parameter is no longer supported. The SIZE parameter for the linkage editor is now contained in the PTS SYSTEM entry and is specified by the UCL SYS LKEDPARM statement.

The NODIS parameter is no longer supported. Directories in storage for the APPLY, ACCEPT, RESTORE, JCLIN and UCLIN functions can be circumvented by specifying DIS(NO) on their respective control statements.

Reorganization of the Publication

The publication has been reorganized; it now consists of ten chapters, described in the preface, and includes more examples and guidelines.

Chapter 1: Introduction

The System Modification Program (SMP) is a service aid that provides the facility to install IBM- or user-supplied system modifications into OS/VS1 or OS/VS2 MVS system libraries and distribution libraries (DLIBs). SMP maintains extensive records of the contents and status of your libraries. In addition, it can be used to back out system modifications from your system libraries, if required.

Distribution Libraries

The distribution libraries are used to generate a new system control program (SCP). Some of the distribution libraries supplied by IBM contain modules that are assembled or link edited during system generation (SYSGEN) into system data sets. These modules might be, for example, utility programs, data management routines, or error recovery routines. Some distribution libraries contain modules that are copied during SYSGEN in their entirety into system data sets, such as macro definitions for system macros, system parameter lists, or cataloged procedures. Some distribution libraries contain macro definitions used during the Stage I or Stage II assemblies in the system generation process.

Creating System Libraries

The system generation process uses the distribution libraries to create a system control program tailored to the data processing and machine configuration requirements of an installation. There are also analogous processes used to generate system libraries for program products.

A complete SYSGEN is done when you are installing an SCP for the first time or when you must modify an existing SCP. An I/O device generation (IOGEN) is done when changes need to be made to the machine configuration only, such as adding I/O devices to an installation.

A SYSGEN is processed in two stages. In Stage I, the SYSGEN macro instructions that you coded are assembled and expanded to form a jobstream. In Stage II, the jobstream is used to assemble, link edit, and copy selected modules from the distribution libraries, and user-supplied components from user data sets, to system data sets to build a new SCP or modify an existing SCP. These system data sets are referred to by SMP as target system libraries, and the level of the system created is referred to as a base level system.

IBM logically groups the modules, macros, and source modules (referred to by SMP as elements) in the base level system into what are known as functions, such as TSO or VTAM. Each function is considered to "own" the elements that comprise it. Ownership of elements and the relationships between elements are specified using the SMP modification control statements, described in Chapters 2 and 8.

The Control Data Set

SMP maintains information about the elements that comprise the system on a Control Data Set (CDS). SMP creates the CDS with information about the modules, macros, and source modules that comprise the new system. SMP also builds load module, assembler, and distribution library entries on the CDS to describe the structure of the system. A description of the processing required to allocate and build the SMP data sets is found in Chapter 4.

SMP maintains the following entries on the CDS:

- Assembler (ASSEM) entries: SMP saves the assembler statements generated by the Stage I SYSGEN process. This allows SMP to automatically reassemble the modules affected by Stage II macro maintenance.

- Load module (LMOD) entries: SMP maintains an entry in the CDS for load modules in the system. Each LMOD entry contains information about the load module's linkage editor attributes and control statements and the system libraries in which the load module resides.

- Macro (MAC) entries: SMP maintains an entry in the CDS for macros in the system. Each MAC entry contains information about the macro's system library (if any), the distribution library, and what Stage II assemblies (see the Assembler entries) that the macro appears in.

- Module (MOD) entries: SMP maintains an entry in the CDS for modules in the system. Each MOD entry contains information about the module's distribution library and which load modules contain this module.

- Distribution library (DLIB) entries: SMP maintains an entry in the CDS for each DLIB that was copied in its entirety to a SYSLIB at SYSGEN time.

- Source (SRC) entries: SMP maintains an entry in the CDS for source modules in the system. Each entry contains the source module name and information about the source modules's system library (if any) and distribution library.

- System modification (SYSMOD) entries: SMP maintains an entry in the CDS for each SYSMOD installed on the system. The entry is used to track the status of that SYSMOD and to maintain a historical record of the modifications made. The entry is also used by SMP when checking whether it can fulfill required prerequisites that are specified in other system modifications.

- SYSTEM entry: The SYSTEM entry is created by the user using the UCL SYS statement described in Chapter 7. SMP uses this entry to verify that a valid CDS is being used and to determine some SMP processing options.

The 'LIST Control Statement' in Chapter 7 contains examples of the CDS entries. Information about the creation and use of these entries is found in 'JCLIN Processing' in Chapter 3.

The Alternate Control Data Set

In addition to the CDS, SMP uses the Alternate Control Data Set (ACDS) to describe the elements and system modifications contained in the distribution libraries. The entries in the ACDS are similar to the CDS entries, except that the ACDS does not have LMOD, DLIB, or ASSEM entries.

SMP Processing

Once the base level system has been generated, SMP is used to install subsequent system modifications to elements on the system libraries or distribution libraries. By installing the system modification on the distribution libraries, future system generations performed using these distribution libraries produce system libraries that reflect the modified elements. A system modification installed in the distribution libraries is considered to be permanent by SMP; SMP cannot be used to remove it.

A conceptual overview of SYSGEN processing, CDS creation, and SMP processing is shown in Figure 1.

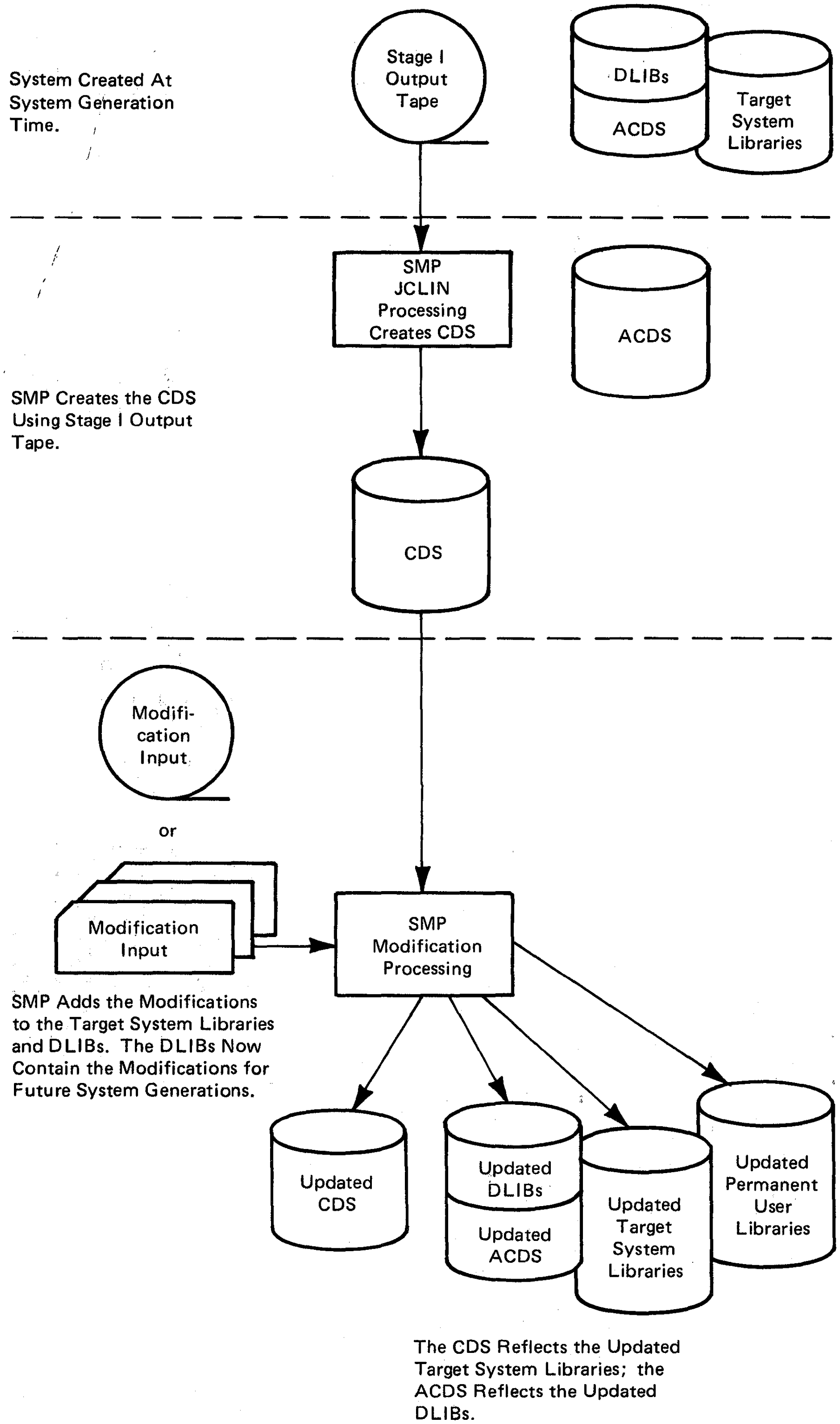

Figure 1. Processing Overview

Types of System Modifications

SMP can process two classes of system modification (SYSMOD), described in detail in Chapter 2:

- Service system modifications - IBM-supplied or locally prepared modifications that update or replace existing elements
- Function modifications - redefine or introduce new elements to the SCP or to a program product

System modifications are constructed using SMP modification control statements, described in Chapter 8. These statements identify the type of modification and the elements to be added to, modified in, or deleted from the system libraries and distribution libraries. In addition, there are modification control statements that describe the environment and conditions that must be met in order for SMP to install the modification.

The modification control statements and the modification text (such as macro definition statements for a macro replacement) that comprise one system modification are referred to by SMP as a package.

Each system modification has an identifier, referred to by SMP as the SYSMOD-ID. SMP uses the SYSMOD-ID to track that status and history of the modifications made and to identify dependencies on other system modifications.

System Modification Processing

The SMP control statements, specified by the user, are used to tell SMP the type of processing to perform. The control statements are described in Chapter 7. The principle control statements are listed alphabetically as follows:

- ACCEPT - places SYSMODs into the distribution libraries
- APPLY - places SYSMODs into the target system libraries
- JCLIN - processes Stage I output from SYSGEN (or similar job step JCL) to create or update the CDS

- RECEIVE - starts processing of a SYSMOD by performing syntax and validity checking and saves the SYSMOD on the PTF Temporary Store Data Set (PTS)

- REJECT - deletes SYSMODs from the PTS

- RESTORE - removes a modification from the target system libraries

- UCLIN - updates SMP data sets

To install a system modification, SMP performs a number of functions, depending upon the type of update and the options requested on the control statements and the modification control statements. The processing that takes place in SMP is described in detail in Chapter 3.

A simplification of the flow of system modification processing is shown in Figure 2.

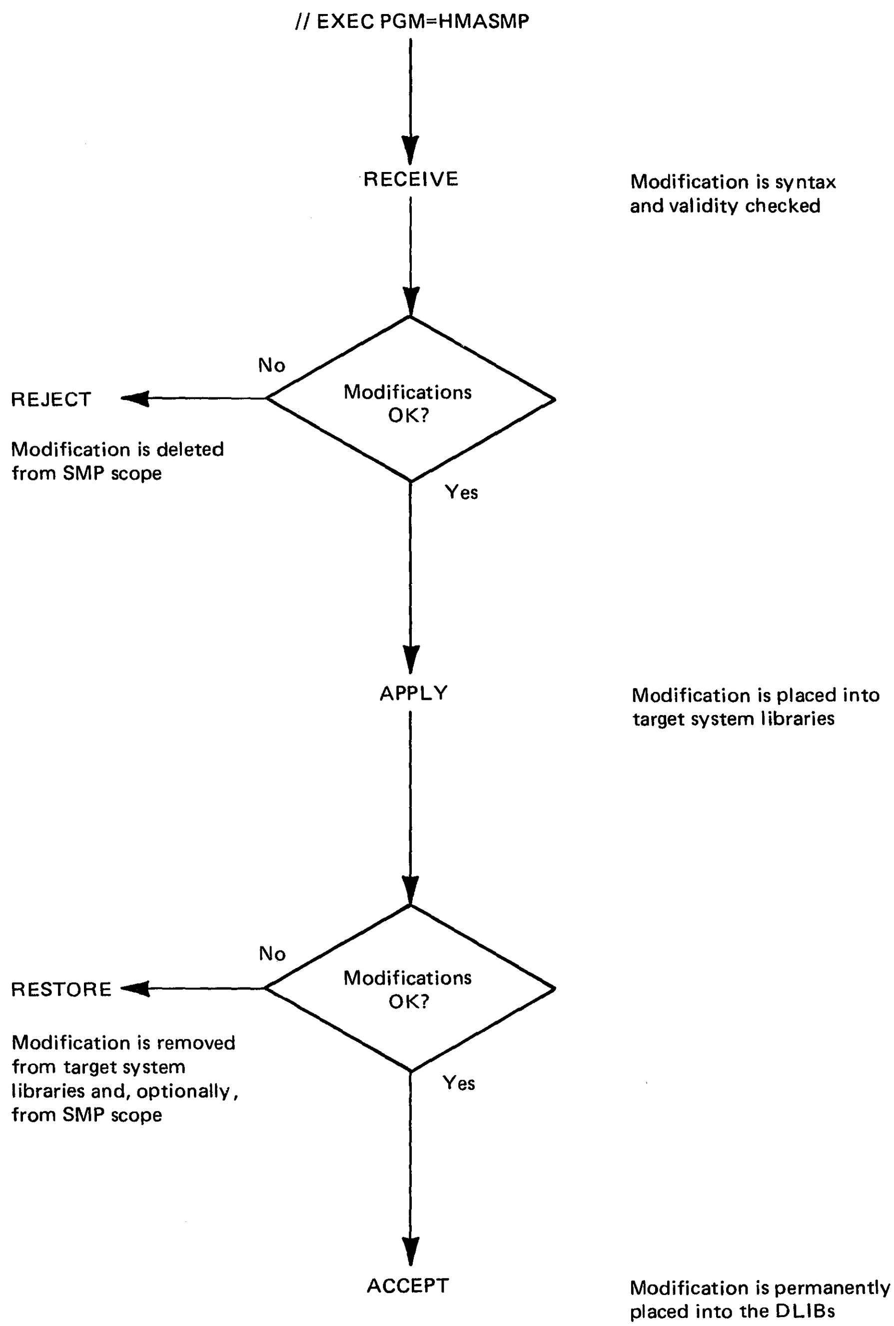

Figure 2. Simplified View of SYSMOD Processing

SMP Data Sets

SMP requires a number of data sets. The total number required is determined by the control statements being executed. The data sets are described in Chapter 9; their use during SMP processing is described in Chapter 3.

Executing SMP

SMP executes as a job running under the operating system. To initiate SMP processing, you code a JOB and an EXEC statement, and DD statements for the data sets that are required by the SMP control statements that you wish to process. Detailed information about executing SMP is contained in Chapter 4.

SMP Reports, Listings, and Messages

SMP produces reports and messages to indicate the status of SYSMOD processing. The reports are described in Chapter 6. The messages are contained in Chapter 10. In addition, the SMP LIST control statement, described in Chapter 7, enables you to list all of the entries or selected entries on the SMP data sets.

System modifications (SYSMODs) are the input data to SMP; they define the additions, replacements, and updates to modules, macros, and source modules, referred to by SMP as elements, in the operating system and associated distribution libraries. This chapter explains the differences between the various types of SYSMODs and highlights the use of the information provided within a SYSMOD.

Characteristics of IBM Operating Systems

An operating system distributed by IBM typically consists of a set of distribution libraries containing macros and modules that are used to create your tailored version of the operating system, referred to by SMP as the target system. You decide what your target system should include by the process of system generation (SYSGEN). After you have done your SYSGEN, you invoke SMP to analyze the Stage I output job stream in order to create entries on the CDS to define load modules (LMOD entries), assembler modules (ASSEM entries), object modules (MOD entries), macros (MAC entries), and distribution libraries (DLIB entries). This is referred to as the base level of your target system.

Having done the above, you have a target system, a set of distribution libraries and some SMP data sets. Any subsequent changes to your target system and distribution libraries should be done via SMP.

Program Products

Program products are generally installed separately from the generation of a target system. Some program products are independent of the operating system with respect to release level or type of operating system. For instance, a release of a program product may be applicable to both OS/VS1 and OS/VS2 systems. Other program products have dependencies upon the contents of your target system and may require different levels of functional capability in your target system. For instance, a program product may need product X, release level 1, on an OS/VS1 system and product X, release level 2, on an OS/VS2 system.

SMP provides the capability to install a program product directly to a target system. Dependencies on content and release levels of both the base level target system and other program products are specified by the developers of the system modification package for the program product. Although there may be a requirement for you to execute some programs and procedures outside the scope of SMP, the most significant portion of the tasks necessary to install the program product are done by SMP.

Types of System Modifications

Four types of system modifications are defined for SMP. The reason for the differentiation is to ensure that the processing of the SYSMODs updates the target system and distribution libraries correctly with respect to functional and service levels. The types of SYSMODs are differentiated by the header modification control statement. Following the header name is an operand which is the name of the SYSMOD. This name is referred to as the SYSMOD-ID.

The first type of SYSMOD is used to package base level system components and program products. This type of SYSMOD has a header modification control statement identified as ++FUNCTION. A function SYSMOD initially defines elements of the base system and program products. Subsequent function SYSMODs, referred to as selectable units (SUs) or features, may redefine elements of the base level system or program products.

The second type of SYSMOD is used to package permanent changes to elements of IBM software components. This type of SYSMOD has a header modification control statement identified as ++PTF. A PTF generally is used to service elements, although new elements may be defined in a PTF. PTFs are usually automatically distributed to users of IBM operating systems and program products.

The third type of SYSMOD is used to package temporary corrective changes to elements of IBM software components. This type of SYSMOD has a header modification control statement identified as ++APAR. An APAR is usually not automatically distributed to users of IBM software and in many instances it is the user's responsibility to create the APAR SYSMOD input prior to executing the RECEIVE function of SMP.

The fourth type of SYSMOD is used for user modifications to IBM software. This type of SYSMOD has a header modification control statement identified as ++USERMOD. You create a user modification to change or replace elements of IBM software. It can also be used to define elements that you have created to interface with IBM software.

Function SYSMODs

Function SYSMODs are used to identify system components, program products, and features of both. The two basic types of function SYSMODs are called base level function SYSMODs and dependent level function SYSMODs, also referred to as a feature level function SYSMOD.

The characteristic that differentiates base level from dependent level function SYSMODs is the presence of the FMID keyword on the ++VER modification control statements of dependent level functions. FMID is the abbreviation for function modification identifier and is used to specify ownership and dependency. The absence of the FMID keyword on base level function SYSMODs means there is no dependency for those functions in terms of requiring a base upon which the functions are built. The FMID keyword for a dependent level function identifies the SYSMOD-ID of a base level or another dependent level function SYSMOD, which is an absolute prerequisite; that is, the dependent level function cannot exist in the operating system or program product environment without the presence of the function specified in the FMID keyword.

Base level functions, when installed on the target system, add new elements to the target system. When the element entries (that is, the MAC, MOD, and SRC entries) are created on the CDS, the SYSMOD-ID from the ++FUNCTION modification control statement becomes the value of the FMID subentry for those entries. This identifies the owning function for those elements. All subsequent SYSMODs that replace or update those elements must specify the SYSMOD-ID of the owning SYSMOD as an operand of either the FMID or VERSION keyword on the ++VER modification control statement or the VERSION keyword on the associated element modification control statement. The VERSION keyword is discussed under that topic later in this chapter. The RMID (replacement modification identifier) subentry in the element entries are also set to the SYSMOD-ID from the ++FUNCTION modification control statement. This identifies the SYSMOD that last replaced the element.

Dependent level functions, when installed on the target system, add new elements to or replace existing elements in the base level function or dependent level functions. The element entry for each applied element in a dependent level SYSMOD has the SYSMOD-ID from the ++FUNCTION modification control statement placed in the FMID subentry. This indicates a change in ownership for those elements that were part of existing base level or dependent level functions. The SYSMOD-ID is also placed in the RMID subentry. For any element entries that existed before the dependent level function was applied, any UMID (update modification identifier) subentries are deleted from the element entries. These subentries identified SYSMODs previously applied that updated, rather than replaced, the element.

The FMID, RMID, and UMID subentries in the element entries are used by SMP to determine whether an element modification within a SYSMOD should be applied to the target system. RMID and UMID checking and updating are described in greater detail under 'APPLY Processing' in Chapter 3.

When function SYSMODs are applied, a SYSMOD entry is created on the CDS. This entry contains information extracted from the applicable ++VER modification control statement and subentries for each element contained in the SYSMOD. In addition, an FMID subentry is also present in the SYSMOD entry. For dependent level functions, the value of the FMID subentry is set to the value in the FMID keyword from the ++VER modification control statement. For base level functions, the FMID subentry value is set to the SYSMOD-ID in the header modification control statement.

When function SYSMODs are accepted into the distribution libraries, the same updating of element and SYSMOD entries occurs on the ACDS as was done on the CDS during application to the target system.

PTF SYSMODs

PTF SYSMODs are generally created to service system components, program products, and their features. However, a PTF may introduce new elements for a function. Every PTF must contain an FMID keyword in its ++VER modification control statements. This identifies the owning function SYSMOD of the elements included in the PTF.

When a PTF is applied to the target system, SMP processes the applicable elements within a PTF and updates or replaces those elements in the target system. When an element is updated, SMP adds a UMID subentry containing the SYSMOD-ID of the ++PTF modification control statement to the associated element entry on the CDS. When an element is replaced, SMP replaces the RMID subentry value in the associated element entry with the SYSMOD-ID from the ++PTF modification control statement and deletes all UMID subentries from the entry.

As with function SYSMODs, PTFs that are accepted into the distribution libraries will result in the same updating of ACDS entries as occurred for the CDS entries during application to the target system.

APAR SYSMODs

APAR SYSMODs are generally created to service system components as a temporary corrective fix. The same type of processing occurs as for a PTF except during ACCEPT processing. To accept an APAR SYSMOD into the distribution libraries, you must specify the APARS keyword on the ACCEPT control statement. This extra action on your part protects you from inadvertently updating a set of distribution libraries that you may wish to keep free of temporary fixes. Usually, APAR SYSMODs will only update, rather than replace, elements.

USERMOD SYSMODs

USERMOD SYSMODs are created by you, the user, to either modify or replace IBM elements or your own applications. The same processing occurs as for a PTF except during ACCEPT processing. To accept a USERMOD SYSMOD into the distribution libraries, you must specify the USERMODS keyword on the ACCEPT control statement. This extra action on your part protects you from inadvertently placing your modifications in a set of distribution libraries that should contain only IBM elements.

The ++VER Modification Control Statement

The ++VER modification control statement is used to ensure that the SYSMOD being processed belongs on your system and the distribution libraries. It also identifies SYSMODs required in order for this SYSMOD to be applied to your target system and accepted into the distribution libraries. The SYSMODs and APARs that are superseded by the SYSMOD are also specified.

Each keyword in the ++VER modification control statement is described briefly in the following topics. More detail on the processing of these keywords is in Chapter 3.

The DELETE Keyword

The DELETE keyword specifies one or more function SYSMODs and their elements that are to be deleted from the target system when the SYSMOD is applied and from the distribution libraries when the SYSMOD is accepted. You can only specify this keyword in a function SYSMOD. There are two primary purposes for this keyword. First, a function SYSMOD can replace previous releases or versions of the same function. Second, a function can have a mutually exclusive relationship with another function. In this case, both function SYSMODs may specify the other as operands of the DELETE keyword.

The documentation that accompanies function SYSMODs should include information about any function SYSMODs that it deletes. This information should help you to determine if you must selectively APPLY and ACCEPT the SYSMOD and help you identify SYSMODs that will be deleted and may have to be reapplied and reaccepted.

The FMID Keyword

The FMID keyword has been described in the above sections.

The NPRE Keyword

The NPRE keyword specifies one or more function SYSMODs that must not be present if the SYSMOD is to be applied to the target system or accepted into the distribution libraries. You can only specify NPRE in function SYSMODs. NPRE (negative prerequisite) is generally used with mutually exclusive functions. In prior releases of SMP, the NPRE keyword could appear in a PTF SYSMOD. This is no longer true for PTFs that are processed by this release of SMP However, PTF SYSMODs can be constructed that are processable by both this release of SMP and the previous release of SMP. This is described under 'Combined Packaging for Compatibility' later in this chapter.

The PRE Keyword

The PRE keyword specifies one or more SYSMODs that must be applied to the target system prior to or concurrent with the SYSMOD and must be accepted into the distribution libraries prior to or concurrent with the SYSMOD. PRE means prerequisite and is used to determine the order of processing. As a rule, you should not specify the PRE operand in base level function SYSMODs. The REQ keyword, described in 'The REQ Keyword,' should be specified instead.

The REQ Keyword

The REQ keyword specifies one or more SYSMODs that must be applied to the target system prior to or concurrent with the SYSMOD and must be accepted into the distribution libraries prior to or concurrent with the SYSMOD. REQ (requisite) is generally used when the SYSMODs, specified as operands, do not have a processing order relationship.

The SUP Keyword

The SUP keyword specifies one or more SYSMODs that are superseded by the SYSMOD and APARs that are fixed by the SYSMOD. SMP does not verify that the superseded SYSMODs are installed on your system; however, this does not cause any processing problems because a superseded SYSMOD does not have to be processed before, concurrent with, or after the SYSMOD that supersedes it. A SYSMOD may or may not contain all the elements modified in the SYSMODs that it supersedes. However, when it does not, it specifies, as operands of the REQ keyword, those SYSMODs that contain modifications to the elements not present in the SYSMOD but present in the superseded SYSMOD.

The VERSION Keyword

The VERSION keyword specifies one or more SYSMODs that contain functionally inferior versions of some or all of the elements present in the SYSMOD. The VERSION keyword identifies other function SYSMODs that are in the same hierarchy as the base level function but are not in the same hierarchical path as the SYSMOD. Figure 3 shows a functional hierarchy. All the functions are in the same hierarchy as function GXY1000. However, functions FXY1020 and FXY1030 are in different hierarchical paths. Therefore, if these two functions have common elements, one must be considered superior to the other. If function FXY1030 is superior to function FXY1020, you must specify function FXY1020 as an operand of the VERSION keyword in SYSMOD FXY1030's ++VER modification control statement. For example, the first two modification control statements for function FXY1030 could be:

```
++FUNCTION(FXY1030).
++VER(Z038) FMID(GXY1000) VERSION(FXY1020,FXY1040).
```

Notice that the base level function (GXY1000) is not specified in the VERSION keyword operand list. This is because FXY1030 is a dependent function of GXY1000, which is specified via the FMID keyword, and any elements common to both are assumed to be superior in the dependent function. In the above example, FXY1040 is also specified as a VERSION operand. This is necessary if the elements in FXY1030 are superior to those in FXY1040. If this is not true, then FXY1040 would not be specified as an operand. It is mandatory that the creators of function SYSMODs ensure that the functional relationships be specified correctly. For instance, if the SYSMOD for FXY1050 does not specify FXY1040 in its VERSION keyword operand list, and vice-versa, then if both are applied to the target system, the correct versions of the elements may not be present depending upon the order of application. That is, if FXY1050 is superior to FXY1040 but does not specify FXY1040 in the VERSION keyword operand list and FXY1040 is already applied to the target system, then when FXY1050 is applied, the elements common to both functions will not be processed from FXY1050.

If, on the other hand, FXY1050 did specify FXY1040 in the VERSION operand list and was applied to your target system, and subsequently FXY1040 was applied, the elements in common would not be selected from FXY1040 since it did not specify FXY1050 in the VERSION operand list.

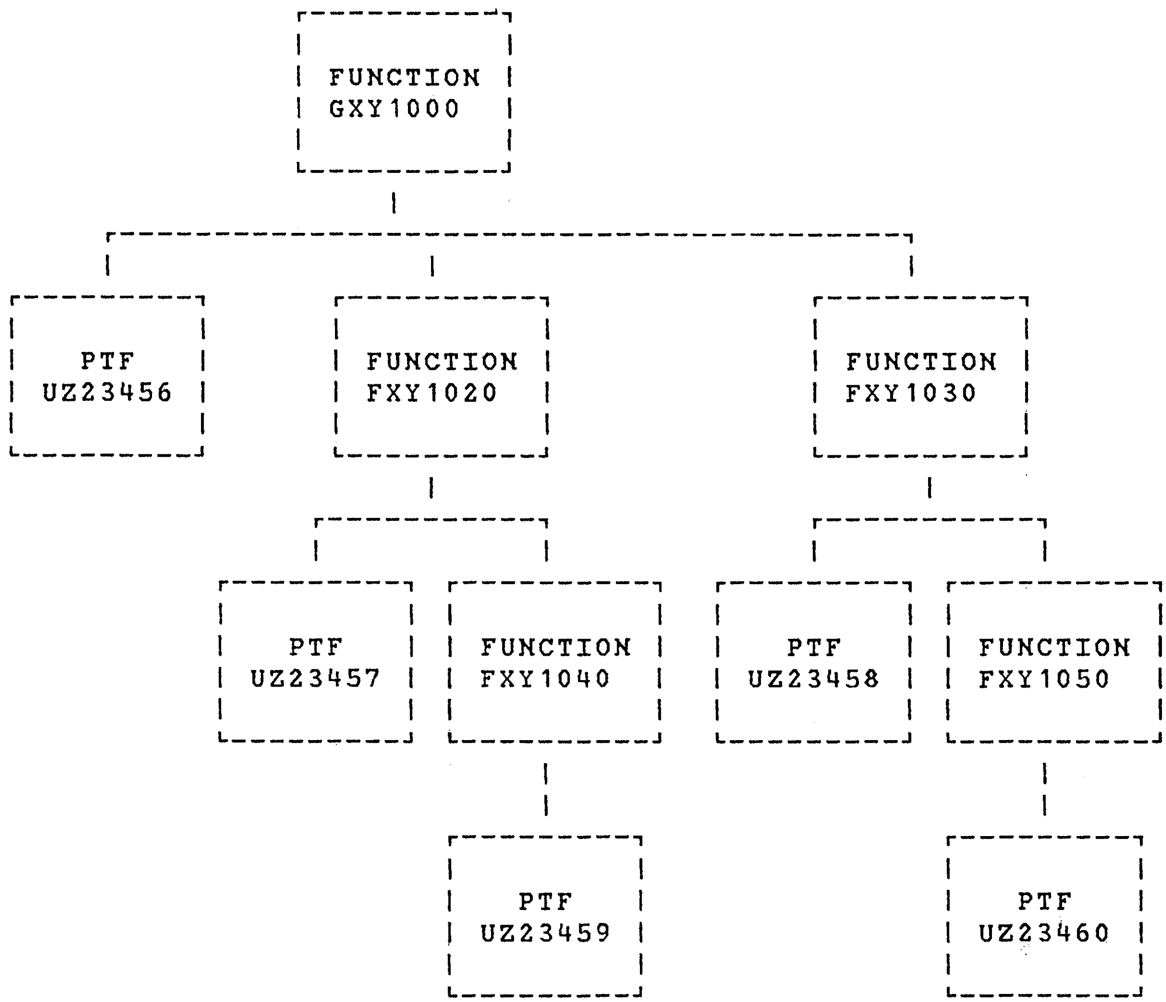

Figure 3. Hierarchy of Function and Service

The VERSION keyword is also used to transfer ownership of elements to another function. This is generally specified in a PTF, as is referred to by SMP as element version collapse. For example, if modules XYZABC and XYZDEF are part of both functions FXY1020 and FXY1030, and it is necessary to remove them from FXY1030, the following PTF could be constructed:

```
++PTF(UZ12345).
++VER(Z038) FMID(FXY1020) VERSION(FXY1030).
++MOD(XYZABC) DISTLIB(DLIB01).
++MOD(XYZDEF) DISTLIB(DLIB01).
```

If both FXY1020 and FXY1030 are applied to your target system, when this PTF is applied, SMP will change the FMID subentries in both MOD entries to FXY1020.

The ++IF Modification Control Statement

The ++IF modification control statement specifies one or more SYSMODs that must be applied to the target system and accepted into the distribution libraries when a particular function SYSMOD is installed. This is referred to as a conditional action. The condition is the presence of a function SYSMOD, specified by the FMID keyword. The action is the requirement for other SYSMODs to be installed, specified by the REQ keyword.

The processing of the conditional action occurs under two conditions, as follows:

- If the function SYSMOD specified in the FMID operand is already applied or accepted when the SYSMOD with the ++IF modification control statement is applied or accepted

- If the SYSMOD containing the ++IF modification control statement is applied or accepted and the function SYSMOD specified in the FMID operand is subsequently applied or accepted.

  For example:

  ```
  ++IF FMID(FXY1030) THEN REQ(UZ12346).
  ```

means that if function SYSMOD FXY1030 is applied, SYSMOD UZ12346 must also be applied, or when function SYSMOD FXY1030 is applied, then the SYSMOD UZ12346 must be applied concurrently, if not already applied.

This statement prevents regression of the system components when the environment changes. SMP saves the ++IF modification control statements from each SYSMOD applied and accepted so that functions not present at the time the SYSMOD was processed are brought up to the required service level when they are subsequently applied and accepted.

Any number of ++IF modification control statements can appear in a SYSMOD. In the following example, which is derived from Figure 3, the PTF modifies elements in the base level function that are also included in dependent level functions.

```
++PTF(UZ23456).
++VER(Z038) FMID(GXY1000).
++IF FMID(FXY1020) THEN REQ(UZ23457).
++IF FMID(FXY1030) THEN REQ(UZ23458).
++IF FMID(FXY1040) THEN REQ(UZ23459).
++IF FMID(FXY1050) THEN REQ(UZ23460).
++MOD(XYZABC) DISTLIB(DLIB01).
++MOD(XYZDEF) DISTLIB(DLIB01).
++MOD(XYZGHI) DISTLIB(DLIB01).
```

Although they are not shown, each of the SYSMODs specified as REQ keyword operands would also have ++IF modification control statements that specify each other in accordance with the superiority of the module versions. This set of SYSMODs is called a requisite SYSMOD set. If all of the function SYSMODs specified in the FMID keywords are applied, all the SYSMODs specified in the REQ keywords must be applied when UZ23456 is applied.

The ++JCLIN Modification Control Statement

The ++JCLIN modification control statement specifies that JCL input data is included in the SYSMOD. JCLIN processing is described in Chapter 3. The JCL input data for the SYSMOD will be processed when the SYSMOD is applied to your target system. By associating the JCL input data with the SYSMOD, you are assured that the needed updates to the CDS will be done at the proper point in the processing. 'APPLY Processing' in Chapter 3 describes how the ++JCLIN modification control statement is processed when the SYSMOD is applied to your target system.

The methods of packaging the JCL input data are described under 'Packaging Techniques for SYSMODs' later in this chapter.

The ++MAC Modification Control Statement

The ++MAC modification control statement describes a macro that is being added to, replaced in, or deleted from the target system and the distribution libraries. Macros can be used in the system generation process, contained in source modules, or intended for general use. Some entities of the target system that are not macros are defined using the ++MAC modification control statement. Examples of these are members of the PARMLIB and HELP data sets.

The methods of packaging a macro are described under 'Packaging Techniques for SYSMODs' later in this chapter.

The ASSEM Keyword

The ASSEM keyword specifies source modules that require assembly due to a change to the macro. The source module names are in the CDS as ASSEM or SRC entries and in the ACDS as SRC entries. After the macro is placed in the appropriate target system library or the MTS data set during APPLY processing, SMP assembles these source modules and link edits the resultant object text into the required target system load module libraries. 'APPLY Processing' in Chapter 3 describes this process in detail and 'ACCEPT Processing' describes the differences for distribution library updating.

The DELETE Keyword

The DELETE keyword specifies the deletion of a macro from the hierarchy of the owning function. DELETE is used to change ownership of the macro or to remove the macro from the target system and distribution libraries.

When changing ownership of a macro, the SYSMOD that contains the ++MAC modification control statement specifies, as requisites, the SYSMODs that are to own versions of the macro. In the following example, which is based on Figure 3, a macro named XYZMAC1 was a part of functions GXY1000, FXY1020, and FXY1030; the version of the macro in FXY1030, which is superior, is to be collapsed.

```
++PTF(UZ12345).
++VER(Z038) FMID(GXY1000) VERSION(FXY1030).
++IF FMID(FXY1020) THEN REQ(UZ12346).
++IF FMID(FXY1030) THEN REQ(UZ12347).
++MAC(XYZMAC1) DISTLIB(XYZMACS).

++PTF(UZ12346).
++VER(Z038) FMID(FXY1020) VERSION(FXY1030).
++IF FMID(FXY1030) THEN REQ(UZ12347).
++MAC(XYZMAC1) DISTLIB(XYZMACS).

++PTF(UZ12347).
++VER(Z038) FMID(FXY1030) REQ(UZ12345).
++MAC(XYZMAC1) DELETE.
```

Notice that the complete set of PTFs is related using the ++IF modification control statements contained in PTFs UZ12345 and UZ12346 and the REQ keyword in the ++VER modification control statement for PTF UZ12347. Since function FXY1030 cannot be present without function GXY1000, and function FXY1020 may or may not be present, the relationship to PTF UZ12346 need not be specified in PTF UZ12347. The same situation is true for function FXY1020 with respect to the other two functions. However, if PTF UZ12345 is never processed, there would be no bond between UZ12346 and UZ12347 without the ++IF modification control statement in UZ12346 or a REQ keyword specifying UZ12345 in the ++VER modification control statement for UZ12346. The choice shown in this example is to use the ++IF modification control statement in UZ12346 since UZ12345 does not have to be processed if function FXY1020 is present.

When deleting a macro, the macro is completely deleted from the target system and distribution libraries. It is critical that you update or replace any source modules that contain a deleted macro to remove any references to the macro.

The DISTLIB Keyword

The DISTLIB keyword specifies the ddname of the distribution library that contains the macro. You must supply a JCL DD statement whose ddname is the same as the DISTLIB operand when you accept the macro replacement into the distribution library. The DD statement must define the data set to be updated. If you are going to accept the macro into an alternate distribution library, the DD statement must define that library.

If service SYSMODs are constructed that change the DISTLIB of macros, you should provide information regarding which DISTLIB subentries SMP should change for each affected MAC entry on the ACDS prior to ACCEPT processing. You should provide a ++JCLIN modification control statement to change DLIB entries during APPLY processing.

The DISTMOD Keyword

The DISTMOD keyword specifies the ddname of the distribution library that will be updated during ACCEPT processing by link editing the modules produced from assemblies of the source modules specified in the ASSEM keyword operand list. See 'APPLY Processing' and 'ACCEPT Processing' in Chapter 3 for a complete description of how this keyword is used. For compatibility with previous releases of SMP, an alternate name for DISTMOD is DISTOBJ.

The DISTSRC Keyword

The DISTSRC keyword specifies the ddname of the distribution library where the source modules specified in the ASSEM keyword operand list reside. See 'APPLY Processing' and 'ACCEPT Processing' in Chapter 3 for a complete description of how this keyword is used. For compatibility with previous releases of SMP, an alternate name for DISTSRC is ASMLIB.

The MALIAS Keyword

The MALIAS keyword specifies alias names of the macro. See 'APPLY Processing' in Chapter 3 for a complete description of how this operand is used.

The RMID Keyword

The RMID keyword specifies the SYSMOD-ID of the last PTF that replaced the macro when that PTF is incorporated into a service updated function SYSMOD. See 'Service Updated Function SYSMODs' in this chapter for a complete description of this operand.

The SSI Keyword

The SSI keyword specifies SSI information to be placed in the directory entry for the macro in the target system library when the SYSMOD is applied and the distribution library when the SYSMOD is accepted. See 'APPLY Processing' in Chapter 3 for a complete description of how this operand is used.

The SYSLIB KEYWORD

The SYSLIB keyword specifies the ddname of the target system library containing the macro. You must supply a JCL DD statement whose ddname is the same as the SYSLIB operand when you apply the macro replacement to the target system library. The DD statement must define the data set to be updated on your target system.

The VERSION Keyword

The VERSION keyword is used in the same manner as the VERSION keyword on the ++VER modification control statement. When it is specified on a ++MAC modification control statement, it overrides the VERSION keyword specified on the ++VER modification control statement.

The ++MOD Modification Control Statement

The ++MOD modification control statement describes a module that is being added to, replaced in, or deleted from the target system and the distribution libraries.

The methods of packaging a module are described in 'Packaging Techniques for SYSMODs' later in this chapter.

The DALIAS and TALIAS Keywords

The DALIAS and TALIAS keywords specify the alias names of the module in the target system library and the distribution library. If the module was copied during system generation to the target system library, the load module will have the alias names specified when the SYSMOD is applied. If the module is link edited with other modules to form a target system load module, the alias names are ignored when the SYSMOD is applied. When the SYSMOD is accepted, the alias names are placed in the distribution library. See 'APPLY Processing' in Chapter 3 for a complete description of how these operands are processed.

The DELETE Keyword

The DELETE keyword specifies the deletion of a module from the hierarchy of the owning function. DELETE changes the ownership of the module or removes the module from the target system and distribution libraries.

When the SYSMOD is applied, if the module was link edited with other modules to form a target system load module, the module is not deleted from the load module because the CSECT name may not match the name of the module and the module may contain more than one CSECT. When a module is deleted, one or more of the other modules in the load module will be modified, either within the SYSMOD or by requisite SYSMODs, so that the deleted module is not referenced or executed.

See 'The DELETE Keyword' for the ++MAC modification control statement for a complete description of the use of this keyword.

The DISTLIB Keyword

The DISTLIB keyword specifies the ddname of the distribution library that contains the module. See 'The DISTLIB Keyword' for the ++MAC modification control statement for a complete description of the use of this keyword.

If service SYSMODs change the DISTLIB of modules, information about which DISTLIB subentries should be changed for each affected MOD entry on the ACDS prior to ACCEPT processing should be provided with the SYSMOD. You can use a ++JCLIN modification control statement to change entries during APPLY processing.

The LEPARM Keyword

The LEPARM keyword specifies the parameters to be passed by SMP to the linkage editor program when the module is link edited during APPLY and ACCEPT processing.

The RMID Keyword

The RMID keyword specifies the SYSMOD-ID of the last PTF that replaced the module when that PTF is incorporated into a service updated function SYSMOD. See 'Service Updated Function SYSMODs' later in this chapter for a complete description of this operand.

The VERSION Keyword

The VERSION keyword is used in the same manner as the VERSION keyword on the ++VER modification control statement. When it is specified on a ++MOD modification control statement, it overrides the VERSION keyword specified on the ++VER modification control statement.

The ++SRC Modification Control Statement

The ++SRC modification control statement describes a source module that is being added to, replaced in, or deleted from the target system and the distribution libraries.

The methods of packaging a source module are described under 'Packaging Techniques for SYSMODs' later in this chapter.

The DELETE Keyword

The DELETE keyword specifies the deletion of a source module from the hierarchy of the owning function. DELETE changes the ownership of the source module or the removes the source module from the target system and distribution libraries. See 'The DELETE Keyword' for the ++MAC modification control statement for a complete description of use.

The DISTLIB Keyword

The DISTLIB keyword specifies the ddname of the distribution library containing the source module. You must supply a JCL DD statement whose ddname is the same as the DISTLIB operand when you accept the source module replacement into the distribution library. The DD statement must define the data set to be updated. If you are going to accept the source module into an alternate distribution library, the DD statement must define that library.

If service SYSMODs change the DISTLIB of source modules, information about which DISTLIB subentries SMP should change for each affected SRC entry on the ACDS prior to ACCEPT processing should be provided with the SYSMOD. You must provide a ++JCLIN modification control statement to change entries during APPLY processing.

The DISTMOD Keyword

The DISTMOD keyword specifies the ddname of the distribution library that is to be updated during ACCEPT processing by link editing the module produced from assembling the source module. See 'APPLY Processing' and ACCEPT Processing' in Chapter 3 for a complete description of how this keyword is used. For compatibility with previous releases of SMP, an alternate name for DISTMOD is DISTOBJ.

The RMID Keyword

The RMID keyword specifies the SYSMOD-ID of the last PTF that replaced the source module when that PTF is incorporated into a service updated function SYSMOD. See 'Service Updated Function SYSMODs' in this chapter for a complete description of this operand.

The SSI Keyword

The SSI keyword specifies the SSI information that SMP is to place in the directory entry for the module in the target system library when the SYSMOD is applied and the distribution library when the SYSMOD is accepted. See 'APPLY Processing' in Chapter 3 for a complete description of how this operand is used.

The SYSLIB Keyword

The SYSLIB keyword specifies the ddname of the target system library that contains the source module. You must supply a JCL DD statement whose ddname is the same as the SYSLIB operand when you apply the source replacement into the target system library. The DD statement must define the data set to be updated on your target system.

The VERSION Keyword

The VERSION keyword is used in the same manner as the VERSION keyword on the ++VER modification control statement. When it is specified on a ++SRC modification control statement, it overrides the VERSION keyword specified on the ++VER modification control statement.

The ++MACUPD Modification Control Statement

The ++MACUPD modification control statement describes a macro that is being updated in the target system and the distribution libraries. For compatibility with previous releases of SMP, ++UPDTE is an alternate name for ++MACUPD. This modification control statement is not allowed in a function SYSMOD. The operands are similar to the ++MAC modification control statement operands and are described under that topic.

Macro update text consists of only those lines of text that have been changed, added, or deleted from the base version or the last replacement. Most IBM PTFs contain cumulative changes to the base version; that is, each successive PTF contains the macro update text from the previous PTF plus new changes.

The ++SRCUPD Modification Control Statement

The ++SRCUPD modification control statement describes a source module that is being updated in the target system and the distribution libraries. This modification control statement is not allowed in a function SYSMOD. The operands are similar to the ++SRC modification control statement operands and are described under that topic.

Source update text consists only of those lines of text that have been changed, added, or deleted from the base version or the last replacement. Most IBM PTFs contain cumulative changes to the base version; that is, each successive PTF contains the source update text from the previous PTF plus new changes.

The ++ZAP Modification Control Statement

The ++ZAP modification control statement describes a module that is being updated in the target system and the distribution libraries. This modification control statement cannot be specified in a function SYSMOD and is seldom present in a PTF. The operands are similar to the ++MOD modification control statement operands and are described under that topic.

This modification contains IMASPZAP control statements that are processed by the IMASPZAP program. The modification may also contain an EXPAND linkage editor control statement.

Packaging Techniques for SYSMODs

There are three techniques for packaging SYSMODs: inline, indirect library, and relative file, as described in the following three topics. A SYSMOD can be constructed using more than one technique.

Inline Packaging Technique

With the inline technique, the entire SYSMOD data is present in a single package. The element data and any JCLIN data for the SYSMOD immediately follow the associated element and ++JCLIN modification control statements. This is the only method used for elements that are updated rather than replaced. When you receive a SYSMOD packaged using this technique, SMP writes the entire SYSMOD to the PTS data set as an MCS entry. During subsequent processing of the SYSMOD by APPLY and ACCEPT, SMP reads the element data from the MCS entry and writes the data to the appropriate work data set prior to invoking the utility programs to update the target system and distribution libraries. Most IBM PTFs are packaged using this technique.

Indirect Library Technique

With the indirect library technique, SYSMODs are packaged with element and JCLIN data in physically different files from the modification control statements. Each indirect library contains one or more members. The individual ++MAC, ++SRC, and ++JCLIN modification control statements specify the TXLIB keyword, and the ++MOD modification control statements specify either the TXLIB or LKLIB keyword. These operands are used during APPLY and ACCEPT processing to locate the libraries by means of JCL DD statements with corresponding ddnames. It is your responsibility to ensure that the indirect libraries are on direct access storage devices prior to executing APPLY or ACCEPT and to provide the DD statements necessary to access the libraries.

The following is an example of a SYSMOD packaged with the indirect library technique:

```
++FUNCTION(FAA1000).
++VER(Z038).
++JCLIN TXLIB(AAJCLIN).
++MAC(AAQRST) TXLIB(AAMACLIB) DISTLIB(AOSMACAA).
++MAC(AAWXYZ) TXLIB(AAMACLIB) DISTLIB(AOSMACAA).
++MOD(AAABCD01) LKLIB(AAMODLIB) DISTLIB(AOSMODAA).
++MOD(AAABCD02) LKLIB(AAMODLIB) DISTLIB(AOSMODAA).
```

After you have loaded the libraries to direct access storage, you must provide DD statements when executing the APPLY function. For example:

```
//AAJCLIN DD DSN=FAA1000.AAJCLIN,VOL=SER=PACK01,
//        UNIT=SYSDA,DISP=OLD
//AAMACLIB DD DSN=FAA1000.AAMACLIB,VOL=SER=PACK01,
//        UNIT=SYSDA,DISP=OLD
//AAMODLIB DD DSN=FAA1000.AAMODLIB,VOL=SER=PACK01,
//        UNIT=SYSDA,DISP=OLD
```

The advantages of this technique over the inline packaging technique are improved performance, since the data does not have to be moved to work data sets during the APPLY and ACCEPT functions, and less space is needed for the PTS.

Relative File Technique

The relative file technique is similar to the indirect library technique in that the element and JCLIN data is packaged in files separate from the modification control statements. With this technique, the FILES keyword is specified on the header modification control statement and the RELFILE keyword is specified on each element and ++JCLIN modification control statement whose data is in a separate file. The FILES keyword specifies the number of files that are associated with the SYSMOD. The RELFILE keyword specifies the relative file number, with respect to other files associated with the SYSMOD, of the file containing the element or JCLIN data.

SMP loads the files onto direct access storage when the SYSMOD is received. You must include the SMPTLIB DD statement when you invoke SMP to process SYSMODs packaged with this technique. The SMPTLIB DD card defines the direct access storage devices to be used to load the files. You may specify up to five (5) volumes on the SMPTLIB DD card. SMP will allocate the space needed for each library on one of the devices specified unless you allocate the data sets before executing the RECEIVE function. These files will be accessed during APPLY and ACCEPT processing similar to that for SYSMODs with unloaded libraries.

This packaging technique permits multiple SYSMODs on the same physical tape. All SYSMOD modification control statements are contained in a single file with their related text files following in the sequence specified by the order of the SYSMODs and, within each SYSMOD, by the RELFILE operands on the element modification control statements. Figure 4 is an example of multiple SYSMODs packaged on a single tape. SMP processing calculates the absolute file number of each file that is loaded during RECEIVE processing although some of the SYSMODs may not be selected or processed.

Tapes containing SYSMODs packaged with this technique must have standard labels. The files containing the unloaded PDSs must have a DSNAME of the form:

    iiiiiii.Fnnnn

where "iiiiiii" is the SYSMOD-ID of the owning SYSMOD and "nnnn" is a one-to four-digit file number corresponding to the value in the associated element or ++JCLIN modification control statement, with no leading zeroes.

See 'RECEIVE Processing' in Chapter 3 for a further description of how relative files are processed.

FILE	DATA
1	++FUNCTION(GBB3100) FILES(3). ++VER(Z038). ++JCLIN RELFILE(1). ++MOD(A) DISTLIB(ABBDLIB) RELFILE(2). ++MOD(B) DISTLIB(ABBDLIB) RELFILE(2). ++MAC(X) DISTLIB(ABBMACS) RELFILE(3). . . . ++FUNCTION(EBB3101) FILES(3). ++VER(Z038) FMID(GBB3100). ++JCLIN RELFILE(1). ++MOD(A) DISTLIB(ABBDLIB) RELFILE(2). ++MOD(C) DISTLIB(ABBDLIB) RELFILE(2). ++MAC(Y) DISTLIB(ABBMACS) RELFILE(3). . . .
2	Unloaded PDS containing member GBB3100, which is JCLIN data for function GBB3100
3	Unloaded PDS containing modules A and B for function GBB3100
4	Unloaded PDS containing macro X for function GBB3100
5	Unloaded PDS containing member EBB3101, which is JCLIN data for function EBB3101
6	Unloaded PDS containing modules A and C for function EBB3101
7	Unloaded PDS containing macro Y for function EBB3101

Figure 4. Physical Organization of Relative File Tape

Service Updated Function SYSMODs

Function SYSMODs may be periodically repackaged to incorporate existing service modifications into the function. The result is called a service updated function SYSMOD. You may choose to reapply and reaccept a function SYSMOD that has been service updated to bring that function up to a higher service level than what you currently have in your target system and distribution libraries.

Rules for Integrating Service SYSMODs

When a function SYSMOD is service updated, the original modification control statements may be changed and new ones added to the SYSMOD, depending on the elements that have been modified since the function was first packaged.

The modifications performed to service update a function SYSMOD are as follows:

- The SYSMOD-IDs of all service SYSMODs integrated into the function SYSMOD are placed in the SUP operand list of the ++VER modification control statement.

- All SYSMOD-IDs from the SUP operand lists on the ++VER modification control statements from integrated service SYSMODs are placed in the SUP operand list of the ++VER modification control statement for the function SYSMOD. No duplicate SYSMOD-IDs will be present in the SUP operand list.

- The ++IF modification control statements from integrated service SYSMODs are included in the function SYSMODs. For each unique FMID operand, the REQ operand list values are placed into a combined ++IF modification control statement; duplicates are eliminated.

- The JCLIN data from integrated service SYSMODs is combined with that from the original function SYSMOD. The merge is done according to service order so that the most recent JCLIN data is the last in the combined data.

- All elements that have been deleted by the inclusion of the DELETE operand on an element modification control statement from an integrated service SYSMOD are deleted from the function SYSMOD. If you reapply and reaccept the service updated function SYSMOD and have not applied and accepted the integrated service SYSMODs that deleted those elements, you might have to delete some elements from the target system and distribution libraries and the element entries from the CDS and ACDS.

- Load module names from the LMOD operand lists of ++MOD modification control statements from integrated service SYSMODs are placed in LMOD operands on the corresponding ++MOD modification control statements in the function SYSMOD; duplicate names are eliminated.

- Element modification control statements for elements added by integrated service SYSMODS are added to the function SYSMOD.

- SYSMOD-IDs from the VERSION operand lists of ++VER modification control statements from integrated service SYSMODs are placed in the VERSION operand list on the ++VER modification control statement for the function SYSMOD; duplicates are eliminated.

- SYSMOD-IDs from the VERSION operand lists of element modification control statements from integrated service SYSMODs are placed in the VERSION operand list on the element modification control statements for the function SYSMOD; duplicates are eliminated.

- For each element that has been modified by integrated service SYSMODs, the SYSMOD-ID of the service SYSMOD that last replaced the element is placed in the element modification control statement as the value of the RMID operand.

  When a service updated function SYSMOD is applied or accepted, the RMID subentry of the element entry of elements selected from the SYSMOD for replacement is replaced with the value from the RMID operand, if it is present.

See 'APPLY Processing' in Chapter 3 for a description of how a service updated function SYSMOD is applied to a target system.

Sample Sevice Updated Function SYSMOD

The following shows a function SYSMOD and four PTFs that service elements within that function SYSMOD.

```
++FUNCTION(FXX4101) FILES(3).
++VER(Z038).
++JCLIN RELFILE(1).
++MAC(IXXKLTD) DISTLIB(AXXMACLB) RELFILE(2).
++MAC(IXXLQIQ) DISTLIB(AXXMACLB) RELFILE(2).
++MAC(IXXMWTS) DISTLIB(AXXMACLB) RELFILE(2).
++MAC(IXXNJDW) DISTLIB(AXXMACLB) RELFILE(2).
++MOD(IXXJWMDW) DISTLIB(AOS98) RELFILE(3).
++MOD(IXXJWXDC) DISTLIB(AOS98) RELFILE(3).
++MOD(IXXJWYCV) DISTLIB(AOS98) RELFILE(3).
++MOD(IXXJWYD1) DISTLIB(AOS98) RELFILE(3).
++MOD(IXXJWYD2) DISTLIB(AOS98) RELFILE(3).
++MOD(IXXJWYD3) DISTLIB(AOS98) RELFILE(3).


++PTF(UZ13579).
++VER(Z038) FMID(FXX4101) SUP(AZ11335).
++MACUPD(IXXKLTD) DISTLIB(AXXMACLB).
++MOD(IXXJWYCV) DISTLIB(AOS98).


++PTF(UZ13601).
++VER(Z038) FMID(FXX4101) PRE(UZ13579) SUP(AZ11442).
++IF FMID(FXX4102) THEN REQ(UZ13607).
++MOD(IXXJWYCV) DISTLIB(AOS98).
++MOD(IXXJWYD1) DISTLIB(AOS98).
++MOD(IXXJWYD2) DISTLIB(AOS98).


++PTF(UZ13613).
++VER(Z038) FMID(FXX4101) PRE(UZ13601) SUP(AZ11456).
++IF FMID(FXX4102) THEN REQ(UZ13614).
++MACUPD(IXXLQIQ) DISTLIB(AXXMACLB).
++MOD(IXXJWMDW) DISTLIB(AOS98).
++MOD(IXXJWXDC) DISTLIB(AOS98).
++MOD(IXXJWYD1) DISTLIB(AOS98).


++PTF(UZ13644).
++VER(Z038) FMID(FXX4101) PRE(UZ13613,UZ13601)
     SUP(AZ11487).
++IF FMID(FXX4102) THEN REQ(UZ13645).
++IF FMID(FXX4103) THEN REQ(UZ13646).
++MOD(IXXJWYCV) DISTLIB(AOS98) VERSION(FXX4102).
++MOD(IXXJWYD3) DISTLIB(AOS98).
```

After integrating the four PTFs, the service updated function SYSMOD would appear as follows.

```
++FUNCTION(FXX4101) FILES(3).
++VER(Z038) SUP(AZ11335,AZ11442,AZ11456,AZ11487,
   UZ13579,UZ13601,UZ13613,UZ13644).
++JCLIN RELFILE(1).
++IF FMID(FXX4102) THEN REQ(UZ13607,UZ13614,UZ13645).
++IF FMID(FXX4103) THEN REQ(UZ13646).
++MAC(IXXKLTD) DISTLIB(AXXMACLB) RELFILE(2).
     RMID(UZ13579).
++MAC(IXXLQIQ) DISTLIB(AXXMACLB) RELFILE(2)
     RMID(UZ13613).
++MAC(IXXMWTS) DISTLIB(AXXMACLB) RELFILE(2).
++MAC(IXXNJDW) DISTLIB(AXXMACLB) RELFILE(2).
++MOD(IXXJWMDW) DISTLIB(AOS98) RELFILE(3)
     RMID(UZ13613).
++MOD(IXXJWXDC) DISTLIB(AOS98) RELFILE(3)
     RMID(UZ13613).
++MOD(IXXJWYCV) DISTLIB(AOS98) RELFILE(3)
     VERSION(FXX4102) RMID(UZ13644).
++MOD(IXXJWYD1) DISTLIB(AOS98) RELFILE(3)
     RMID(UZ13613).
++MOD(IXXJWYD2) DISTLIB(AOS98) RELFILE(3).
     RMID(UZ13601).
++MOD(IXXJWYD3) DISTLIB(AOS98) RELFILE(3)
     RMID(UZ13644).
```

SYSMOD Construction Techniques

The only elements allowed in a system modification package are those belonging to one function. The owning function is identified by the operand value of the ++FUNCTION modification control statement for function packages or the value of the FMID operand on ++VER modification control statements for service packages. Furthermore, all service packages must identify the owning function of the elements in the package. These restrictions remove ambiguity with respect to determining function ownership.

To demonstrate some of the problems that the SYSMOD formulation technique solves, it is necessary to understand relationships of functions and their associated elements. The following figure shows functions and module relationships:

```
                      MODULES
               *----------------*
FUNCTIONS    | A | B | C | D |        Assume:
*----------||---|---|---|---|        UZ89900 is superior
| UZ89700 || x | x | x | x |        to UZ89800;
|----------||---|---|---|---|        UZ89800 is superior
| UZ89800 || x | x | x |   |        to UZ89700
|----------||---|---|---|---|
| UZ89900 ||   | x | x | x |        "x" means that the module
*---------------------------*        is present in the function
```

The examples that follow assume that function UZ89700 must be present for either function UZ89800 or UZ89900 to be applicable. Either UZ89800 or UZ89900 can be present without the other but, if both are present, function UZ89900 is superior to function UZ89800.

Previous Packaging Methods

With previous releases of SMP, if one or more APARs were fixed that encompassed all four modules in all three functions, two PTF packaging methods were available to modify all possible environments; the case method and the corequisite PTF method.

The Case Method

With the case method, four possible environments might exist on a system. They are as follows:

- UZ89700 only
- UZ89700 and UZ89800
- UZ89700 and UZ89900
- UZ89700, UZ89800, and UZ89900

The PTFs necessary to fix all the possible environments were constructed as follows:

```
++PTF(UZ13001).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900).
++MOD(A)                 /* FOR UZ89700 */.
++MOD(B)                 /* FOR UZ89700 */.
++MOD(C)                 /* FOR UZ89700 */.
++MOD(D)                 /* FOR UZ89700 */.

++PTF(UZ13002).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900).
++MOD(A)                 /* FOR UZ89800 */.
++MOD(B)                 /* FOR UZ89800 */.
++MOD(C)                 /* FOR UZ89800 */.
++MOD(D)                 /* FOR UZ89700 */.

++PTF(UZ13003).
++VER(Z037) PRE(UZ89700,UZ89900) NPRE(UZ89800).
++MOD(A)                 /* FOR UZ89700 */.
++MOD(B)                 /* FOR UZ89900 */.
++MOD(C)                 /* FOR UZ89900 */.
++MOD(D)                 /* FOR UZ89900 */.

++PTF(UZ13004).
++VER(Z037) PRE(UZ89800,UZ89900).
++MOD(A)                 /* FOR UZ89800 */.
++MOD(B)                 /* FOR UZ89900 */.
++MOD(C)                 /* FOR UZ89900 */.
++MOD(D)                 /* FOR UZ89900 */.
```

The ownership of the modules in the last three PTFs cannot be determined from the modification control statements. Furthermore, no relationship is specified between any of the PTFs. As a result, you cannot easily determine the complete set of PTFs required to fix the APAR(s) in all environments, especially if all of the PTFs required are not being installed concurrently.

The Corequisite PTF Method

With the corequisite PTF method of packaging, each element common to more than one function is in a separate PTF. The set of PTFs is related by the specification of multiple ++VER modification control statements with REQ operands, as follows:

```
++PTF(UZ13001).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900)
    REQ(UZ13002,UZ13003).
++VER(Z037) PRE(UZ89700,UZ89900) NPRE(UZ89800)
    REQ(UZ13006).
++MOD(A)                      /* FOR UZ89700 */.

++PTF(UZ13002).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900)
    REQ(UZ13001,UZ13003).
++MOD(B)                      /* FOR UZ89700 */.
++MOD(C)                      /* FOR UZ89700 */.

++PTF(UZ13003).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900)
    REQ(UZ13001,UZ13002).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900)
    REQ(UZ13004,UZ13005).
++MOD(D)                      /* FOR UZ89700 */.

++PTF(UZ13004).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900)
    REQ(UZ13003,UZ13005).
++VER(Z037) PRE(UZ89700,UZ89800,UZ89900)
    REQ(UZ13006).
++MOD(A)                      /* FOR UZ89800 */.

++PTF(UZ13005).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900)
    REQ(UZ13003,UZ13004).
++MOD(B)                      /* FOR UZ89800 */.
++MOD(C)                      /* FOR UZ89800 */.

++PTF(UZ13006).
++VER(Z037) PRE(UZ89700,UZ89900) NPRE(UZ89800)
    REQ(UZ13001).
++VER(Z037) PRE(UZ89700,UZ89800,UZ89900)
    REQ(UZ13004).
++MOD(B)                      /* FOR UZ89900 */.
++MOD(C)                      /* FOR UZ89900 */.
++MOD(D)                      /* FOR UZ89900 */.
```

This approach can result in more PTFs than with the case method; however, each PTF conveys enough information in the REQ operand field to enable you to find the requisite PTFs, which in turn have requisites and eventually complete the set information. Also notice that modules B and C can be packaged together by function because for any given environment, they are always together for the same function.

Although the corequisite PTF method has advantages over the case method, neither method minimizes the number of PTFs necessary to fix a set of APARs that encompass all three functions.

Furthermore, if another function package is released that contains any of the elements of any of the three previous functions, any available service must be explicitly superseded by the new function package. This prevents regression of the new function if you attempt to apply service after the new function is applied to your system.

The SYSMOD Formulation Method

The following example shows how you could construct the three function SYSMOD packages with the SYSMOD formulation method:

```
++FUNCTION(UZ89700).
++VER(Z038).
++MOD(A)                    /* FOR UZ89700 */.
++MOD(B)                    /* FOR UZ89700 */.
++MOD(C)                    /* FOR UZ89700 */.
++MOD(D)                    /* FOR UZ89700 */.

++FUNCTION(UZ89800).
++VER(Z038) FMID(UZ89700).
++MOD(A)                    /* FOR UZ89800 */.
++MOD(B)                    /* FOR UZ89800 */.
++MOD(C)                    /* FOR UZ89800 */.

++FUNCTION(UZ89900).
++VER(Z038) FMID(UZ89700) VERSION(UZ89800).
++MOD(B)                    /* FOR UZ89900 */.
++MOD(C)                    /* FOR UZ89900 */.
++MOD(D)                    /* FOR UZ89900 */.
```

The following example shows how you would construct the PTFs required to fix APARs spanning all four modules in three functions, as previously discussed in the case method and corequisite method:

```
++PTF(UZ13001).
++VER(Z038) FMID(UZ89700).
++IF FMID(UZ89800) THEN REQ(UZ13002).
++IF FMID(UZ89900) THEN REQ(UZ13003).
++MOD(A)                 /* FOR UZ89700 */.
++MOD(B)                 /* FOR UZ89700 */.
++MOD(C)                 /* FOR UZ89700 */.
++MOD(D)                 /* FOR UZ89700 */.

++PTF(UZ13002).
++VER(Z038) FMID(UZ89800) REQ(UZ13001).
++IF FMID(UZ89900) THEN REQ(UZ13003).
++MOD(A)                 /* FOR UZ89800 */.
++MOD(B)                 /* FOR UZ89800 */.
++MOD(C)                 /* FOR UZ89800 */.

++PTF(UZ13003).
++VER(Z038) FMID(UZ89900) REQ(UZ13001).
++IF FMID(UZ89800) THEN REQ(UZ13002).
++MOD(B)                 /* FOR UZ89900 */.
++MOD(C)                 /* FOR UZ89900 */.
++MOD(D)                 /* FOR UZ89900 */.
```

Only three PTFs, the minimum possible, are required to service all the elements in all the functions. Each PTF has information that refers to the other PTFs in the set. The ++IF modification control statements are processed only when the function SYSMOD specified is present.

Combined Packaging For Compatibility

You can construct a set of PTFs that will service the above functions and can be processed by both this and previous releases of SMP. The approach that you must use combines the corequisite method with the SYSMOD formulation method as follows:

```
++PTF(UZ13001).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900)
     REQ(UZ13002,UZ13003).
++VER(Z037) PRE(UZ89700,UZ89900) NPRE(UZ89800)
     REQ(UZ13006).
++VER(Z038) FMID(UZ89700) REQ(UZ13002,UZ13003).
++IF FMID(UZ89800) THEN REQ(UZ13004,UZ13005).
++IF FMID(UZ89900) THEN REQ(UZ13006).
++MOD(A)                 /* FOR UZ89700 */.
```

```
++PTF(UZ13002).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900)
     REQ(UZ13001,UZ13003).
++VER(Z038) FMID(UZ89700) REQ(UZ13001,UZ13003).
++IF FMID(UZ89800) THEN REQ(UZ13004,UZ13005).
++IF FMID(UZ89900) THEN REQ(UZ13006).
++MOD(B)                  /* FOR UZ89700 */.
++MOD(C)                  /* FOR UZ89700 */.

++PTF(UZ13003).
++VER(Z037) PRE(UZ89700) NPRE(UZ89800,UZ89900)
     REQ(UZ13001,UZ13002).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900)
     REQ(UZ13004,UZ13005).
++VER(Z038) FMID(UZ89700) REQ(UZ13001,UZ13002).
++IF FMID(UZ89800) THEN REQ(UZ13004,UZ13005).
++IF FMID(UZ89900) THEN REQ(UZ13006).
++MOD(D)                  /* FOR UZ89700 */.

++PTF(UZ13004).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900)
     REQ(UZ13003,UZ13005).
++VER(Z037) PRE(UZ89700,UZ89800,UZ89900)
     REQ(UZ13006).
++VER(Z038) FMID(UZ89800) REQ(UZ13005,UZ13003).
++IF FMID(UZ89900) THEN REQ(UZ13006).
++MOD(A)                  /* FOR UZ89800 */.

++PTF(UZ13005).
++VER(Z037) PRE(UZ89700,UZ89800) NPRE(UZ89900)
     REQ(UZ13003,UZ13004).
++VER(Z038) FMID(UZ89800) REQ(UZ13004,UZ13003).
++IF FMID(UZ89900) THEN REQ(UZ13006).
++MOD(B)                  /* FOR UZ89800 */.
++MOD(C)                  /* FOR UZ89800 */.

++PTF(UZ13006).
++VER(Z037) PRE(UZ89700,UZ89900) NPRE(UZ89800)
     REQ(UZ13001).
++VER(Z037) PRE(UZ89700,UZ89800,UZ89900)
     REQ(UZ13004).
++VER(Z038) FMID(UZ89900) REQ(UZ13001).
++IF FMID(UZ89800) THEN REQ(UZ13004).
++MOD(B)                  /* FOR UZ89900 */.
++MOD(C)                  /* FOR UZ89900 */.
++MOD(D)                  /* FOR UZ89900 */.
```

When this set of PTFs is processed by previous versions of SMP, the new operands and the ++IF modification control statements are ignored. If you are using a previous version of SMP, the CDS SYSTEM entry must not have "Z038" as the SREL subentry value. If you are using this version of SMP, the PTS SYSTEM entry and the CDS SYSTEM entry must not contain "Z037" as an SREL subentry value so that all ++VER modification control statements with "Z037" will be ignored.

Chapter 3: SMP Processing

Processing of system modifications (SYSMODs) is controlled by the SMP control statements that you specify. The purpose of this chapter is to explain the processing that takes place for each of the major control statements, and to describe the options and restrictions that pertain to each one. The information provided in this chapter describes in detail what SMP does internally with the various operands that you might specify on each of the control statements.

| The major SMP functions are:

- RECEIVE - places SYSMODs into the SMP PTF Temporary Store Data Set (PTS) for subsequent processing by REJECT, APPLY, RESTORE and ACCEPT.
- REJECT - deletes SYSMODs from the PTS data set.
- APPLY - places SYSMODs into the target system libraries.
- RESTORE - removes SYSMODs processed by APPLY from target system libraries.
- ACCEPT - places SYSMODs into the distribution libraries (DLIBs) or permanent user libraries. Once ACCEPT processing completes, SMP cannot remove the SYSMOD.
- JCLIN - reads in Stage I output from system generation (SYSGEN) or similar job step JCL to create or update the SMP Control Data Set (CDS).
- UCLIN - updates, adds, or deletes entries on the ACDS, ACRQ, CDS, CRQ, MTS, PTS, SCDS, or STS data sets.
- | RETRY - provides a recovery facility for dataset out of space conditions during APPLY, ACCEPT, and RESTORE.

RECEIVE Processing

RECEIVE processing places SYSMODs in the PTS for subsequent processing by the REJECT, APPLY, RESTORE, and ACCEPT functions. Thus RECEIVE processing must be invoked before any other SMP processing can be performed for a SYSMOD.

You can control the SYSMODs that are received by using either the SELECT or EXCLUDE keywords on the RECEIVE control statement. RECEIVE processing can be invoked any number of times in the same SMP job step, which could be the case if you wished to select or exclude different SYSMODs in each invocation.

If the SYSMODs that you wish to receive are packaged using relative files (described in "Relative File Packaging Techniques" in Chapter 2), RECEIVE processing allocates data sets on direct access storage devices described by the SMPTLIB DD statement and loads the members from the relative files to those data sets.

The PTS Data Set

The PTS serves as a staging data set for SYSMODs, and contains three types of entries: a SYSTEM entry, Modification Control Statement (MCS) entries, and SYSMOD entries. The SYSTEM entry is used to control which SYSMODs are received, and specifies the environment of the system(s) being maintained. For each SYSMOD received, an MCS entry and a SYSMOD entry are created.

The entries on the PTS are described in Chapter 7 under the "LIST Control Statement," and are illustrated in Figure 5.

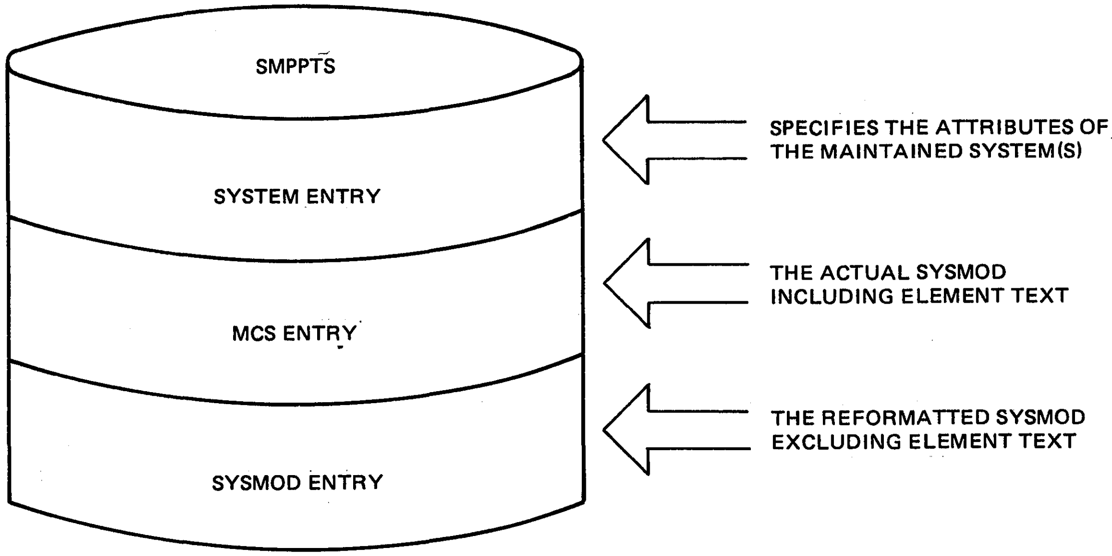

Figure 5. The PTS Entries

Using the PTS enables you to receive service or function SYSMODs that apply to functions that have not been received. When those functions are subsequently installed, SMP examines the SMP Conditional Requisite Queue Data Set (CRQ) to see if there are requisite SYSMODs, uses the PTS to see if the requisite service or function SYSMODs were saved and, if present, installs them at the same time. Thus, you do not need to save or research SYSMODs required when new functions are installed, facilitating pre-installation planning.

You may specify the functions that are not currently installed for which you want applicable service SYSMODs saved on the PTS. To do this, you either place the SYSMOD-ID of the function SYSMOD that has not been received in the PTS as an FMID subentry of the PTS SYSTEM entry using the UCL SYS statement, or you may specify BYPASS(FMID) on the RECEIVE control statement. The former method is recommended.

Until the new function SYSMODs are installed on the target system or distribution libraries, the saved SYSMODs are ignored by APPLY or ACCEPT processing.

The PTS MCS Entry

The complete SYSMOD, including any modification text for modules, macros, and source modules, is stored without change as an MCS entry. Because the SYSMODs are stored as separate, distinct members, you can receive multiple modifications against the same element. You can use IEBPTPCH or any comparable utility program to print or punch the PTS MCS entries.

For SYSMODs that contain elements or inline JCLIN data in relative files, LKLIB, or TXLIB data sets, the MCS entries do not contain the modification text. The TXLIB data sets are specified using the TXLIB keyword on the element modification control statement. You can use IEBPTPCH or any comparable utility program to print or punch the modification text from the TXLIB or SMPTLIB data sets containing fixed length record data. The LKLIB data sets are specified using the LKLIB keyword on the ++MOD modification control statement. The members of these data sets, as well as SMPTLIB data sets loaded for modules, are load modules in undefined record format.

The PTS SYSMOD Entry

A SYSMOD entry, similar to those defined for the CDS and ACDS, is created for each SYSMOD received. In the SYSMOD entry, the data from the modification control statements is reformatted for SMP processing. Because of this reformatting, other SMP functions do not have to read the MCS entry for a SYSMOD to find out which elements are in the SYSMOD and what information is contained in the ++VER modification control statements. The SYSMOD entries do not include the text of the modification or the information from the ++IF modification control statements.

The PTS SYSTEM Entry

Receiving of SYSMODs can be controlled so that SYSMODs that do not apply to your system are ignored. This is done using the PTS SYSTEM entry, which specifies the environment of the system or systems being maintained.

The SYSTEM entry contains at least one system release (SREL) subentry and any number of function modification identifier (FMID) subentries. These subentries are comparable to the SREL and FMID operands on the ++VER modification control statements of the SYSMODs being processed. In order to be received, a SYSMOD must have at least one SREL operand on a ++VER modification control statement that matches one of the SREL subentries in the PTS SYSTEM entry.

Function SYSMODs do not require an FMID operand. They are received when an SREL operand on a ++VER modification control statement matches one of the SREL subentries in the PTS SYSTEM entry. If FMID operands are present, they are ignored.

Service SYSMODs must have an FMID operand. If the SREL operand matches, as described above, but none of the ++VER modification control statements with matching SRELs specifies an FMID operand that matches an FMID subentry in the PTS, the SYSMOD is not received. Figure 6 illustrates the SREL and FMID comparisons performed to control which SYSMODs are received.

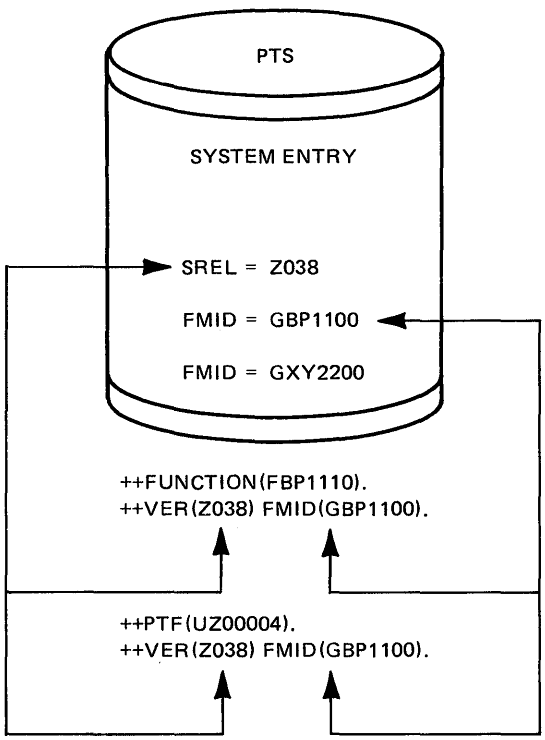

Figure 6. Controlling the Receiving of SYSMODS

You can override the FMID checking by specifying BYPASS(FMID) on the RECEIVE control statement. If BYPASS(FMID) is specified, all of the SYSMODs with matching SRELs in ++VER modification control statements are received unless already received.

Function and service SYSMODs can be received regardless of the state of the CDS. For installations with more than one version of an operating system, this allows one RECEIVE operation to be valid for several system versions.

Creating the PTS SYSTEM Entry

The UCLIN function creates the SYSTEM entry on the PTS. Using the UCL SYS statement, you can specify the FMIDs and SRELs to be placed in the SYSTEM entry. See "UCLIN Processing" in this chapter for further information.

Updating the PTS SYSTEM Entry

When a function SYSMOD is received, the PTS SYSTEM entry is updated to include the new SYSMOD-ID as an FMID subentry.

Syntax and Operand Validity Checking

SMP checks all modification control statements for selected SYSMODs for correct syntax and lists the statements on the SMPOUT data set. Each statement is checked and listed even when a syntax error or validity check error was encountered on a previous modification control statement. If an error is found, the SYSMOD is not received, and SMP issues a message describing the error.

SMP performs the following validity checking:

- If the SYSMOD being processed is a ++APAR, ++PTF, or ++USERMOD (service SYSMODs), the FMID operand must be specified on every ++VER modification control statement with an SREL operand value that matches an SREL subentry in the PTS SYSTEM entry. If the FMID operand is not specified and the SREL value(s) are not present in the PTS SYSTEM entry, the ++VER modification control statement is ignored and validity checking is not continued.

- Service SYSMODs cannot specify the DELETE or NPRE keywords.

- SMP compares the SREL and FMID operand values, as pairs, with the SREL and FMID operand values, as pairs, from other ++VER modification control statements to ensure that the same FMID operand value is not specified with the same SREL operand value more than once. The following example shows a SYSMOD construction error:

```
++PTF(UZ00005).
++VER(Z038) FMID(GBP1502).
++VER(Z038) FMID(GBP1502).
```

- A function cannot be a base level function for one system release and a dependent level function for a different release at the same time. Therefore, for function SYSMODs, if the FMID operand is specified on one ++VER modification control statement, then it must be present on all of the other ++VER modification control statements, as shown in the following erroneous SYSMOD construction:

```
++FUNCTION(HVT1403).
++VER(Z039).
++VER(Z038) FMID(HVT1303).
```

- SMP checks each ++VER modification control statement for duplication of operand values. If the same SYSMOD-ID is specified more than once in the same operand list or is specified in more than one operand list, the ++VER modification control statement is considered to be incorrect. In the following example, the same SYSMOD-ID was erroneously specified twice in the PRE operand list:

```
++PTF(UZ00079).
++VER(Z038) FMID(GVT1202) PRE(UZ00010,UZ00010).
```

  There is an exception to this rule: a SYSMOD-ID that is specified in the VERSION operand list can also be specified in any one of the other operand lists except for the FMID operand.

- SMP checks the FMID operand values on every ++IF modification control statement to ensure that the SYSMOD-ID specified is not the same as the FMID operand value specified in the associated ++VER modification control statement or the SYSMOD-ID specified in the header modification control statement. The following example shows a SYSMOD with an incorrect ++IF modification control statement specification:

```
++PTF(UZ00079).
++VER(Z038) FMID(GVT1202) PRE(UZ00010).
++IF FMID(GVT1202) THEN REQ(UZ00021).
```

The SYSMOD is considered to be eligible to be received if it is error-free. If the SYSMOD is eligible, the contents of each ++VER modification control statement are placed in the PTS SYSMOD entry. Thus, SYSMODs that apply to more than one system release can be received even though an SREL was not present in the PTS SYSTEM entry for a given ++VER modification control statement. If such an SREL value is subsequently added to the PTS SYSTEM entry, the ++VER modification control statements containing that SREL can be processed by the APPLY and ACCEPT functions for that system.

Processing of Relative Files

Library Loading

If the SYSMODs that you are receiving were constructed using the relative file packaging technique described in "Relative File Technique" in Chapter 2, you must use the SMPTLIB DD statement to specify at least one direct access storage device volume to hold the loaded partitioned data sets. Up to five volumes can be used.

RECEIVE processing dynamically allocates storage on a volume for each partitioned data set being loaded. This allocation is accomplished using the DADSM SVC (SVC 32). Using the UCL SYS statement, you define the space allocation parameters to be used by SMP in the DSSPACE subentry of the PTS SYSTEM entry.

You may specify high level data set name qualifiers for these loaded partitioned data sets by creating a DSPREFIX subentry in the PTS SYSTEM entry using the UCL SYS statement. If DSPREFIX is not specified in the SYSTEM entry, no high level data set name qualifiers are used. If the DSPREFIX subentry is present in the SYSTEM entry, the value is placed in the SYSMOD entry so that the REJECT, APPLY, RESTORE, and ACCEPT functions can construct the appropriate data set names (DSNAME) for the libraries. The DSNAME is in the format "dsprefix.sysmodid.Ffile#".

If the same data set already exists on one of the SMPTLIB volumes, it is used rather than deleting and reallocating. This permits you to preallocate and, optionally, catalog these data sets.

The ERROR indicator is set in the SYSMOD entry before the files are loaded, but after the SYSMOD and MCS entries are created. Members from the relative files are loaded during RECEIVE processing onto direct access storage if the SYSMODs are being received. This process is done using IEBCOPY. Each element modification control statement included in the SYSMOD for a specific file is selectively copied. Every alias specified in the DALIAS, MALIAS, and TALIAS operands is also selectively copied. This selective copying ensures that the contents of the unloaded partitioned data sets are correct. If IEBCOPY returns a value as a return code that is higher than the SMP default value or the COPYRC value in the PTS SYSTEM entry, if present, the SYSMOD is terminated as described in this chapter under "Termination of SYSMODs with Relative Files." If the loading of the libraries is successful, the ERROR indicator in the SYSMOD entry is reset and any space unused in the libraries is released.

Termination of SYSMODs with Relative Files

If, during the allocation or loading of any libraries for a SYSMOD, an invoked utility function encounters an error condition that causes termination of the SYSMOD, then any libraries already loaded or allocated for that SYSMOD are deleted. This occurs even if you have preallocated the libraries. The SYSMOD and MCS entries for the SYSMOD are also deleted.

If you are processing SYSMODs that were constructed using relative files and an abend occurs during the load of the partitioned data sets onto direct access storage, SMP does not set off the ERROR indicator in the SYSMOD entry. If you try to receive the SYSMOD again, SMP recognizes that the ERROR indicator is set and deletes the MCS and SYSMOD entries before it creates new entries for the SYSMOD.

Inline JCLIN

Job Control Language (JCL) input data can immediately follow a ++JCLIN modification control statement or can reside in a library that SMP can access during APPLY processing. If the input data is in a library, the library can be loaded external to SMP using IEBCOPY, or, if RELFILE is indicated, handled as described in "Library Loading" earlier in this chapter. SMP accesses these libraries in the same way as it accesses libraries specified using the TXLIB or RELFILE keywords on the element modification control statements. The name of the member that contains the JCL data must be the same as the SYSMOD-ID of the associated SYSMOD.

Text Restriction for ++SRCUPD and ++MACUPD SYSMODs

The modification text for ++SRCUPD, ++MACUPD, and ++UPDTE SYSMODs must contain a ./ CHANGE control statement preceding the text and can contain a ./ ENDUP control statement as the last input statement. All other IEBUPDTE control statements are considered invalid, and, when encountered, an error message is issued and the SYSMOD is not received. If the member name specified in the ./ CHANGE control statement does not match the name in the modification control statement, an error message is issued and the SYSMOD is not received.

Reprocessing Received SYSMODs

If a SYSMOD was received successfully, it exists in the PTS without the ERROR indicator set in the SYSMOD entry. It cannot be received again. If you want to make changes to a SYSMOD, you should reject it using the REJECT control statement, modify the contents, and receive the modified SYSMOD. You should not attempt to modify the SYSMOD or MCS entries on the PTS, and there is no UCLIN facility provided to do this. The BYPASS keyword, explained earlier in "The PTS SYSTEM Entry," cannot be used to receive SYSMODs that are already received on the PTS, nor can the SELECT keyword.

User modifications (++USERMOD) that already exist on the PTS and require modification to apply to a more current level of IBM function or service are treated in the same way. They must be rejected using the REJECT control statement, modified, and received again.

RECEIVE Output

RECEIVE processing creates a time and date-stamped record of all activity on a history log data set, SMPLOG, and produces a listing of all modification control statements and SMP messages on SMPOUT.

When running with the SELECT or EXCLUDE operands, the modification control statements for SYSMODs that were either excluded or not selected are not written to SMPOUT, and are not syntax checked by RECEIVE processing. As an exception, header modification control statements whose sysmodid operand does not appear on the first record of the header statement are written to SMPOUT. An example of this case is when comments appear between '++PTF' and the '(sysmodid)' operand, rather than specifying '++PTF(sysmodid)'.

In addition, a RECEIVE SUMMARY REPORT, summarizing the processing of SYSMODs encountered during RECEIVE processing, is produced on SMPOUT or SMPRPT, if an SMPRPT DD card is present. This report lists every SYSMOD that was processed by RECEIVE with an indication of its type and status and, if it terminated, the reason why it was not received. A sample of this report appears in Chapter 6.

Obtaining the PEMAX value

The PEMAX keyword specifies the maximum number of subentries that can exist in a single SYSMOD on the CDS, ACDS, or PTS. The default number is 500 subentries. Chapter 4 describes usage of PEMAX.

Since the CDS is not required during RECEIVE processing, a PEMAX value can be specified for the PTS SYSTEM entry using the UCLIN control statement. If neither the CDS nor the ACDS has been defined through JCL DD statements at the time that the RECEIVE function is invoked, the PEMAX value in the PTS SYSTEM entry is used, if present, or the default value is used. If the CDS and/or the ACDS is available, the PEMAX value that is the greatest of all the PEMAX values in the available SYSTEM entries is used.

User Exit 1

You may supply a user exit that is invoked after every record is read from the PTFIN data set. The details are described under the topic 'User Exit 1' in Chapter 4.

The purpose of this user exit is to enable you do examine every modification control statement, including comments, and text record in the PTFIN data set input stream, delete records that are not desired, and add records at any place in the input stream. After every record is read, your exit routine is invoked. If you determine that a record should be added following a record, you do so by returning the appropriate return code. If you determine that a record should be deleted, you do so by returning the appropriate return code. You may also determine that the current SYSMOD being read should be terminated, the RECEIVE function should be terminated, or SMP should be terminated, and by setting the appropriate return code, the requested action will be taken.

This user exit is activated after the first record is read from PTFIN and is deactivated when the RECEIVE function is terminated.

REJECT Processing

REJECT processing deletes SYSMODs from the PTS and deletes any temporary libraries loaded during RECEIVE processing for those SYSMODs.

Selection of SYSMODs

You can select specific SYSMODs to be rejected, exclude SYSMODs from mass rejection, or, by default, cause mass rejection of every eligible SYSMOD on the PTS.

Mass Rejection

Mass rejection is the default if you do not specify the SELECT or EXCLUDE keywords on the REJECT control statement. The SYSMODs eligible for mass rejection are the ones that have never been applied or accepted. Every SYSMOD that does not have any APPID or ACCID subentries in its SYSMOD entry on the PTS is selected. An APPID subentry identifies a CDS to which a SYSMOD has been applied and an ACCID subentry identifies an ACDS to which a SYSMOD has been accepted.

EXCLUDE

To exclude SYSMODs from mass rejection, you must specify the SYSMODs you want to exclude as values of the EXCLUDE keyword on the REJECT control statement. Every SYSMOD that does not have any APPID or ACCID subentries in its SYSMOD entry is selected for rejection, except for those SYSMODs identified using the EXCLUDE keyword.

SELECT

To selectively reject SYSMODs, you must specify the SYSMODs you want to reject as values of the SELECT keyword on the REJECT control statement. Every SYSMOD specified is selected regardless of whether any APPID or ACCID subentries are present in the SYSMOD entry. You must specify any SYSMODs that are requisites of the SYSMODs that you have selected, if you want them rejected at the same time, because SMP does not automatically reject them.

It is possible for you to selectively reject a SYSMOD that has been applied or accepted. However, because a SYSMOD should usually be both applied and accepted, a message is issued to warn you that you have rejected an incompletely processed SYSMOD.

Updating the PTS SYSTEM Entry

When a function SYSMOD is rejected and has not been applied or accepted, the SYSMOD-ID of the function is deleted from the list of FMIDs in the PTS SYSTEM entry.

Temporary Library Deletion

When you delete SYSMODs that had temporary libraries loaded during RECEIVE processing, those libraries are also deleted. If you neglected to include the SMPTLIB DD statement that defines the direct access volumes containing the temporary libraries, all SYSMODs with relative files are terminated. If a library is not found, a warning message is issued and the associated SYSMOD is deleted. If more files are specified in the FILES operand on a header modification control statement than are referenced in the RELFILE operand, messages are issued for those files not found when the data sets are deleted from the SMPTLIB volumes.

REJECT Messages

REJECT processing produces messages on SMPOUT. No reports are generated during REJECT processing.

APPLY Processing

APPLY processing updates the target system libraries and SMP data sets for those SYSMODs processed.

APPLY processing consists of the following major steps:

- SYSMOD selection
- Processing ++IF modification control statements
- Processing inline JCLIN
- Element selection
- Updating target system libraries and the SMP data sets
- Completion processing
- Termination processing
- Issuing APPLY reports and messages

In addition, APPLY also has special logic to handle:

- Source and macro updates
- Reprocessing previously applied SYSMODs

Before APPLY processing can take place, checks are accomplished to ensure that the SMP data sets defined in the DD statements are valid and consistent. If any of the following checks fail, an error message is issued and APPLY processing is terminated:

- The CDS must have a SYSTEM entry.
- The PTS must have a SYSTEM entry.
- The CDS SYSTEM entry must indicate that it is processable by this version of SMP.
- The CDS SYSTEM entry must have a CDSID subentry.
- The SREL subentry value in the CDS SYSTEM entry must be present as an SREL subentry in the PTS SYSTEM entry.

SMP does not check to determine whether the CRQ and SCDS you are using correspond to the CDS to be used; therefore, you must ensure that these data sets are the correct ones to be used.

Obtaining the PEMAX Value

The PEMAX value used is the greatest of all the PEMAX values in the available SYSTEM entries on the CDS, PTS, or ACDS, if present. If no PEMAX value is present in the SYSTEM entries, SMP uses the default value of 500 subentries. See Chapter 4 for a discussion of PEMAX values.

SYSMOD Selection

To select SYSMODs for processing, APPLY uses the PTS, which contains SYSMODs that may be applicable to your target system, the CNTL data set, which contains the APPLY control statement and options, the CDS, which defines the environment of the target system and describes functions and service that are already applied, and the CRQ, which contains data from ++IF modification control statements of SYSMODs previously applied.

There are three options available on the APPLY control statement that govern SYSMOD selection:

- SELECT - selects specific SYSMODs for processing
- EXCLUDE - excludes specific SYSMODs from mass application
- GROUP - groups SYSMODs together

If no option is specified, the default is mass application; that is, every eligible SYSMOD on the PTS is applied.

Function SYSMODs that contain a ++VER modification control statement with a DELETE operand must be explicitly specified on the SELECT or GROUP keyword.

<u>Mass Application</u> (<u>with or without EXCLUDE</u>)

Mass application occurs if the APPLY control statement does not specify either the SELECT or GROUP keywords. Every eligible SYSMOD in the PTS is selected for application except for the SYSMODs that have already been successfully applied or for which a RESTORE operation was attempted and failed.

If you specify the EXCLUDE keyword, the specified SYSMODs are not selected for processing.

If the SYSMODs that you are mass applying include a function SYSMOD that contains a ++VER modification control statement with a DELETE keyword, SMP issues an error message and terminates APPLY processing.

SELECT

Only the SYSMODs that you specify in the SELECT operand list on the APPLY control statement are processed. If you select an ineligible SYSMOD, APPLY processing issues an error message and terminates the SYSMOD.

If you select a SYSMOD that has requisite service or function SYSMODs that are not already applied, you must also specify those requisite SYSMODs as SELECT operands, or APPLY processing issues an error message and terminates the SYSMOD.

GROUP

The SYSMODs that you specify in the GROUP operand list on the APPLY control statement are selected for application. In addition, the SYSMODs that are requisites of those SYSMODs (and their requisites, and so forth) are also selected for application.

Requisite SYSMODs are the SYSMODs referenced by the PRE and REQ keywords on the associated ++VER modification control statement, and the SYSMODs referenced by the REQ keyword on the ++IF modification control statements.

As a result of specifying GROUP, additional function or service SYSMODs may also be automatically processed to prevent you from inadvertently omitting a SYSMOD that should have been applied with the others.

There are, however, some cases when a SYSMOD that was not specified in the GROUP list and is not already applied, but is not selected for application. The following defines these cases:

- If a function SYSMOD is specified in the FMID keyword on a ++VER modification control statement and that function has not been applied, you must include the SYSMOD-ID of that function SYSMOD in the GROUP operand, or the SYSMOD that included the FMID keyword is terminated.

  For example, PTF UZ00004 was specified in the GROUP operand, but the function SYSMOD that PTF UZ00004 specified in the FMID operand was not applied and not specified in the GROUP operand. Processing of PTF UZ00004 is terminated. The APPLY control statement and the ++PTF and ++VER modification control statement for PTF UZ00004 follow:

```
APPLY GROUP(UZ00004).
```

```
++PTF(UZ00004).
++VER(Z038) FMID(GBP1501).
```

  The function SYSMOD should have been specified in the GROUP operand. To correct the error, specify:

```
APPLY GROUP(UZ00004,GBP1501).
```

- If a dependent level function SYSMOD specifies a function SYSMOD as a requisite in the REQ operand list on the ++VER modification control statement, but the requisite function is not applied, is not included in the GROUP operand list, and is a base level function, the specifying SYSMOD is terminated. A base level function does not have an FMID keyword on its ++VER modification control statement and defines elements of the base system.

  For example, function GCD1702 specified function HVT1500 as a requisite SYSMOD using the REQ keyword, but function HVT1500 was not applied, not included in the GROUP operand and is a base level function. In addition, function GCD1702 is a dependent level function because it has an FMID keyword on its ++VER modification control statement. Function GCD1701 is already applied. Processing of function GCD1702 is terminated. The following shows the APPLY control statement, the ++FUNCTION and ++VER modification control statements for functions GCD1702 and HVT1500, and the relationships between the SYSMODs:

```
APPLY GROUP(GCD1702).

++FUNCTION(GCD1702).
++VER(Z038) REQ(HVT1500) FMID(GCD1701).

++FUNCTION(HVT1500).
++VER(Z038).
```

```
        ┌───────────┐
        │  GCD1701  │
        └───────────┘
              │
              │
        ┌───────────┐                ┌───────────┐
        │  GCD1702  │──────────>     │  HVT1500  │
        └───────────┘     REQ        └───────────┘
```

Termination can be avoided by specifying function HVT1500 in the GROUP operand list. To correct the error, specify:

```
APPLY GROUP(GCD1702,HVT1500).
```

You could optionally specify the BYPASS(REQ) option on the APPLY control statement to avoid processing function HVT1500.

- If a service SYSMOD specifies a function SYSMOD as a requisite in the REQ operand list on a ++VER modification control statement or in the REQ operand list on an ++IF modification control statement, but the function SYSMOD is not applied or specified in the GROUP operand list, the service SYSMOD is terminated.

  For example, PTF UZ00019 specifies function GVT1603 as a requisite in the REQ operand list on an ++IF modification control statement, but function GVT1603 was not applied or specified in the GROUP operand list and function HED1201 is applied. Processing of PTF UZ00019 is terminated. The following shows the APPLY control statement and the ++PTF, ++VER, and ++IF modification control statements for PTF UZ00019:

```
APPLY GROUP(UZ00019).

++PTF(UZ00019).
++VER(Z038) FMID(GVT1509).
++IF FMID(HED1201) THEN REQ(GVT1603).
```

To correct the error, specify:

```
APPLY GROUP(UZ00019,GVT1603).
```

Optionally, you could specify the BYPASS(IFREQ) option on the APPLY control statement to avoid processing function GVT1603.

Using the CRQ Entries

APPLY processing saves the ++IF modification control statements from applied SYSMODs in the CRQ as SYSMOD entries. In addition, FMID entries are created or updated for each FMID operand value specified in those ++IF modification control statements. An FMID entry names each SYSMOD that specified that FMID as the value of its FMID operand in the ++IF modification control statements. Creating the CRQ entries is illustrated in Figure 7.

```
++PTF(UZ00004).

++VER(Z038) FMID(GVT1501).

++IF FMID(GVT1502) REQ(UZ00005).


++PTF(UZ00005).

++VER(Z038) FMID(GVT1502).

++IF FMID(GVT1503) REQ(UZ00007,UZ00009).
```

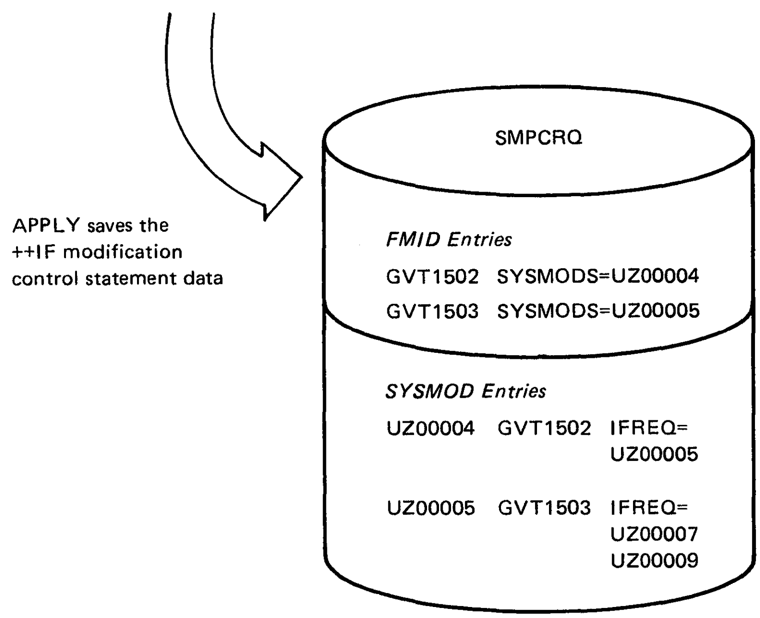

Figure 7. Creating the CRQ Entries

When a function SYSMOD not previously applied is selected for application, SMP compares the FMID entries on the CRQ with the SYSMOD-ID of the function SYSMOD to see if there is a match. If an FMID entry is found, then each of the CRQ SYSMOD entries for the SYSMOD named in the FMID entry are read to determine the requisite SYSMODs that are now necessary to be applied.

Use of the CRQ entries when a function SYSMOD is applied is shown in Figure 8.

When you apply a function . . . . . .

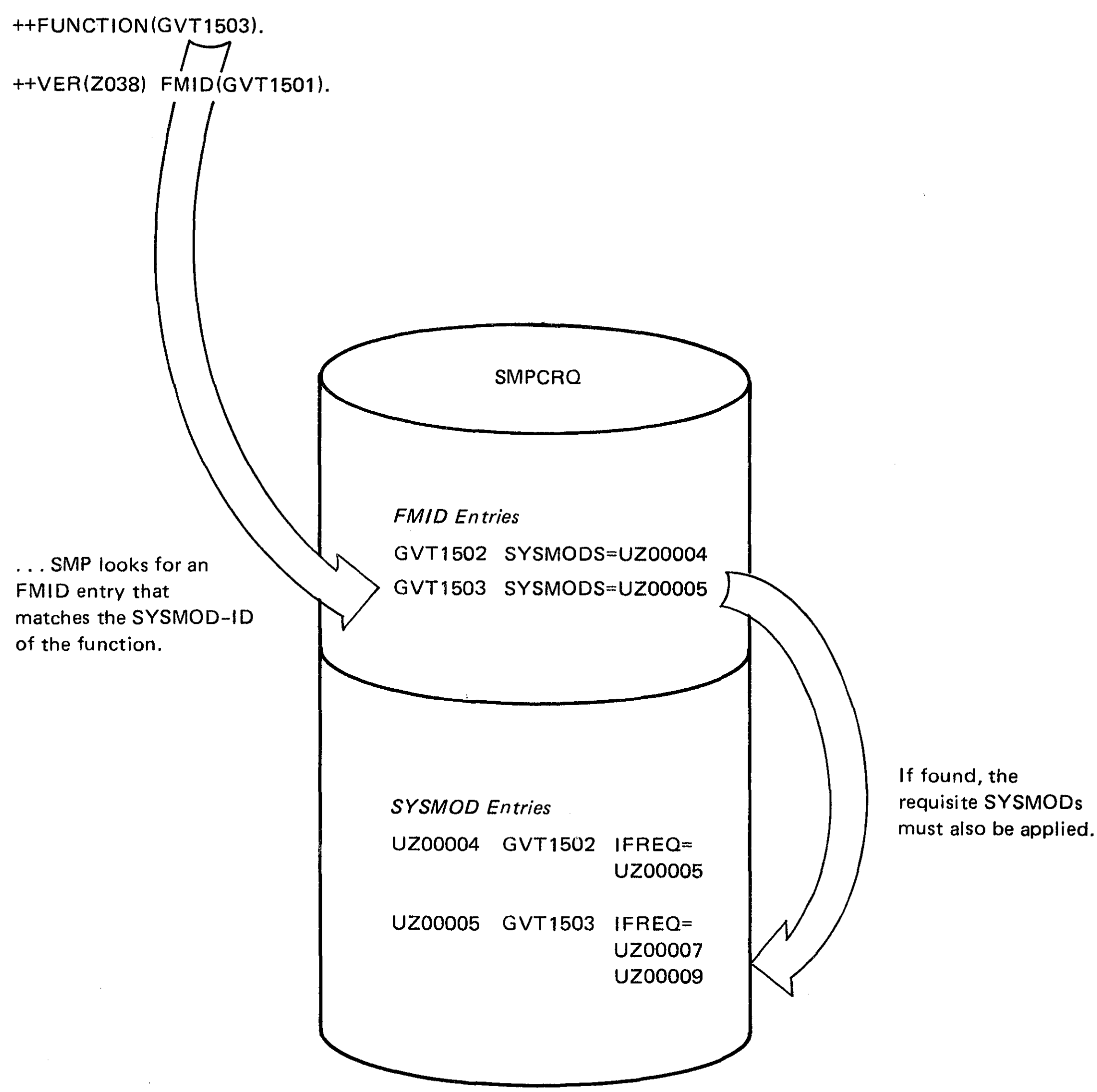

Figure 8. Applying a FUNCTION SYSMOD

If a requisite SYSMOD has not been applied and is in the PTS, it is automatically selected for application if either mass mode or GROUP mode processing is taking place. This facility prevents regression of your target system when new function is applied.

Using the ++IF Modification Control Statements

When a SYSMOD is selected, the ++IF modification control statements immediately following the ++VER modification control statement that applies to the target system are read from the MCS entry on the PTS and used to determine which SYSMODs are requisites of the selected SYSMOD. Requisite SYSMODs are specified in the REQ operands of the ++IF modification control statements. SMP processes any ++IF modification control statements with FMID operands that specify function SYSMODs that are already applied or are selected for concurrent application. Any requisite SYSMOD that has not been applied or selected for application and that exists on the PTS is selected for application when processing in GROUP mode.

Processing Inline JCLIN

When a selected SYSMOD contains a ++JCLIN modification control statement, APPLY processing performs the JCLIN updates to the CDS before it processes the updates specified in the element modification control statements.

Each entry in the CDS that is affected by the JCLIN update is saved before the update is performed in a BACKUP entry on the SMP Save Control Data Set (SCDS). Each BACKUP entry in the SCDS records the SYSMOD-ID of the SYSMOD that contained the inline JCLIN and the type of update performed.

The contents of the BACKUP entries are described in the "LIST SCDS Operands" discussion in Chapter 7.

Inline JCLIN is not processed for SYSMODs that are selected for APPLY processing but that are either superseded or deleted due to specifications in other SYSMODs being processed concurrently.

Permanent updating of the CDS entries occurs when the associated SYSMOD entry is written to the CDS with the ERROR indicator set.

The NOJCLIN option on the APPLY control statement is used to prevent the processing of inline JCLIN. NOJCLIN can be used if the JCLIN contained data that would overlay user modified entries in the CDS.

If you specify NOJCLIN without an operand list, inline JCLIN is not processed for any SYSMOD that was selected and contained ++JCLIN modification control statements. If you specify NOJCLIN with an operand list, inline JCLIN is not processed for the specified SYSMODs.

Element Selection

Because any combination of function and service SYSMODs can be processed concurrently, and because you can apply multiple versions of the same element at the same time, APPLY processing has to choose the correct modification(s) to the element from the selected SYSMODs. Element selection is based on information included in the modification control statements of the selected SYSMODs, including function hierarchy (FMID, VERSION) and service order (PRE, SUP). These are primarily ++VER modification control statement operands.

To determine which modification(s) of the element should be selected, SMP uses the following rules based on the order of appearance. Note that the FMID subentry of an element entry can dynamically change during the selection process when a selection rule is met. However, the actual element entry is not updated until the SYSMOD(s) affecting that element are successfully processed.

- If the element entry already exists on the CDS and you are processing a function SYSMOD that updates the element, then, in order for the element to be selected, the value of the FMID subentry in the element entry in the CDS must appear:

    - In the FMID operand on the ++VER modification control statement (see Example 1),
    - In the VERSION operand on the ++VER modification control statement (see Example 2), or
    - In the VERSION operand on the element modification control statement (see Example 3).

  The following are examples of the preceding selection rule: HDP1602 is the value of the FMID subentry in MOD entry IEYXXMOD on the CDS:

Example 1:

```
++FUNCTION(HDP1702).
++VER(Z038) FMID(HDP1602).
++MOD(IEYXXMOD).
```

Example 2:

```
++FUNCTION(HDP1702).
++VER(Z038) VERSION(HDP1602).
++MOD(IEYXXMOD).
```

Example 3:

```
++FUNCTION(HDP1702).
++VER(Z038).
++MOD(IEYXXMOD) VERSION(HDP1602).
```

- If function SYSMODs that contain versions of the same element are being processed concurrently, and one function SYSMOD specifies the other function SYSMOD in the FMID operand on the ++VER modification control statement, then the element is selected from the specifying SYSMOD.

  The following example shows two function SYSMODs that contain macro DPMACRO and the hierarchical structure of the functions. Because function HDP1804 specifies function HDP1803 in the FMID operand on its ++VER modification control statement, the version of the macro is selected from HDP1804.

```
                                         ┌------------┐
                                         |  HDP1802   |
                                         └------------┘
                                               |
                                               |
++FUNCTION(HDP1803).                     ┌------------┐
++VER(Z038) FMID(HDP1802).               |  HDP1803   |
++MAC(DPMACRO).                          └------------┘
                                               |
                                               |
++FUNCTION(HDP1804).                     ┌------------┐
++VER(Z038) FMID(HDP1803).               |  HDP1804   |
++MAC(DPMACRO).                          └------------┘
```

- If function SYSMODs that contain versions of the same element are being processed concurrently, the version of the element that is selected is from the SYSMOD that specifies the other SYSMODs in the VERSION operand of the ++VER or element modification control statement.

  The following example shows two function SYSMODs that contain macro DPMACRO and the hierarchical structure of the functions. Since HDP1803 specifies HDP1802 in the VERSION operand of its ++VER modification control statement, the version of the macro is selected from HDP1803.

```
++FUNCTION(HDP1802).
++VER(Z038) FMID(HDP1801).
++MAC(DPMACRO).

++FUNCTION(HDP1803).
++VER(Z038) FMID(HDP1801).
++MAC(DPMACRO) VERSION(HDP1802).
```

```
              ┌────────────┐
              | HDP1801    |
              └────────────┘
                    |
                    |
          ┌─────────────────┐
          |                 |
   ┌────────────┐     ┌────────────┐
   | HDP1802    |     | HDP1803    |
   └────────────┘     └────────────┘
```

- If function and service SYSMODs applied concurrently contain the same element, but the service SYSMOD specifies the function SYSMOD in the FMID operand on the ++VER modification control statement and replaces the element, the element is selected from the service SYSMOD.

  The following example shows a function and service SYSMOD that contain macro DPMACRO and the hierarchical structure of the SYSMODs. Since PTF UZ00063 specifies function HDP1803 in the FMID operand of its ++VER modification control statement and replaces the element, the element is selected from the PTF.

```
++FUNCTION(HDP1803).
++VER(Z038) FMID(HDP1801).
++MAC(DPMACRO) VERSION(HDP1802).

++PTF(UZ00063).
++VER(Z038) FMID(HDP1803).
++MAC(DPMACRO).
```

```
                  ┌────────────┐
                  │  HDP1801   │
                  └────────────┘
                        │
                        │
             ┌──────────────────┐
             │                  │
      ┌────────────┐     ┌────────────┐
      │  HDP1802   │     │  HDP1803   │
      └────────────┘     └────────────┘
                               │
                               │
                         ┌────────────┐
                         │  UZ00063   │
                         └────────────┘
```

- If a service SYSMOD contains a VERSION operand (on either the element or ++VER modification control statement) and the FMID of the element is equal to one of the VERSION operand values, the element is selected. When more than one service SYSMOD meets this criteria, one of them must specify in its VERSION operand the SYSMOD-IDs that are values in the FMID operands in the ++VER modification control statements of each of the other service SYSMODs.

  The following example shows two PTFs that replace module IFTAABB and the hierarchical structure of the SYSMODs. The functions FVT3101, FVT3102 and FVT3103 have been applied. Both FVT3102 and FVT3103 specify FVT3101 in the FMID operand on their ++VER modification control statements. Using the VERSION operand, function FVT3103 indicates that the elements that it contains are superior to those in function FVT3102. The FMID subentry of MOD entry IFTAABB has a value of FVT3103. Because both PTFs specify FVT3103 in the VERSION operands of their ++VER modification control statements and PTF UZ00002 specifies FVT3101 in the VERSION operand, the module is selected from PTF UZ00002.

```
++PTF(UZ00001).
++VER(Z038) FMID(FVT3101) VERSION(FVT3103).
++IF FMID(FVT3102) THEN REQ(UZ00002).
++MOD(IFTAABB) DISTLIB(AOS99).
  .

++PTF(UZ00002).
++VER(Z038) FMID(FVT3102) VERSION(FVT3101,FVT3103).
++MOD(IFTAABB) DISTLIB(AOS99).
```

```
                    ┌------------┐
                    |  FVT3101   |
                    └------------┘
                          |
       ┌-------------------------------------┐
       |                  |                  |
┌------------┐            |           ┌------------┐
|  FVT3102   |            |           |  FVT3103   |
└------------┘            |           └------------┘
       |                  |
┌------------┐     ┌------------┐
|  UZ00002   |     |  UZ00001   |
└------------┘     └------------┘
```

- If two or more service SYSMODs that replace the same element and specify the same FMID are applied concurrently, the element is selected from the SYSMOD that specifies (either directly or indirectly) the other SYSMODs in either the PRE or SUP operands of the ++VER modification control statement.

  An indirect specification can occur when one SYSMOD specifies a second SYSMOD as a prerequisite which in turn supersedes a third SYSMOD. For example, PTF UZ00003 specifies PTF UZ00002 in the PRE operand of its ++VER modification control statement and PTF UZ00002 specifies PTF UZ00001 in the SUP operand of its ++VER modification control statement. PTF UZ00001 is ignored because it is superseded. The element is selected from PTF UZ00003 because it contains a later replacement of the element, indicated by the PRE operand.

- If one or more SYSMODs contain updates to an element (that is, ++MACUPD, ++UPDTE or ++SRCUPD), and none of the SYSMODs have been superseded by SYSMODs being applied at the same time, then all of the updates are selected. See "Processing Source Module and Macro Updates" later in this chapter for further discussion.

- If two or more SYSMODs modify the same module using IMASPZAP modifications, only one is selected. If SMP can determine the service order, the SYSMOD that is a prerequisite for any others is selected. If the lowest service level SYSMOD is superseded by the next highest service level SYSMOD, then the latter is selected. If no service order can be determined, then the SYSMOD that SMP selects cannot be determined; the other SYSMODs are terminated.

  Thus, SMP can serialize the processing of some SYSMODs. You must, however, use multiple APPLY statements in the same invocation of SMP to separately apply each SYSMOD with ++ZAP modification control statements to the same module.

SMP does not check for function SYSMODs that are specified in the NPRE operand of the ++VER modification control statement of a previously applied SYSMOD. This condition does not cause termination, but you should keep a record of those function SYSMODs that specify other function SYSMODs as negative prerequisites.

SUP and DELETE Processing During Element Selection

Some SYSMODs selected for APPLY processing might be either superseded or indirectly deleted by other SYSMODs also selected. When this occurs, SMP treats these SYSMODs as if they do not exist and does not process their ++IF and ++JCLIN modification control statements. Element selection ignores any SYSMODs that are deleted or superseded, but records their existence in the CDS.

The CRQ entries for SYSMODs that have been applied but are superseded by selected SYSMODs are treated as if they did not exist.

Deletion of SYSMODs can only occur at the function SYSMOD level. Deletion is specified by the DELETE keyword on the ++VER modification control statements of function SYSMODs. When a function SYSMOD selected for APPLY processing specifies another function SYSMOD in the DELETE operand, SMP checks the CDS to determine if the function to be deleted was applied. If so, then every entry in the CDS, including SYSMOD entries, that contains an FMID subentry that matches the SYSMOD-ID in the DELETE operand is logically deleted; that is, the entries are treated as if they did not exist, but are not physically deleted.

Additional function and service SYSMODs might also be deleted if they are dependent on the SYSMODs that were deleted in the previous step. This is done by looking at each CDS entry that contains an FMID subentry that is the same as those SYSMODs that have already been logically deleted. This process is repeated until no further SYSMODs are found within the hierarchy of the function SYSMOD specified for deletion.

The result of this process is the logical deletion of all SYSMODs within the hierarchy of the specified function SYSMOD.

In the following example, function SYSMODs GDE1203, GDE1303, and GDE1403, and service SYSMODs UZ00009, UZ00010 and UZ00004 are logically deleted as a result of specifying 'DELETE(GDE1203)' on the ++VER modification control statement.

```
++FUNCTION(GDE2000).
++VER(Z038) FMID(HVT1500) DELETE(GDE1203).
```

```
                  ┌───────────┐
                  │  GDE1203  │
                  └───────────┘
                        │
         ┌──────────────────────────────────┐
         │              │                   │
   ┌───────────┐  ┌───────────┐  ┌───────────┐
   │  UZ00004  │  │  UZ00009  │  │  GDE1303  │
   └───────────┘  └───────────┘  └───────────┘
                                        │
                                 ┌───────────┐
                                 │  GDE1403  │
                                 └───────────┘
                                        │
                                 ┌───────────┐
                                 │  UZ00010  │
                                 └───────────┘
```

All CRQ entries associated with deleted SYSMODs are also logically deleted, with the exception of entries that reference the SYSMOD containing the DELETE keyword. This ensures that the conditional action that should occur as a result of applying the SYSMOD that contains the DELETE keyword does occur, thus preventing regression.

DISTLIB Operand Checking

When an element is selected for application and a CDS entry for that element already exists, the value of the DISTLIB operand on the element modification control statement is compared with the DISTLIB subentry in the CDS element entry. If they are not equal, SMP issues a message to inform you of an error condition and terminates the SYSMOD containing the element.

If service and function SYSMODs are being processed and contain the same element, and an element entry does not exist on the CDS, the service SYSMODs must specify the same DISTLIB as the function SYSMODs on the element modification control statements. If they do not, SMP issues an error message and the APPLY function is terminated.

If two service SYSMODs update or replace the same element, have different DISTLIB operand values, and are both eligible for processing, but an entry for the element does not exist on the CDS, then the first SYSMOD processed causes a CDS element entry to be created containing the DISTLIB from its modification control statement. SMP terminates the second SYSMOD.

Updating Target System Libraries and SMP Data Sets

Updating of target system libraries and SMP data sets does not occur until all possible verification checks have been made. SMP attempts to reflect the state of the target system in the SMP data sets even if other error conditions occur while updating the libraries or data sets, resulting in either the termination of APPLY processing or SMP.

Creation of SYSMOD Entries on the CDS

For each SYSMOD that was not terminated, a SYSMOD entry is created on the CDS or replaced if an entry already exists. The entry includes the data from the applicable ++VER modification control statement, subentries for each of the elements included in the SYSMOD package, and indicators that are set when ++IF and ++JCLIN modification control statements are present. At this point in the processing, no updates have been made to the target system libraries. The ERROR status indicator is set in the SYSMOD entry, and is later reset if the SYSMOD is processed successfully.

APPLY Indication in the PTS SYSMOD Entries

For each SYSMOD that was not terminated, the identifier of the CDS to which the SYSMOD has been applied is added to the SYSMOD entry in the PTS. This is done by adding the CDSID (from the CDS SYSTEM entry) to the APPID subentries in the PTS SYSMOD entry, unless a subentry already exists for that CDS identifier.

Data Set Usage

After SMP has selected a SYSMOD and created any required CDS entries, the target system libraries are updated. In addition to the PTS, CRQ, SCDS, and CDS data sets, APPLY processing uses the WRK1, WRK2, WRK3, WRK4, and WRK5 data sets for temporary storage when it invokes programs such as the linkage editor and IEBUPDTE. Usage of these data sets is described in Chapter 9.

When processing SYSMODs that specify the FILES operand on the header modification control statement, the temporary libraries on the volumes identified by the SMPTLIB DD statement are used directly by the invoked programs. If element modification control statements in the SYSMODs contain the TXLIB or LKLIB keywords, you must provide the appropriate DD statements that reference these libraries.

For macro and source modules, SMP copies the individual members of these libraries to the appropriate target system libraries using IEBCOPY. When MALIAS is specified on the ++MAC modification control statement or MALIAS subentries exist in the MAC entry on the CDS, SMP uses IEBUPDTE instead of IEBCOPY.

For load module data, SMP copies the individual members of these libraries to the appropriate target system libraries only if the load module on the target system consists of only one module. This can occur for modules copied from the distribution libraries to the target system libraries during system generation. Otherwise, the temporary libraries are used as input to the linkage editor.

The MTS and STS are used to store macros and source modules, respectively, when there is no SYSLIB subentry in the MAC or SRC entry and the SYSLIB keyword is not specified on the modification control statement. The individual members remain stored in these data sets until they are replaced by subsequent APPLY processing or are deleted by ACCEPT or UCLIN processing.

Use of the LMOD Operand

When the LMOD operand is specified on a ++MOD modification control statement, the values in the operand list are added to the MOD entry on the CDS as LMOD subentries. If an LMOD entry does not exist for a name that is specified in the list, no LMOD entry is created. No link edit is performed for that load module, and SMP issues a warning message and creates a BACKUP entry on the SCDS for the CDS MOD entry before modification.

Deletion of Elements

When an element selected for application is to be deleted from the target system (specified by the DELETE operand on its associated modification control statement), the CDS entry for that element is copied to the SCDS for use if the SYSMOD is later restored. A situation can occur where an element is both added to and deleted from the target system by SYSMODs that are applied concurrently. In this case, the element entry may be created as a result of processing a ++JCLIN modification control statement for the adding SYSMOD or by APPLY processing prior to the deletion. This is necessary because the adding SYSMOD, and any other SYSMODs affecting the element other than the deleting SYSMOD, may be accepted into the distribution libraries without accepting the deleting SYSMOD, thereby allowing for the restoration of the deleting SYSMOD by itself.

In addition to the deletion of the element entry from the CDS, the element is deleted from the target system libraries. For macros and source modules, the element is deleted from either the target system library, or the MTS or STS. For modules, if the corresponding load module is comprised solely of deleted modules, the load module is deleted.

Updating the MODID Subentries of Element Entries

Updates occur to the MODID fields of element entries on the CDS as follows:

- The FMID subentry is replaced with the FMID of the SYSMOD from which the modification to the element was selected. If this is a function SYSMOD, then it is the SYSMOD-ID of the function itself.

- The RMID subentry is replaced with the SYSMOD-ID of the SYSMOD from which the replacement modification to an element was selected. A replacement modification causes the deletion of any existing UMID subentries.

  If a MOD entry is being updated as the result of an assembly of an ASSEM entry, the RMID is replaced with the SYSMOD-ID of the SYSMOD that modified the macro causing the assembly and the RMID subentry indicator is set to reflect this occurence.

  If a service SYSMOD containing a replacement of an element does not supersede every SYSMOD indicated in the UMID subentries for the element entry and BYPASS(ID) was specified, SMP issues a warning message for each SYSMOD that was not superseded. The update type subentry (that is SRCUPD, MACUPD or ZAP) in the nonsuperseded SYSMOD entry is indicated as regressed, unless it is being concurrently reapplied to the replacement copy of the element. See "Processing Service Updated Function SYSMODs" later in this chapter for further information on RMID processing.

- UMID subentries are added for every SYSMOD that has update modifications to an element that specifies the RMID of the element as a prerequisite SYSMOD. If both replacement and update modifications are being concurrently applied and the SYSMODs with the updates specify the SYSMOD from which the element was selected as a prerequisite, the RMID is replaced and the UMIDs are added. If a SYSMOD with an update modification to an element supersedes another SYSMOD with an update modification to the same element, then the UMID subentry for the superseded SYSMOD, if it exists, is deleted from the element entry.

If a SYSMOD containing an update to an element does not specify the SYSMOD in the RMID subentry as a prerequisite (unless the FMID is the same as the RMID) and BYPASS(ID) was specified, SMP issues a warning message and adds the SYSMOD-ID to the element entry as a UMID subentry.

Completion Processing

After the target system libraries are updated, the CDS SYSMOD and associated element entries are updated. If any target system library updating did not complete successfully, SMP does not change the CDS entries.

SYSMOD Completion

A SYSMOD is considered completely processed when:

- All of its elements have been updated or replaced using either the element data provided with the SYSMOD or other SYSMODs that had superior function or service levels, and
- All of its requisite SYSMODs have also been completely processed.

When these conditions are true, SMP sets the ERROR indicator off in the CDS SYSMOD entry and issues a completion message.

SUP Processing after SYSMOD Completion

CDS SYSMOD entries include multiple subentries for superseding SYSMODs, referred to as SUPBY subentries. When a SYSMOD supersedes another SYSMOD, as specified by a value in the SUP operand on the ++VER modification control statement, the SYSMOD-ID becomes a SUPBY subentry in the SYSMOD entry created or updated for the superseded SYSMOD.

DELETE Processing after SYSMOD Completion

When a function SYSMOD that deleted another function SYSMOD is successfully processed, all CDS entries that were logically deleted and not replaced by the deleting SYSMOD or other SYSMODs being processed concurrently are physically deleted from the CDS. If any of these entries are for elements, then the elements are removed from target system libraries where applicable. The following actions occur for deleted elements:

- Macros that were members of target system libraries, indicated by the SYSLIB subentries of the CDS MAC entries, are deleted from those libraries.
- Source modules that were members of target system libraries, indicated by the SYSLIB subentries of the CDS SRC entries, are deleted from those libraries.
- Load modules are deleted from the target system libraries if all of the modules in the load modules are deleted.

When a function SYSMOD that deleted another function SYSMOD is successfully processed, the CRQ FMID entries for the deleted function are removed from the CRQ.

Each SYSMOD that was specified in the DELETE operand of the ++VER modification control statements of successfully processed SYSMODs has a SYSMOD entry created or updated on the CDS indicating the deleting SYSMOD. The purpose for this entry is to prevent deleted function SYSMODs from being reprocessed by APPLY.

Updating the CRQ

The CRQ SYSMOD entries are created when the associated SYSMOD is successfully processed. The FMID entries affected are created or updated at the same time.

Termination Processing

There are two types of termination that can occur as a result of APPLY processing. In the first case, the SYSMOD currently being processed is terminated and APPLY processing continues for the remaining SYSMODs. In the second case, APPLY processing is terminated.

SYSMODs that you specify for APPLY processing must meet eligibility requirements in order to be selected for processing. Eligible SYSMODs are those SYSMODs on the PTS that do not have the ERROR indicator set in the PTS SYSMOD entry, and that have at least one ++VER modification control statement with an SREL operand value equal to the SREL subentry in the CDS SYSTEM entry. In addition, those SYSMODs with FMID keywords on ++VER modification control statements are eligible if the function identified is applied or being selected for application concurrently.

Termination of SYSMOD Processing

Termination of a SYSMOD causes a return code of 8; termination occurs in response to any of the following conditions:

- If you have specified the GROUP keyword on the APPLY control statement and implicitly or explicitly specified an ineligible SYSMOD, SMP terminates the entire group of SYSMODs.

- Inline JCLIN processing failure. The entries that are affected are restored to the state that existed before JCLIN processing.

- DISTLIB operand checking failure.

- For GROUP and mass mode processing, a requisite SYSMOD has not been applied and is not available in the PTS for concurrent application.

- A requisite SYSMOD has not been applied and is specified as a value of the EXCLUDE operand on the APPLY control statement.

- A SYSMOD specified as a value of the SELECT operand on the APPLY control statement has a requisite SYSMOD that has not been applied and was not specified in the SELECT list.

- A SYSMOD does not pass the ID validation checking. This can occur when APAR fixes have been applied but are not superseded by the SYSMOD for any element. The following MODID verification checks are done for service SYSMODs only:

- If the SYSMOD is replacing an element, the RMID of the element entry must be specified in the PRE or SUP operand of the ++VER modification control statement unless the RMID is equal to the FMID. Any UMIDs in the element entry must be specified in the SUP operand of the ++VER modification control statement. For every MODID that is not specified, SMP issues an error message.

- If the SYSMOD is updating an element, the RMID of the element entry must be specified in the PRE operand of the ++VER modification control statement unless the RMID is equal to the FMID. If the above condition is not true, an error message is issued. If any UMIDs are present in the element entry, then for each one that is not specified in the PRE or SUP operand of the ++VER modification control statement, SMP issues a warning message that indicates a possible regression of a previous modification.

These checks are performed internally when multiple service SYSMODs replace or update the same element. That is, the CDS element entry image is dynamically updated according to function hierarchy and service order algorithms and the appropriate MODID verification checks are performed.

- A SYSMOD has a requisite relationship with a SYSMOD that has been terminated.

- Function SYSMODs, not already applied and not specified in the GROUP operand, can cause termination, described under "SYSMOD Selection" earlier in this discussion.

- If, during the selection process, an eligible SYSMOD has more than one ++VER modification control statement meeting the selection criteria, and the functions specified in both ++VER modification control statements are applied, the SYSMOD is terminated because SMP cannot determine which ++VER modification control statement to choose, since they are all applicable. This is illustrated in the following example:

```
++PTF(UZ00029).
++VER(Z038) FMID(GVT1502).
++VER(Z038) FMID(GVT1505).
```

- A DD statement is missing for a target system library.

APPLY Processing Termination

APPLY processing termination causes a return code of 12. For each of the following conditions, SMP issues an error message. APPLY reports are not produced when a function SYSMOD is terminated before selection processing completes. Termination can be caused by any of the following conditions:

- Termination of processing of any function SYSMOD.
- Two function SYSMODs are specified in the SELECT or GROUP list and one specifies the other in the DELETE operand of its ++VER modification control statement.
- Two function SYSMODs are specified in the SELECT or GROUP list, or are selected in mass mode, and one specifies the other in the NPRE operand of its ++VER modification control statement.
- A function SYSMOD that specifies a previously-applied SYSMOD in the NPRE operand of its ++VER modification control statement is specified in the SELECT or GROUP list.
- A function SYSMOD that has been deleted by a previously-applied SYSMOD (that is, a SYSMOD entry on the CDS indicates that the SYSMOD has been deleted) is specified in the SELECT or GROUP list.
- A function SYSMOD that has been superseded by a previously-applied SYSMOD (that is, a SYSMOD entry on the CDS indicates that the SYSMOD is superseded) is specified in the SELECT or GROUP list. A service SYSMOD in the same situation is not processed but the APPLY function is not terminated.

Avoiding Termination of a SYSMOD

Certain error conditions that cause the termination of a SYSMOD can be avoided by specifying the BYPASS operand on the APPLY control statement. In BYPASS mode, some error conditions are treated as warning conditions. The following operand values can be specified with the BYPASS operand to avoid termination:

- ID - specifies that error conditions in the ID validation check of the RMID and UMID fields do not cause termination. Where error messages would ordinarily be issued, warning messages are issued instead.

- PRE - specifies that SYSMODs that are not present on the CDS and are not currently candidates for application, but that are specified in the PRE operand of ++VER modification control statements, do not cause termination.

- REQ - specifies that SYSMODs that are not present on the CDS and are not currently candidates for application, but that are specified in the REQ operand of ++VER modification control statements, do not cause termination.

- IFREQ - specifies that SYSMODs that are not present on the CDS and are not currently candidates for application, but that are specified in the REQ operand of ++IF modification control statements, do not cause termination.

Processing of Partially Applied SYSMODs

A SYSMOD is considered only partially applied if the ERROR indicator is set in the SYSMOD entry on the CDS and the RESTORE indicator is not set. A subsequent APPLY pass can process the SYSMOD, and if successfully processed, SMP resets the ERROR indicator.

APPLY CHECK Facility

The intent of the CHECK option is to perform a 'dry run' to inform you of possible error conditions and to provide reports of SYSMOD status, libraries that will be updated, regression conditions, and SYSMODs that will be deleted. Permanent updating of SMP data sets and target system libraries does not occur.

APPLY Reports and Messages

Four reports are produced as a result of APPLY processing and are explained in Chapter 6. In addition, messages are issued to inform you of error and warning conditions that are detected before the reports are produced. The messages are produced on the SMPOUT data set unless you have provided an SMPRPT DD statement. In this case, the reports are produced on the SMPRPT data set.

Reports are not produced if a function SYSMOD terminates.

Processing Source Module and Macro Modifications

SMP allows you to receive and apply multiple modifications to the same macro or source module. These modifications can exist in different types of SYSMODs (++USERMODs, ++PTFs, or ++APARs) and can be processed concurrently by the APPLY function.

When two or more updates to the same source module or macro are being processed concurrently, the text from each is automatically merged based on the sequence numbers in columns 73 to 80. The order of the merge is based on the service order expressed by the PRE and SUP keywords. If SMP finds a service order relationship between all of the SYSMODs being processed, the merge occurs according to that order. When no service order is found, the SYSMODs are merged according to the type of SYSMOD, in which case updates from PTFs are merged first, followed by updates from APARs. Finally, updates from USERMODs are merged.

If any of the SYSMODs being processed do not have a service order relationship with other SYSMODS that do have a service order, the updates from the unrelated SYSMODs are merged after the updates from SYSMODs that have a service order relationship.

When SYSMODs are superseded by other SYSMODs being processed, the superseded SYSMODs are ignored, but a record of their existence is made in the CDS.

As a general rule, any object text supplied in the superior SYSMOD is used, rather than reassembling the updated source module, if all the relationships between SYSMODs are specified. The superior SYSMOD is the last SYSMOD to be merged. You can prevent APPLY processing from using the object text and assembling the updated source module by specifying the ASSEM keyword on the APPLY control statement. The assembly of the updated source module occurs automatically when:

- The object text is not present in the superior SYSMOD,
- All relationships to other SYSMODs updating the source module are not specified, or
- All relationships to previously applied SYSMODs that updated or replaced the source module are not specified.

See the "Assembly of Source Text" later in this discussion for further information.

The following examples demonstrate the rules for source module update and replacement processing:

Example 1:

```
++PTF(UZ01234).
++VER(Z038) FMID(GXY3100) SUP(AZ10022).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).

++PTF(UZ01255).
++VER(Z038) FMID(GXY3100) SUP(AZ10022,UZ01234).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).
```

The source module updates that are being applied are related by the SUP keyword. PTF UZ01234 is ignored because it is superseded by PTF UZ01255. The source module in the target system library or the STS is updated using IEBUPDTE with the source module text cards in PTF UZ01255. Because object text is included for module IXYMOD01, no assembly of the source module is done. The object text is processed by the linkage editor to replace the appropriate load module(s).

Example 2:

```
++PTF(UZ01234).
++VER(Z038) FMID(GXY3100) SUP(AZ10022).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).

++PTF(UZ01266).
++VER(Z038) FMID(GXY3100) SUP(AZ10022,UZ01234).
++SRC(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).

++PTF(UZ01277).
++VER(Z038) FMID(GXY3100) SUP(AZ10033) PRE(UZ01266).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).
```

The SYSMODs have a service order specified by the SUP and PRE keywords. PTF UZ01234 is ignored because it is superseded by PTF UZ01266. PTF UZ01277 specifies PTF UZ01266 as a prerequisite. Because the source module is replaced by PTF UZ01266, IEBCOPY is used to copy the source replacement text to the target system library or the STS. The source module is then updated using the source update text in PTF UZ01277 by invoking IEBUPDTE. The object text supplied in PTF UZ01277 is used by the linkage editor to replace the load module(s).

Example 3:

```
++PTF(UZ01234).
++VER(Z038) FMID(GXY3100) SUP(AZ10022).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).

++APAR(AZ10022).
++VER(Z038) FMID(GXY3100).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).

++USERMOD(MY00010).
++VER(Z038) FMID(GXY3100).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
```

All of the SYSMODs are not interrelated. The user modification does not specify either PTF UZ01234 or APAR AZ10022 and they do not reference USERMOD MY00010 in the PRE or SUP operands. Because the PTF supersedes the APAR, the APAR is ignored and the source update text from the PTF is used. The source update text from the user modification is merged into the update text from the PTF. Any lines of code with the same sequence numbers are taken from the user modification. The resulting merged update text is used to update the source module using IEBUPDTE. Because the user modification does not include object text and it is the superior SYSMOD, the source module is assembled and the resulting object text is link edited as or with the load module in the target system library.

Example 4:

```
++PTF(UZ01234).
++VER(Z038) FMID(GXY3100) SUP(AZ10033).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).

++APAR(AZ10022).
++VER(Z038) FMID(GXY3100).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).

++USERMOD(MY00010).
++VER(Z038) FMID(GXY3100).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
```

This example is similar to the previous one except that the PTF does not supersede the APAR. Because there is no relationship between any of the SYSMODs involved, the merge order for the source update text is the PTF, then the APAR, and finally the user modification. The resulting merged update text is then merged into the source module using IEBUPDTE and the source module is assembled because the user modification did not contain object text for the module. The resulting object text is then link edited as or with the load module in the target system library.

Example 5:

```
++PTF(UZ01234).
++VER(Z038) FMID(GXY3100) SUP(AZ10033).
++SRC(IXYMOD01) DISTLIB(XYSRCLIB).
++MOD(IXYMOD01) DISTLIB(XYMODLIB).

++APAR(AZ10022).
++VER(Z038) FMID(GXY3100).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).

++USERMOD(MY00010).
++VER(Z038) FMID(GXY3100).
++SRCUPD(IXYMOD01) DISTLIB(XYSRCLIB).
```

In this example, the PTF contains a source replacement. Because there is no relationship between any of the SYSMODs, the processing of the SYSMODs cannot take place without some special action on your part. If you want both the APAR and the user modification to update the source module text supplied with the PTF, you must specify 'BYPASS(ID)' on the APPLY control statement. If BYPASS is specified, the source module is replaced in the target system library or STS with the source text from the PTF using IEBCOPY and the text from the user modification is merged with the text from the APAR. The merged update text is merged with the replaced source module using IEBUPDTE, the resulting source module is assembled, and the resulting object text is link edited as or with the load module in the target system library.

If you do not specify BYPASS(ID), only the PTF is processed because the APAR and USERMOD do not specify the PTF as a prerequisite. You can subsequently process both the APAR and the user modification by specifying BYPASS(ID) or by modifying their respective ++VER modification control statements to specify the PTF in the PRE operand list.

Assembly of Source Text

When source text is selected from a SYSMOD for processing, the SYSMOD might also contain object text for the same module. When this occurs, the source text is not assembled unless one of the following conditions is true:

- The ASSEM keyword is specified on the APPLY control statement.

- The source text is an update (that is, the source text follows a ++SRCUPD modification control statement) and the SYSMOD does not specify all of the UMIDs in the SRC entry on the CDS in its PRE or SUP keyword values. This condition occurs when a PTF is selected and you previously applied user modifications or APARs that are not superseded by the PTF. Note that it may be necessary for you to reapply your user modifications or APARS because the PTF might replace lines of code. You might also need to modify your user modifications.

- The BYPASS(ID) option is specified on the APPLY control statement to permit the concurrent processing of updates and replacements. This is needed when APARs or user modifications are applied to a source module that is being replaced by a PTF or function SYSMOD when the APARs or user modifications do not specify the PTF or RMID subentry value as a prerequisite.

- Multiple updates to the same source module are being processed concurrently and the superior SYSMOD does not contain object text for the module.

- A MOD entry for the source module with the indicator set in the RMID subentry shows the source module was assembled because of a macro change.

IEBUPDTE Control Cards

The only IEBUPDTE control statements allowed in the SYSMOD are the ./ CHANGE and ./ ENDUP. SMP generates any ./ ALIAS statements needed and places them in the IEBUPDTE input data following the last text statement. The ./ ALIAS control statements are generated only for macro updates. Any MALIAS operand values are added to those already existing in the MAC entry on the CDS, and any duplicates are ignored. SMP generates a ./ ALIAS control statement for each MALIAS subentry in the CDS entry.

The ./ CHANGE statement, which is finally passed to IEBUPDTE along with the merged update text, is taken from the last update to be merged into the text. This permits the SSI information, if any, from the ./ CHANGE statement of the last merged update to be processed by IEBUPDTE.

IEBUPDTE Input and Output Libraries

When IEBUPDTE is invoked, the work data set containing the macro or source updates becomes the SYSIN data set. The version of the macro or source module used as the base for the updates is found in a target system library, the MTS or STS, or a distribution library, depending upon previous APPLY processing and whether the macro or source module exists in one of those libraries. The input library used becomes the SYSUT1 data set for IEBUPDTE, and the output library used becomes the SYSUT2 data set.

If the macro or source module is in a target system library, then the target system library is used for the input and output data sets.

If the macro or source module was not previously modified and is not in a target system library, then the input data set is the distribution library and the output data set is either the MTS or STS.

If previous modifications were accepted into the distribution library and the member no longer exists on the MTS or STS, then the distribution library is used as input, and the MTS or STS is used as output.

If previous modifications were accepted and the member still exists on the MTS or STS, then that data set is used for both input and output.

If previous modifications have not been accepted and the member exists on the MTS or STS, then that data set is used for both input and output.

See "ACCEPT Processing" in this chapter for related processing when SYSMODs containing source module and macro updates are accepted.

Processing the SSI Keyword

The SSI keyword can be specified on the ++MAC and ++SRC modification control statements. However, if either the TXLIB or RELFILE keyword is also specified, no action is taken because the element is copied to the target system library, or the MTS or STS.

When neither the TXLIB nor RELFILE keyword is specified, the replacement text for the element is included in the SYSMOD itself. SMP writes the replacement text to a work data set before invoking IEBUPDTE. A ./ REPRO control statement is written as the first text statement for the element in the work data set. If SSI was specified, the SSI keyword is included on the ./ REPRO statement. IEBUPDTE is then invoked to place the replacement copy in either the target system library, or the MTS or STS.

If any other SYSMODs that update the element are also being processed, they are processed as shown in "Processing Source Module and Macro Updates" earlier in this chapter, causing another invocation of IEBUPDTE.

Usage of DISTSRC, ASSEM and DISTMOD Operands

Because SMP cannot determine from the data processed by JCLIN what source modules are contained in a totally copied library, the DISTSRC, ASSEM, and DISTMOD operands are provided to pass this information to SMP when a macro is replaced or updated that results in the reassembling of source modules. The DISTSRC keyword value specifies the name of the distribution library containing the source modules. The ASSEM keyword value specifies a list of source modules and/or SYSGEN assembly modules that should be assembled during APPLY processing. The DISTMOD keyword value specifies the name of the distribution library containing the load modules. These three keywords are specified on ++MAC, ++MACUPD, and ++UPDTE modification control statements.

The ASSEM keyword values are placed in the associated SYSMOD entry on the CDS as ASSEM subentries. If any of the modules specified in the ASSEM keyword values are found on the CDS as SRC or ASSEM entries, then the DISTLIB and SYSLIB subentry values are used in lieu of the DISTSRC keyword value.

If neither a SRC nor an ASSEM entry exists for a module in the ASSEM keyword values, then a SRC entry is created. The DISTSRC keyword value is placed in the SRC entry as the DISTLIB subentry. If there is a DLIB entry on the CDS for the DISTSRC keyword value, then the SYSLIB subentry(s) from the DLIB entry are placed in the SRC entry as SYSLIB subentry(s). If no DLIB entry exists, the SYSLIB subentry in the SRC entry is left as null and the STS is used in place of a target library.

If there is no MOD entry on the CDS for a module in the ASSEM operand list, one is created. The DISTMOD keyword value is placed in the MOD entry as the DISTLIB subentry.

If no LMOD entry exists for a module, one is created, provided there is a DLIB entry on the CDS for the DISTMOD keyword value. The SYSLIB subentry(s) from the DLIB entry are placed in the LMOD entry as SYSLIB subentry(s) and the LMOD subentry is placed in the MOD entry. If no DLIB entry exists, then no LMOD subentry exists in the MOD entry and, therefore, no executable load module can be updated in the target system for that module.

After the macro update or replacement is accomplished, the assemblies of all modules specified in the ASSEM operand list are performed. If no member is found in either the source target system library or STS, or in in the distribution library for a source module specified in the ASSEM operand list, a warning message is issued and processing of the SYSMOD continues without assembling or link editing the module. If an assembly completes with a return code greater than the one that you specified in the ASMRC subentry of the PTS SYSTEM entry (or the SMP default of 4, if the ASMRC subentry is null), the processing of the SYSMOD is terminated. If the resulting object text from a successful assembly can be link edited into a load module, then the link edit is performed.

Reprocessing Applied SYSMODs

A SYSMOD that is already applied can be reapplied by specifying either the SELECT or GROUP operand with the SYSMOD-ID as an operand value. If a SYSMOD is selected for reapplication but it is indicated as superseded on the CDS, it is not reapplied. If GROUP is specified, only those SYSMODs not previously applied and/or those specified in the GROUP list are applied. Eligibility and termination rules as previously described are enforced in GROUP processing.

The processing of the individual elements within a SYSMOD occurs only when one of the following conditions is true:

- If the modification is an element replacement, the RMID of the element must be the same as the SYSMOD-ID of the SYSMOD. Any UMIDs must be superseded by the SYSMOD.

- If the modification is an element update, the SYSMOD must specify the RMID value as a prerequisite or specify BYPASS(ID) on the APPLY control statement. It is possible for SMP to change the RMID for an element during APPLY processing, but this occurs before the element update is processed. For example, if you specify a SYSMOD in the SELECT or GROUP operand that replaces the element and also specify a SYSMOD that updates the element, the SYSMOD containing the replacement is processed before the SYSMOD containing the update. This feature allows for the reapplication of user modifications and APARs (that contain updates to source modules or macros) concurrent with the application of a SYSMOD replacing those source modules or macros.

Automatic Reapplication of SYSMODs

It is possible that an applied SYSMOD might be selected for reapplication as a result of selection of a function SYSMOD being applied for the first time. This can occur if the modification is applicable to more than one function. For example, consider the following SYSMOD:

```
++PTF(UZ00001).
++VER(Z038) FMID(GVT3100).
++IF FMID(GVT3101) THEN REQ(UZ00001).
++VER(Z038) FMID(GVT3101).
++MOD(IFTABCD) DISTLIB(AOS99).
```

If PTF UZ00001 is already applied as service for function GVT3100, then SMP selects the first ++VER modification control statement and creates a CRQ entry for the ++IF modification control statement that follows the ++VER modification control statement. As a result, when function GVT3101 is selected for application, PTF UZ00001 is also selected because its version of module IFTABCD is at a higher service level than that of function GVT3101. This would be true even if function GVT3101 specified DELETE(GVT3100), which would result in the deletion of PTF UZ00001, because the deletion is logical until the deleting SYSMOD GVT3101 is successfully processed. The ++VER modification control statement that SMP selects for PTF UZ00001 is the one with the FMID(GVT3101) operand.

Processing a Service Updated Function SYSMOD

A function SYSMOD that is already applied can be reapplied to replace elements with higher service level versions of the elements. This is possible when the function SYSMOD package has been changed by a service update process. See "Service Updated Function SYSMODs" in Chapter 2 for a discussion of service updated SYSMOD construction.

Some elements in the service updated function SYSMOD are excluded from processing. This exclusion is normal and occurs when any one of the following conditions is true:

- The FMID in the element entry on the CDS is not the SYSMOD-ID of the SYSMOD and the FMID was not specified as a value of the VERSION operand on either the ++VER or element modification control statement. In the following example, the MOD entry for module IFTAAR has an FMID subentry of HVT1502. The module is not selected from function HVT1501 because it does not specify HVT1502 in either its ++VER or ++MOD modification control statement.

```
++FUNCTION(HVT1501).
++VER(Z038).
++MOD(IFTAAR).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1502

- The RMID in the element entry is not specified in the SUP operand of the appropriate ++VER modification control statement, but the SYSMOD entry for the RMID specified a direct or indirect PRE or SUP relationship with the SYSMOD specified in the RMID operand on the element modification control statement. This situation can occur if the service level of the element on the CDS is higher than that of the service updated function SYSMOD.

```
++FUNCTION(HVT1501).
++VER(Z038) SUP(UZ00001).
++MOD(IFTAAR) RMID(UZ00001).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1501
			RMID=UZ00002
	SYSMOD	UZ00002	FMID=HVT1501
			PRE=UZ00001
	SYSMOD	UZ00001	FMID=HVT1501

- The RMID in the element entry is not the FMID and/or there are UMIDs in the element entry, and no RMID operand is specified on the element modification control statement from the SYSMOD. This can occur when the service level of the element on the CDS is higher than that of the service updated function SYSMOD.

```
++FUNCTION(HVT1501).
++VER(Z038).
++MOD(IFTAAR).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1501
			RMID=UZ00002
		or	
	MOD	IFTAAR	FMID=HVT1501
			RMID=HVT1501
			UMID=AZ00001

- The RMID in the element entry is the same as the RMID operand value in the element modification control statement from the SYSMOD and there are UMIDs in the element entry that are not superseded by the SYSMOD.

```
++FUNCTION(HVT1501).
++VER(Z038).
++MOD(IFTAAR) RMID(UZ00001).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1501
			RMID=UZ00001
			UMID=AZ00001

If an individual element was not excluded from processing, it is selected from the service updated function SYSMOD when one of the following conditions is true:

- The FMID in the element entry on the CDS is specified as a value of the VERSION operand in either the ++VER or element modification control statement.

```
++FUNCTION(HVT1502).
++VER(Z038) VERSION(HVT1501).
++MOD(IFTAAR).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1501

- The RMID in the element entry on the CDS matches the RMID operand value on the corresponding element modification control statement and there are no UMID subentries in the element entry.

```
++FUNCTION(HVT1502).
++VER(Z038) SUP(UZ00001).
++MOD(IFTAAR) RMID(UZ00001).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1502
			RMID=UZ00001
			UMID=null

- The RMID in the element entry on the CDS matches one of the SUP operand values in the appropriate ++VER modification control statement, or equals the FMID, and any UMIDs are also specified in the SUP operand of the appropriate ++VER modification control statement.

```
++FUNCTION(HVT1502).
++VER(Z038) SUP(UZ00001,AZ00001,UZ00002).
++MOD(IFTAAR) RMID(UZ00002).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1502
			RMID=UZ00001
			UMID=AZ00001

- The RMID operand is not specified in the element modification control statement, the RMID subentry in the element entry on the CDS equals the SYSMOD-ID of the function (that is, the FMID), and there are no UMID subentries in the element entry.

```
++FUNCTION(HVT1502).
++VER(Z038) SUP(UZ00001).
++MOD(IFTAAR).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1502
			RMID=HVT1502
			UMID=null

If an element is neither excluded nor selected, the SYSMOD is terminated unless BYPASS(ID) is specified on the APPLY control statement. This condition occurs if the element is selected but the UMID subentries contain one or more SYSMOD-IDs that are not superseded or prerequisites of the function SYSMOD. The following shows this case:

```
++FUNCTION(HVT1502).
++VER(Z038) SUP(UZ00001).
++MOD(IFTAAR) RMID(UZ00001).
```

CDS Entries:	Type	Name	Subentries
	MOD	IFTAAR	FMID=HVT1502
			RMID=HVT1502
			UMID=AZ00005

APAR AZ00005 was applied to module IFTAAR but was not superseded by function HVT1502. The service level of the function is assumed to be higher because the RMID operand is specified on the ++MOD modification control statement and the RMID subentry in the MOD entry equals the SYSMOD-ID of the function. However, if the module is selected, the APAR is regressed; therefore, you must reapply the APAR concurrently or specify BYPASS(ID) on the APPLY control statement.

If you want SMP to concurrently apply the requisite SYSMODs at the same time that the service updated function SYSMOD is processed, you must specify the GROUP keyword on the APPLY control statement. As a result of specifying GROUP, SMP also selects requisite SYSMODs that were not previously applied.

If a service updated function SYSMOD is being applied for the first time, for each element selected from the SYSMOD, SMP sets the RMID subentry of each element to the RMID operand value, if present, on the associated element modification control statement.

RESTORE Processing

RESTORE processing removes SYSMODs that have been processed by APPLY from the target system libraries. SYSMODs that have been accepted into the distribution libraries cannot be restored.

RESTORE processing takes the version of the module, source module or macro that was in use before RESTORE processing from the distribution library and places it back into the target system library. In addition, required modules are reassembled and placed back into the target system libraries, and the CDS, SCDS, and CRQ are returned to the state they were in before the SYSMODs were applied.

SMP Data Set Validation

Before RESTORE processing can take place, SMP checks to ensure that the SMP data sets defined in the DD statements are valid and consistent. If any of the following checks fail, SMP issues an error message and RESTORE processing terminates:

- The CDS must have a SYSTEM entry.
- The CDS SYSTEM entry must indicate that it is processable by this version of SMP.
- The CDS SYSTEM entry must have a CDSID subentry.
- The ACDS must have a SYSTEM entry.
- The ACDS SYSTEM entry must indicate that it is processable by this version of SMP.
- The ACDS SYSTEM entry must have a CDSID subentry.
- The PTS must have a SYSTEM entry.
- The SREL subentry value in the CDS SYSTEM entry must exist as an SREL subentry in the PTS SYSTEM entry and match the SREL subentry in the ACDS SYSTEM entry.

SMP does not check to see if the CRQ and SCDS that you are using are compatible with the CDS defined. It is your responsibility to ensure that these data sets are the correct ones to be used.

Obtaining the PEMAX Value

The PEMAX value that SMP uses is the greatest of all the PEMAX values in the SYSTEM entries on the CDS, PTS, or ACDS, if available, or the default value of 500 subentries if no PEMAX values exist in the SYSTEM entries. See Chapter 4 for a discussion of PEMAX values.

SYSMOD Selection

When you specify SYSMODs to be restored, you should be aware that SYSMODs can be interrelated by the PRE, REQ, FMID, and SUP keywords on ++VER modification control statements and by the REQ keyword on ++IF modification control statements. Interrelated SYSMODs that have been applied but not accepted are considered members of a restore group.

A restore group for a SYSMOD consists of all the SYSMODs that have specified that SYSMOD in a PRE, FMID, REQ, or SUP operand in their ++VER modification control statements or in a REQ operand in their ++IF modification control statements and any SYSMODs that reference those SYSMODs in FMID, PRE, REQ, or SUP operands. In other words, all SYSMODs that have a direct or indirect dependency with a SYSMOD specified for RESTORE processing are considered part of the restore group.

RESTORE processing can be performed for SYSMODs that are not members of a restore group as well as all the SYSMODs in a restore group.

You can selectively restore SYSMODs, or you can restore SYSMODs in restore groups. These options are specified by the SELECT and GROUP keywords, respectively, on the RESTORE control statement. The following explains these options:

SELECT

Only the SYSMODs that you specified are selected. You should be aware that SYSMODs other than those specified may be required to synchronize the target system to the level of the distribution libraries. In SELECT mode, you must specify all of the SYSMODs that are requisites of a specified SYSMOD, or the specified SYSMOD is terminated.

GROUP

Each SYSMOD specified in the GROUP operand and all of their dependent SYSMODs are considered as a restore group and are selected for RESTORE processing.

SYSMOD Ineligibility

Certain conditions exist that can cause SYSMODs to be considered ineligible for RESTORE processing. These conditions cause SMP to terminate processing of the ineligible SYSMODs and issue messages to inform you of the error conditions.

The following conditions cause SMP to consider a SYSMOD as ineligible for RESTORE processing:

- An element being restored has a MODID in the element entry on the ACDS that does not have a corresponding SYSMOD entry on the CDS. This can occur if a SYSMOD has been is accepted without being applied and, as a result, the distribution library is at a higher function or service level than the target system library.
- The version of an element being restored is the same in the target system library as it is in the distribution library. This can occur if a SYSMOD is both applied and accepted.
- A SYSMOD that should have been selected for RESTORE processing was not specified in the SELECT operand list. This condition can occur if one of the SYSMODs specified in the list is part of a restore group that is not fully specified.

- The service level of an element in the distribution library is not the correct one. This can occur if several modifications to the same element are applied at different points in time, none of which were accepted, and the later modifications are the ones that are selected for RESTORE processing.

  Consider the following example. The ACDS shows that an element was last replaced on the distribution libraries by PTF UZ00001, but the CDS indicates that the last replacement to the element on the system was by PTF UZ00004. In addition, the element was also modified on the system by PTFs UZ00002 and UZ00003. The following figure shows the SYSMODs on the CDS and ACDS in service order:

```
     CDS SYSMODs              ACDS SYSMODs

    ┌────────────┐           ┌────────────┐
    │  UZ00001   │           │  UZ00001   │
    └────────────┘           └────────────┘
          │
    ┌────────────┐
    │  UZ00002   │
    └────────────┘
          │
    ┌────────────┐
    │  UZ00003   │
    └────────────┘
          │
    ┌────────────┐
    │  UZ00004   │
    └────────────┘
```

  If you specified: 'RESTORE GROUP(UZ00004).', PTFs UZ00002 and UZ00003 would not be considered part of the restore processing group since they are not dependent on PTF UZ00004. To correct the error, specify: 'RESTORE GROUP(UZ00002).'.

  When this condition is detected, SMP issues messages to inform you of the SYSMODs that must be restored along with the specified SYSMOD or accepted prior to restoring that SYSMOD.

- Ineligibility of a member of a restore group terminates processing for the entire group. This can occur both in GROUP and SELECT mode.

- Function SYSMODs that contained the DELETE keyword on the ++VER modification control statement used for APPLY processing are not eligible for RESTORE processing.

If a function SYSMOD is terminated for any of the above conditions, the RESTORE function is also terminated.

Inline JCLIN

If a SYSMOD that had inline JCLIN is restored, SMP attempts to restore the CDS entries affected by the JCLIN to their state before the SYSMOD was applied. This is done by accessing the related SYSMOD entry and associated BACKUP entries in the SCDS. For each BACKUP entry, SMP checks the corresponding CDS entry to ensure that the last modification to the CDS entry was for the SYSMOD being restored. If it was, then the entry is replaced from the SCDS BACKUP entry. If it was not, SMP issues a message to indicate that the entry was not replaced with the SCDS BACKUP entry and RESTORE processing continues. This condition can occur if you used UCLIN or JCLIN to update an entry after you applied the SYSMOD being restored, or if a subsequent SYSMOD was applied that updated the entry but did not have a dependency relationship with the SYSMOD being restored. The latter condition should only occur for LMOD entries.

As each entry is completed, SMP deletes the BACKUP entry. When all BACKUP entries have been processed, SMP deletes the related SYSMOD entry from the SCDS. This processing is done prior to updating target system libraries.

JCLIN processing occurs in the reverse order of application; that is, the latest update is restored first and the earliest update is restored last. The order is determined by the dependency relationships of the SYSMODs being restored.

Element Restoration

Each element modified by the SYSMODs being restored is altered by one of the following processes:

- If the modification being removed deleted the element using a DELETE operand on the element modification control statement, the element entry that was backed up on the SCDS is restored on the CDS as described earlier in "Inline JCLIN." The copy of the element is restored from the distribution library specified by the DISTLIB subentry of the element entry to the target system library, MTS, or STS.

- If the modification being removed was a complete replacement of the element, then the copy of the element in the distribution library is used to replace the element in the target system library. If the element has no copy in the distribution library, the element is deleted from the target system library and the element entry is deleted from the CDS. If the module is also a complete load module, then the load module is deleted and the LMOD entry in the CDS is deleted.

- If the modification being removed contained a ++MOD modification control statement with an LMOD operand, the MOD entry is restored with the copy from the SCDS as described earlier in "Inline JCLIN."

- If the modification being removed is an IMASPZAP, the target system library load modules are link edited as described above provided that other IMASPZAP modifications to the module are also being restored or have been accepted into the distribution library copy. If not, the associated SYSMOD and all related SYSMODs are terminated.

- If the modification being removed is a macro or source module update, the element is replaced with the copy of the element in the distribution library provided that any other updates to the element are also being restored or have been accepted into the distribution library copy. If not, SMP terminates the associated SYSMOD and all of its related SYSMODs. All assemblies are accomplished after this restoration completes.

Avoiding Termination of SYSMOD Processing

You can avoid certain error conditions that would terminate a SYSMOD by specifying the BYPASS(ID) operand on the RESTORE control statement. In this way, error conditions in the ID validation checking do not cause SYSMOD termination but are treated as warnings.

The first two conditions described earlier in "SYSMOD Ineligibility" can be bypassed using this option. However, in the first case, the target system library contains a version of the element that is probably functionally superior to that version being removed. This can cause the executable code in the target system library to be inoperable. In addition, SMP updates the element entry on the CDS to reflect the UMID and RMID subentry contents from the element entry on the ACDS. In this case, the SYSMOD entry might not exist on the CDS because the NOAPPLY keyword was probably used on the ACCEPT control statement; thus, the SYSMOD was never applied to the target system. You should avoid using the BYPASS(ID) option unless it is absolutely necessary.

Updating the MODID Subentries of Element Entries

The MODID fields of element entries are replaced with those from the ACDS element entry.

Supersede Processing

All SYSMOD entries that are superseded by SYSMODs being restored have the SUPBY subentries for those SYSMODs deleted. If all SUPBY subentries are deleted for a superseded SYSMOD entry, and the entry was created by APPLY processing, then the entry itself is deleted. As a result of restoring a SYSMOD that superseded a previously applied SYSMOD, CRQ entries that might have been ignored during APPLY processing may now be applicable. This condition is not acted upon by RESTORE processing. Therefore, subsequent APPLY processing may request requisite SYSMODs that are now applicable because of previously applied function SYSMODs.

Deleting Data from the CRQ

When a SYSMOD is successfully restored, any associated SYSMOD entry is deleted, and the related FMID entries are updated to remove the reference on the CRQ to the restored SYSMOD.

Updating the PTS

When a SYSMOD is successfully restored and the REJECT indicator is on in the PTS SYSTEM entry, the SYSMOD is also rejected from the PTS as described earlier in "REJECT Processing." Any temporary libraries associated with the SYSMOD are also deleted.

When a SYSMOD is successfully restored and the REJECT indicator in the PTS SYSTEM entry is off, the APPID subentry matching the CDSID in the CDS SYSTEM entry is deleted from the PTS SYSMOD entry to indicate that the element is no longer applied to the target system library represented by that CDS.

Deleting Entries from the CDS

When a SYSMOD is successfully restored, any associated CDS element entries are replaced with those from the ACDS. If an ACDS entry is not present, the CDS entry is deleted.

Deleting Members from the MTS and STS

When a successfully restored SYSMOD contains modifications to macros or source modules that were placed in the MTS or STS during APPLY processing, those members are deleted from the appropriate data set.

RESTORE Reports and Messages

RESTORE processing produces two reports. They are described in Chapter 6. If a function SYSMOD is selected but terminates, no reports are produced.

In addition, SMP may issue messages to inform you of error and warning conditions detected prior to producing the reports.

ACCEPT Processing

The ACCEPT process updates the distribution libraries or permanent user libraries and the ACDS.

In general, ACCEPT processing is very similar to APPLY processing, except that the SYSMODs are placed into permanent libraries, and the ACDS and ACRQ data sets are used rather than the CDS and CRQ data sets. Eligibility, selection, termination, and exception processing are handled in much the same way as they are during APPLY processing. Therefore, review "APPLY Processing" earlier in this chapter because the following text describes only the differences between the two.

SMP Data Set Validation

Before ACCEPT processing can take place, SMP checks to ensure that the SMP data sets defined in the DD statements are valid and consistent. If any of the following checks fail, SMP issues an error message and ACCEPT processing terminates:

- The CDS must have a SYSTEM entry.
- The CDS SYSTEM entry must indicate that it is processable by this version of SMP.
- The CDS SYSTEM entry must have a CDSID subentry.
- The ACDS must have a SYSTEM entry.
- The ACDS SYSTEM entry must indicate that it is processable by this version of SMP.
- The ACDS SYSTEM entry must have a CDSID subentry.
- The PTS must have a SYSTEM entry.
- The SREL subentry value in the CDS SYSTEM entry must exist as an SREL subentry in the PTS SYSTEM entry and match the SREL subentry in the ACDS SYSTEM entry.

SMP does not check to see if the SCDS that you are using is compatible with your CDS. It is your responsibility to ensure that the SCDS is the correct one to be used.

Obtaining the PEMAX Value

The PEMAX value that SMP uses is the greatest of all the PEMAX values in the SYSTEM entries on the CDS, PTS, or ACDS, if available, or the default value of 500 subentries if no PEMAX values exist in the SYSTEM entries. See Chapter 4 for a discussion of PEMAX values.

SYSMOD Selection Methods

To select SYSMODs eligible for processing, APPLY uses the PTS, which contains SYSMODs that might be applicable to your distribution libraries or permanent user libraries, the ACDS, which defines the environment of the distribution libraries and describes functions and service already accepted, the ACRQ, which contains data from ++IF modification control statements of SYSMODs previously accepted, and the CNTL input stream, which contains the ACCEPT control statement with options.

SYSMOD selection is similar to selection in APPLY processing, except that the SYSMODs that are selected for mass acceptance are those that have been applied but not accepted. This means that the CDS is required for ACCEPT processing. To bypass the requirement for the CDS, the NOAPPLY operand must be specified on the ACCEPT control statement. Specifying 'ACCEPT NOAPPLY.' causes SMP to use the same selection criteria that it uses during APPLY processing for 'APPLY.'

Inline JCLIN

When a SYSMOD with associated JCLIN is accepted, the related SYSMOD and BACKUP entries on the SCDS are deleted. This processing occurs only when the SYSMOD is successfully processed by ACCEPT and the NOAPPLY keyword was not specified on the ACCEPT control statement.

JCLIN processing against the ACDS entries is not necessary because the affected entry types do not require any data, except DISTLIB changes, to be carried across from the associated CDS entries during ACCEPT processing. See 'DISTLIB Operand Checking' below.

DISTLIB Operand Checking

Inline JCLIN processing is not done against the ACDS. If the SYSMOD being processed changes the DISTLIB for any elements, the CDS element entries must reflect those changes, or you must make the necessary changes to the ACDS using UCLIN processing before ACCEPT processing.

The ACDS element entry is updated with the DISTLIB subentry from the CDS when the following conditions are met:

- The NOAPPLY keyword is not specified on the ACCEPT control statement
- The SYSMOD that changed the DISTLIB has been processed by APPLY processing
- The DISTLIB subentry in the ACDS element entry is not the same as the corresponding DISTLIB subentry in the CDS
- The SYSMOD-ID of the SYSMOD being processed appears in any of the FMID, RMID, or UMID subentries in the CDS element entry
- The DISTLIB value in the CDS element entry is the same as the DISTLIB value in the element modification control statement

ACCEPT Indication in the PTS SYSMOD Entries

For each SYSMOD that was not terminated, an ACCID subentry is added to the associated SYSMOD entry in the PTS using the CDSID from the ACDS SYSTEM entry, unless one already exists for that CDSID.

Data Set Use

The ACDS and ACRQ are used rather than the CDS and CRQ.

The CDS is required unless the NOAPPLY keyword is specified on the ACCEPT control statement.

Load modules in temporary libraries are always copied to the appropriate distribution libraries using IEBCOPY.

Members stored in the MTS or STS might be deleted if the following conditions are met:

- The NOAPPLY operand was omitted on the ACCEPT control statement.
- The SAVEMTS and/or SAVESTS indicators in the CDS SYSTEM entry are reset.
- The MODID subentries in the CDS element entry match the MODID subentries in the ACDS element entry.

Using DISTSRC, ASSEM, and DISTMOD Operands

Because there are no ASSEM entries on the ACDS, SMP only checks the SRC entries to see if there is an entry for the modules in the ASSEM operand list.

DLIB and LMOD entries do not exist on the ACDS; therefore, the SYSLIB subentries of the SRC and MOD entries have no meaning.

The copy of the source module is obtained from the distribution library referenced by the DISTSRC operand.

Assemblies of the source modules are not done if a distribution library for the resulting modules does not exist.

The object text is link edited into the distribution library referenced by the DISTMOD operand, if specified, or the distribution library referenced by the DISTLIB subentry of the associated MOD entry.

Deletion of Elements

The SCDS is not used to back up element entries that are deleted during ACCEPT processing. Element entries are deleted from the ACDS and the distribution libraries. If NOAPPLY was not specified and there is no inline JCLIN for a SYSMOD, the BACKUP entries for those elements that are deleted during APPLY processing are deleted from the SCDS.

Deleting SYSMOD Entries from the PTS

For each SYSMOD successfully processed by ACCEPT, the associated SYSMOD and MCS entries on the PTS are deleted. Temporary libraries are deleted if the PURGE indicator is set in the PTS SYSTEM entry and the NOAPPLY keyword was not specified on the ACCEPT control statement.

Processing APARs and User Modifications

In order to process ++APAR and ++USERMOD SYSMODs, you must specify the keywords APARS and USERMODS, respectively, on the ACCEPT control statement. This is required to avoid inadvertent acceptance of corrective service and user modifications into the distribution libraries.

UCLIN Processing

UCLIN processing adds, deletes, and changes entries and entry data in the ACDS, ACRQ, CDS, CRQ, MTS, PTS, SCDS, and STS data sets.

UCLIN processing is invoked by specifying the UCLIN control statement, followed by the UCL statements and the ENDUCL control statement.

The PTS SYSTEM Entry

UCLIN processing is used to build the SYSTEM entry on the PTS. The following subentries and indicators can be specified:

- The names of programs invoked by SMP to assemble, copy, compress, link edit, perform IOSUP, IMASPZAP, and text update.
- The ddnames of the SYSOUT data sets used by these programs.
- The return code values that are compared with the codes returned from these programs in order to control SMP processing.
- Additional processing parameters used by these programs.
- Space parameters and high level data set name qualifiers for data sets allocated during RECEIVE processing for relative files.
- The SRELs of the system(s) being maintained.
- FMIDs for function SYSMODs.
- A PEMAX value used by SMP functions.
- The PURGE indicator for controlling the deletion of SYSMODs from the PTS during ACCEPT processing.
- The REJECT indicator for controlling the deletion of SYSMODs from the PTS during RESTORE processing.

The ACDS and CDS SYSTEM Entries

The ACDS and CDS SYSTEM entries are created and modified by UCLIN processing. The following subentries and indicators can be specified:

- The CDSID of the ACDS or CDS that identifies the distribution library or target system.
- The SREL of the system being maintained.
- A PEMAX value used by SMP functions.
- The NUCID, which is the identifier of the saved nucleus stored during APPLY processing.
- The SAVEMTS indicator for controlling the deletion of members from the MTS during ACCEPT processing.
- The SAVESTS indicator for controlling the deletion of members from the STS during ACCEPT processing.

The last three are only meaningful for the CDS SYSTEM entry.

Entry Update Indication in ACDS and CDS Entries

When an entry is updated by UCLIN, the character string 'UCLIN ' is placed in the UPDID subentry. This prevents loss of updates, as described in the following text.

When an entry is updated with inline JCLIN, a back-up copy of the entry is saved on the SCDS. If you then update the CDS entry using UCLIN, the character string 'UCLIN ' is placed in the CDS UPDID subentry. If you then attempt to restore the SYSMOD, the BACKUP entry in the SCDS is not used to restore the entry because the UPDID subentry value does not match the SYSMOD-ID of the SYSMOD being restored.

When ACDS and CDS entries are listed, the field labelled 'LAST UPDATE' shows the content of the UPDID subentry.

UCLIN Messages

UCLIN processing produces messages on SMPOUT.

JCLIN Processing

JCLIN processing adds and changes entries and entry data in a CDS by analyzing JCL input streams.

Types of CDS Entries Affected

JCLIN processing creates and updates ASSEM, DLIB, LMOD, MAC, and MOD entries on the CDS as follows:

- ASSEM entries are created or replaced when JCL contains an assembler job step. The name of the ASSEM entry is determined from the DSN operand value of the SYSPUNCH DD statement, when the EXEC statement is for an assembler program, or the name specified in the MOD operand, when the EXEC statement is for the ASMS procedure. The ASSEM entry includes the source text following the SYSIN DD statement.

- DLIB entries are created when the JCL contains a COPY job step that totally copies members from a distribution library to a target system library. A library is considered to be totally copied when no SELECT control statements are encountered. EXCLUDE control statements may be present in the control statement input. The name of the DLIB entry is taken from the INDD operand value on the COPY control statement. The SYSLIB subentry is the operand value of the OUTDD operand on the COPY control statement. This control statement follows the SYSIN DD statement in the JCLIN input data.

- LMOD entries are created or updated when JCL contains a link edit or COPY job step. For link edit steps, an LMOD entry is created or updated for the name specified in the NAME linkage editor control statement following the SYSIN DD statement in the JCLIN input data or the member name in the DSNAME operand of the SYSLMOD DD statement.

  The library name is found in the SYSLMOD DD statement, or, if the LINKS procedure is specified on the EXEC statement, is the value of the NAME operand. All linkage editor control statements except INCLUDE and NAME are included in the data portion of the LMOD entry. If the LMOD entry already exists, any control statements except CHANGE and REPLACE are replaced with those in the JCLIN input stream. The parameters in the PARM operand of the EXEC statement are interpreted and the corresponding indicators are set in the LMOD entry.

An entry is also created when JCL contains a COPY job step that selectively copies members from a distribution library to a target system library. Each member specified in the MEMBER operand of the SELECT control statement causes the creation or update of an LMOD entry with a SYSLIB subentry equal to the operand value of the OUTDD operand on the COPY control statement.

- MAC entries are created or updated when JCL contains an assembler job step. The assembler input data is scanned for macros that expand or are copied via the COPY assembler statement. An assembler instruction that is greater than five characters in length is considered to be a macro name. Each one encountered causes a MAC entry to be created or updated, and adds a GENASM subentry to the entry for the ASSEM entry that is created or replaced for the same job step.

- MOD entries are created or updated when JCL contains a link edit or COPY job step. For link edit steps, a MOD entry is created or updated when an INCLUDE control statement is encountered in the SYSIN DD data. The ddname following the INCLUDE operand is for the distribution library and becomes the DISTLIB subentry of the MOD entry. The name in parentheses following the ddname is the name of the MOD entry itself. An LMOD subentry is added to the MOD entry for the load module specified in the NAME control statement.

  For COPY steps, a MOD entry is created or updated when selective copies of distribution library members to a target system library are required. For each name specified in the MEMBER operand of the SELECT control statement, a MOD entry is created or updated with that name, an LMOD subentry is added to the entry, and the ddname specified in the INDD operand of the COPY statement becomes the DISTLIB subentry. MAC entries are not created from IEBCOPY job step input.

  If the SELECT control statement renamed the load module (that is, 'SELECT MEMBER=(modname,lmodname,R))' is specified) then the lmodname is used to create the LMOD entry and to add the LMOD subentry to the MOD entry.

  When a COPY statement is encountered in an IEBCOPY job step that has more than one value for the INDD operand, only the first value is used and becomes the DISTLIB subentry for any entries created or updated.

Inline JCLIN and the JCLIN Control Statement

To support inline JCLIN, JCLIN processing creates BACKUP entries on the SCDS for the CDS entries affected by JCLIN input data. Each entry created or updated has the SYSMOD-ID of the associated SYSMOD placed in the UPDID subentry. This field is checked during RESTORE processing to ensure that the BACKUP entry on the SCDS can replace the entry on the CDS without losing subsequent updates.

BACKUP entries on the SCDS are not created when the JCLIN control statement is used to invoke JCLIN processing. For this processing, each entry created or updated has the UPDID subentry set to the character string 'JCLIN '. This prevents the loss of updates, as described in the following text.

When an entry on the CDS is updated from the JCL input data within a SYSMOD, a BACKUP entry is created on the SCDS and the UPDID subentry is set to the SYSMOD-ID. A subsequent update to the entry on the CDS using the JCLIN control statement places the 'JCLIN ' character string in the UPDID subentry. If you then request SMP to restore the SYSMOD, the 'JCLIN ' character string prevents SMP from replacing the updated entry on the CDS with the BACKUP entry on the SCDS.

When CDS entries are listed, the contents of the UPDID subentry are shown for the 'LAST UPDATE' field.

JCLIN Messages

JCLIN processing produces messages on SMPOUT.

RETRY Processing

RETRY provides a STAE/ESTAE environment for APPLY, ACCEPT, and RESTORE functions which allow for the compressing of partitioned data sets that are the target libraries for a UTILITY and which become full during service or function installation. After the compress, the failing UTILITY operation will be re-executed. The types of ABENDS for which the COMPRESS operation will be attempted (PROVIDED THAT THE DATA SET IS ELIGIBLE) are a B37-04, D37-04, and a E37-04. These types of ABENDS are referred to as 'X37' in the remainder of the X37 RETRY function description.

The RETRY processing is controlled by the user:

- A list of DD names eligible for X37 RETRY processing is added to the CDS system entry for the APPLY/RESTORE functions, and to the ACDS system entry for the ACCEPT function. The value 'ALL' in the list indicates that all UTILITY target DD names are eligible for RETRY.

- The RETRY processing is the default mode of processing for APPLY, ACCEPT, and RESTORE, provided that a list of eligible DD names is available in the appropriate system entry. The user may prevent the RETRY processing by specifying the keyword RETRY(NO) on the APPLY ACCEPT or RESTORE control statement or my removing all DD names from the appropriate system entry.

  Since the UTILITY routines (IEBCOPY, IEWL, IEBUPDTE, ETC) are attached, rather then linked to, SMP is not terminated if the UTILITY fails during RETRY processing. Instead, the failing UTILITY is detached and the SMP function is terminated. (RC=12)

  The user may specify the name of the program to be used for COMPRESS at recovery. This is specified in the SMPPTS system entry; If it is not specified the default of IEBCOPY is used.

  An additional SMP DD statement (SYSUT4) is required for use as the SYSIN data set for the RETRY compress program. If RETRY is requested or defaulted on an APPLY ACCEPT or RESTORE, SMP is terminated if the SYSUT4 DD statement is not present. The space allocation for this sequential data set need be no more than a single track since it contains only a single eighty (80) BYTE control statement suitable for IEBCOPY. The LRECL for the SYSUT4 data set is 80 and the BLKSIZE may be any multiple of 80.

<u>USER EXIT</u> 2
Your may supply a user exit that is invoked before COMPRESS and RETRY are attempted. This user exit allows you to stop RETRY processing(see 'User Exit 2' in Chapter 4.)

This chapter provides information to assist you in the initialization and execution of SMP and in the preparation of user routines and modifications. The chapter is organized into the following topics:

- Creating and modifying SMP primary data sets
- Storage estimates
- Executing SMP
- User written exit routines
- User modifications

Creating and Modifying SMP Primary Data Sets

SMP controls the processing of SYSMODs by examining the contents of the primary data sets, specifically the ACDS, ACRQ, CDS, CRQ, PTS, and SCDS. The ACRQ, CRQ, MTS, STS, and SCDS are initially empty and need only be allocated as partitioned data sets as described in 'SMP Data Set Requirements' later in this chapter. You must, however, update the ACDS, CDS, and PTS following allocation.

Creating the CDS and CRQ

When you create a target system by performing a system generation (SYSGEN), you can create the CDS using the ACDS and the CRQ using the ACRQ.

The ACDS/ACRQ reflects the status and contents of the distribution libraries and is distributed by IBM with some of its software products. The distribution libraries are used during SYSGEN to build a target system; however, you may be required to receive and accept SYSMODs into the distribution libraries prior to SYSGEN. In this case, SMP updates the ACDS/ACRQ to reflect the SYSMODs that were accepted.

To create the CDS, you must copy all of the entries on the ACDS to the CDS. To create the CRQ you must copy the ACRQ to the CRQ. You then perform Stage I SYSGEN processing. Using the output from Stage I SYSGEN processing as input to

SMP, you execute the SMP JCLIN control statement to create and update CDS entries that describe the structure of your target system. This is described in 'JCLIN Processing' in Chapter 3. When JCLIN processing completes, you can use the UCL SYS statement to make any required changes to the CDSID and NUCID subentries in the CDS SYSTEM entry.

Null CDS and ACDS

It may be necessary to allocate the CDS and/or ACDS and create the initial entries yourself. This is referred to as a null CDS or ACDS. The UCL SYS statement description in Chapter 7 provides an explanation of what subentries and indicators are needed in the SYSTEM entry when these data sets are created.

Null PTS

Before you can receive any SYSMODs, you must allocate the PTS and create the SYSTEM entry using the UCL SYS statement. The UCL SYS statement description in Chapter 7 provides an explanation of the required subentries and the default indicator settings for this entry.

PEMAX Values

SMP uses the PEMAX (PTF entry maximum count) to allocate the storage required to read in a single entry from the ACDS, CDS, PTS, or SCDS. Each entry read consists of all of its subentries. For example, a SYSMOD entry on the CDS has subentries for every operand specified on the ++VER modification control statement determined to be applicable during APPLY processing, and for every element modification control statement in that SYSMOD. The SYSMOD entry may increase in size during subsequent APPLY processing if other SYSMODs specify that SYSMOD in a ++VER modification control statement.

You can specify a PEMAX value from 50 to 9999 in the ACDS, CDS, or PTS SYSTEM entries using the UCL SYS statement. The default value for PEMAX, if it is not specified in the ACDS, CDS, or PTS SYSTEM entries, is 500.

The criteria used by SMP to choose a PEMAX value when each function is invoked is described in each major SMP function in Chapter 3. If, during the processing of entries, SMP determines that an entry exceeds the largest PEMAX value specified in the SYSTEM entries at the beginning of processing of that function, SMP issues message HMA219 and terminates the processing of the SYSMOD, the entry, or the function. You must then modify the PEMAX subentry in one or more of the SYSTEM entries to increase the size, assuming that the PEMAX used was not 9999.

Storage Estimates

Prior to the execution of SMP, space must be allocated on various storage devices and reserved in main storage. This section describes recommended minimum SMP primary data set allocations and an algorithm for determining main storage requirements.

SMP Program Requirements

SMP normally resides in SYS1.LINKLIB. The SMP program must be in an authorized library and must be authorized itself.

SMP requires storage for program execution. The following algorithm will help you in determining the amount of storage needed by the APPLY, RESTORE, and ACCEPT functions with directories in storage mode of operation:

```
     450K  (Size of SMP program)
 +      2 x largest ACDS/CDS/SCDS blocksize
 +      2 x largest MTS/PTS/STS blocksize
 +      2 x largest ACRQ/CRQ blocksize
 +      2 x largest WRK1/WRK2/WRK3 blocksize
 +      2 x WRK4 blocksize
 +      1 x largest LKLIB/TXLIB blocksize
 + 4 x 9 x largest PEMAX value
 +    252 x number of directory blocks in ACDS/CDS
 +      1 x largest size of programs invoked by SMP
 +      1 x calculated storage for processing SYSMODs
 ---------
 TOTAL SIZE
```

To determine the sizes of the programs invoked by SMP, refer to the following publications:

- OS/VS Linkage Editor and Loader
- OS/VS and DOS/VS Assembler Language
- OS/VS Utilities
- OS/VS1 Service Aids or OS/VS2 System Programming Library: Service Aids

To calculate the amount of storage needed to process a set of SYSMODs, the following algorithm can be used:

```
    124 * number of SYSMODs being processed
  + 160 * number of elements in SYSMODs being processed
  + 160 * number of ASSEM and SRC entries referenced by
          macros modified by SYSMODs being processed
  + 160 * number of ASSEM and SRC entries that are to
          be assembled
  +  64 * number of unique load modules that are to be
          link edited or modified by IMASPZAP
  +   8 * number of DELETE, NPRE, PRE, REQ, SUP, and
          VERSION operands in the ++VER modification
          control statements in SYSMODs being processed
  +  16 * number of REQ operands in ACRQ/CRQ and ++IF
          modification control statements in SYSMODs
          being processed that are applicable
          to your environment
  +  22 * number of REQ operands in ++IF modification
          control statements in SYSMODs being processed
  ------------------------------------------------------
  = Total storage for processing set of SYSMODs
```

The algorithm is based on approximate sizes of internal entries used during APPLY, RESTORE, and ACCEPT processing. If you are doing RESTORE processing, do not include calculations for the REQ operand from the CRQ or the REQ operands from the SYSMODs being processed.

The maximum amount of storage that SMP attempts to obtain for internal entries is 800K. If your calculations show that the processing of a set of SYSMODs might exceed this amount, you should process smaller subsets of that set.

The following is an example of this algorithm based on the SYSMODs shown in Figure 9.

```
SMPPTS   M.C.S. ENTRIES
 NAME

AZ00124   M.C.S. ENTRIES  = ++ APAR(AZ00124).
                            ++ VER(Z038) FMID(GXY1000) PRE(UZ00010).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).

GXY1000   M.C.S. ENTRIES  = ++ FUNCTION(GXY1000) FILES(4).
                            ++ VER(Z038).
                            ++ JCLIN RELFILE(1).
                            ++ MAC(MACRO1) DISTLIB(DLIBMAC1) RELFILE(2).
                            ++ MAC(MACRO2) DISTLIB(DLIBMAC1) RELFILE(2).
                            ++ MOD(MOD001) DISTLIB(DLIB01) RELFILE(3).
                            ++ MOD(MOD002) DISTLIB(DLIB01) RELFILE(3).
                            ++ SRC(MOD001) DISTLIB(DLIBSRC1) RELFILE(3).
                            ++ SRC(MOD002) DISTLIB(DLIBSRC1) RELFILE(3).

HXY1010   M.C.S. ENTRIES  = ++ FUNCTION(HXY1010) FILES(4).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00123,AZ00124) PRE(UZ00010).
                            ++ JCLIN RELFILE(1).
                            ++ MAC(MACRO1) DISTLIB(DLIBMAC2) RELFILE(2).
                            ++ MAC(MACRO3) DISTLIB(DLIBMAC2) RELFILE(2).
                            ++ MOD(MOD001) DISTLIB(DLIB02) RELFILE(3).
                            ++ MOD(MOD003) DISTLIB(DLIB02) RELFILE(3).
                            ++ SRC(MOD001) DISTLIB(DLIBSRC2) RELFILE(3).
                            ++ SRC(MOD003) DISTLIB(DLIBSRC2) RELFILE(3).

UZ00010   M.C.S. ENTRIES  = ++ PTF(UZ00010).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00123).
                            ++ MOD(MOD001) DISTLIB(DLIB01).
                            ++ MOD(MOD002) DISTLIB(DLIB01).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
                            ++ SRCUPD(MOD002) DISTLIB(DLIBSRC1).

UZ00012   M.C.S. ENTRIES  = ++ PTF(UZ00012).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00124) PRE(UZ00010).
                            ++ MOD(MOD001) DISTLIB(DLIB01).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).

UZ00014   M.C.S. ENTRIES  = ++ PTF(UZ00014).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00136,UZ00012) PRE(UZ00010).
                            ++ IF FMID(HXY1010) THEN REQ(UZ00015).
                            ++ MOD(MOD001) DISTLIB(DLIB01).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
                            ++ MACUPD(MACRO2) DISTLIB(DLIBMAC1).

UZ00015   M.C.S. ENTRIES  = ++ PTF(UZ00015).
                            ++ VER(Z038) FMID(HXY1010) SUP(AZ00136) REQ(UZ00014).
                            ++ MOD(MOD001) DISTLIB(DLIB02).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC2).

XY10001   M.C.S. ENTRIES  = ++ USERMOD(XY10001).
                            ++ VER(Z038) FMID(GXY1000).
                            ++ IF FMID(HXY1010) THEN REQ(XY10101).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).

XY10101   M.C.S. ENTRIES  = ++ USERMOD(XY10101).
                            ++ VER(Z038) FMID(HXY1010).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC2).
```

Figure 9. Sample SYSMOD List

There are nine SYSMODs in this sample list. If you wanted to apply these SYSMODs to your target system in mass mode (that is, you specify 'APPLY.'), you would use the following values in your calculations of the amount of storage needed to process this set of SYSMODs:

```
   124 *  9 = 1116   The number of SYSMODs is 9.
 + 160 * 26 = 4160   The total element count is 26.
 + 160 *  3 =  480   ASSEM1, ASSEM2, and MOD002 are
                     referenced by modified macros.
 + 160 *  4 =  640   ASSEM1, ASSEM2, MOD001, and
                     MOD002 are assembled.
 +  64 *  2 =  128   LMOD1 and MOD003 are load modules.
 +   8 * 12 =   96   There are 12 ++VER modification
                     control statement operands.
 +  16 *  2 =   32   There are two applicable REQ operands
                     on the ++IF modification control
                     statements and the CRQ.
 +  22 *  2 =   44   The total REQ operands from ++IF
                     modification control statements is two.
 -----------------------------------------------------
             6696    or 6.7K total storage needed
                     for this set of SYSMODs
```

This value of 6.7K is used in the algorithm to determine the amount of storage required by APPLY, RESTORE, and ACCEPT in a directories-in-storage mode. This is shown in the following example, which assumes that the blocksize for all SMP data sets is 3200, that the PEMAX value is 500, that 3000 CDS directory blocks are used, and that the size of the largest program invoked by SMP is 184K. The total storage needed for processing is calculated as follows:

```
     450K        = 450.0K
 +      2 x 3200 =   6.4K
 +      2 x 3200 =   6.4K
 +      2 x 3200 =   6.4K
 +      2 x 3200 =   6.4K
 +      2 x 3200 =   6.4K
 +      1 x 3200 =   3.2K
 + 4 x 9 x  500  =  18.0K
 +    252 x 3000 = 756.0K
 +      1 x 184K = 184.0K
 +      1 x 6.7K =   6.7K
 -----------------------
 TOTAL SIZE       1449.9K
```

The REGION parameter on the JOB or STEP statement must be at least '1450K' according to the above calculation. If possible, you should specify an even greater REGION size.

If you are executing SMP on a system with insufficient storage for processing the ACDS or CDS directory in storage, you must use the DIS(NO) option on the SMP control statements. See 'Directories in Storage' in this chapter for additional information.

SMP Nucleus Storage Requirements

The nucleus data set (SYS1.NUCLEUS) must be large enough to contain three copies of the IEANUC01 member to allow for link edits required for modules within IEANUC01 that are modified. With this minimum allocation:

- If you maintain a backup copy of the nucleus in your target system, the nucleus data set must be compressed after each modification to it.

- If you do not maintain a backup copy of the nucleus in your target system, the nucleus data set must be compressed after every second modification to it.

SMP Data Set Requirements

SMP has primary data sets that you allocate immediately after system generation, and secondary data sets that are defined by DD statements when you execute SMP.

For detailed descriptions of the use and purpose of each data set, see Chapter 9.

<u>Primary Data Set Requirements</u>

The primary data sets that you allocate immediately after system generation are:

• SMPACDS	• SMPCRQ	• SMPPTS
• SMPACRQ	• SMPLOG	• SMPSCDS
• SMPCDS	• SMPMTS	• SMPSTS

If the SMPACDS is provided with the distribution libraries, you may have to reallocate the SMPACDS to satisfy storage requirements.

Figures 10 and 11 provide guidelines for estimating the storage requirements of the primary SMP data sets. Figure 10 lists the number of tracks of direct access storage you can initially estimate for the primary data sets. For performance considerations, the SMPACDS and SMPCDS should be allocated in cylinders. This figure also shows the number of tracks needed in the LINKLIB data set for the SMP program. Figure 11 lists the directory block allocation and data set organization for the primary data sets when a 3330 storage device is used. All the numbers in both figures are for the base level system only.

		Track Requirements by Device				
		2305	2314/2319	3330/3333	3340	3350
Data Set Names	SMPACDS	41 *	409 *	225 *	367 *	155 *
		62 +	614 +	338 +	551 +	231 +
	SMPACRQ	11 *	109 *	60 *	98 *	41 *
		16 +	164 +	90 +	147 +	62 +
	SMPCDS	55 *	545 *	300 *	489 *	206 *
		82 +	818 +	450 +	734 +	309 +
	SMPCRQ	11 *	109 *	60 *	98 *	41 *
		16 +	164 +	90 +	147 +	62 +
	SMPLOG	70	140	76	126	52
	SMPMTS	52	104	57	94	39
	SMPPTS	580 *	1153 *	634 *	1040 *	433 *
		870 +	1730 +	950 +	1560 +	650 +
	SMPSCDS	5 *	55 *	30 *	49 *	21 *
		8 +	82 +	45 +	73 +	31 +
	SMPSTS	52	104	57	94	39
	LINKLIB	46	93	51	91	35

* - VS1 Systems
\+ - VS2 Systems

Figure 10. SMP Primary Data Set Requirements in Tracks

Data Set Name	3330 Directory Blocks	Data Set Organization
SMPACDS	1500 (VS1) 2250 (VS2)	PDS
SMPACRQ	350 (VS1) 500 (VS2)	PDS
SMPCDS	2000 (VS1) 3000 (VS2)	PDS
SMPCRQ	350 (VS1) 500 (VS2)	PDS
SMPLOG	N/A	Sequential
SMPMTS	50	PDS
SMPPTS	500	PDS
SMPSCDS	75 (VS1) 100 (VS2)	PDS
SMPSTS	50	PDS

Figure 11. SMP Primary Data Set Organization and Directory Block Allocation on a 3330 Device

Secondary Data Set Requirements

The remaining, or secondary, SMP data sets are defined by DD statements you provide when executing an SMP job. They are:

- SMPCNTL
- SMPJCLIN
- SMPLIST
- SMPOUT
- SMPPTFIN
- SMPRPT
- SMPTLIB
- SMPWRK1
- SMPWRK2
- SMPWRK3
- SMPWRK4
- SMPWRK5
- SYSLIB
- SYSPRINT
- SYSUT1
- SYSUT2
- SYSUT3
- distlib
- lklib
- tgtlib
- txlib

Executing SMP

SMP is executed as a job running under the operating system. You must specify JCL statements to define the job and the data sets to be used by SMP to perform its functions.

JCL Required for SMP

The JCL statements required for SMP include the JOB, EXEC and DD statements:

- The JOB statement describes your installation-dependent parameters. You may also specify the REGION parameter to set the size of the region or partition in which SMP executes.

- The EXEC statement must specify PGM=HMASMP or the name of your cataloged procedure. The following parameters can be specified in the PARM operand of the EXEC statement:

  'DATE=date'

  where "date" can be:

  U or IPL - to use the IPL date of the system.

  REPLY - to request the date from the operator. SMP issues message HMA399 as a result.

  yyddd - to specify a specific date where "yy" is the year and "ddd" is the day of the year.

  If this parameter is not specified, the IPL date of the system is used.

  'FMID=sysmodid'

  where "sysmodid" is the identifier of a function-like SYSMOD that is packaged using the techniques supported in earlier versions of SMP. See Appendix C for a description of the usage of this parameter.

- The DD statements specify the data sets that are required by or are optional for the SMP function. See Chapter 9 for information about the data sets and the ddnames associated with each.

SMP Cataloged Procedure

A cataloged procedure is a set of job control statements that are placed in a partitioned data set and subsequently retrieved by naming the procedure in an EXEC statement. The following is a sample SMP cataloged procedure that can be placed in a cataloged procedure library and used during the execution of an SMP job:

```
//SMPJOB    PROC
//SMPSTEP   EXEC PGM=HMASMP
//SMPOUT    DD   SYSOUT=A
//SMPRPT    DD   SYSOUT=A
//SMPLIST   DD   SYSOUT=A
//SYSPRINT  DD   SYSOUT=A
//SMPLOG    DD   DSN=SYS1.SMPLOG,DISP=MOD
//SMPCDS    DD   DSN=SYS1.SMPCDS,DISP=OLD
//SMPCRQ    DD   DSN=SYS1.SMPCRQ,DISP=OLD
//SMPACDS   DD   DSN=SYS1.SMPACDS,DISP=OLD
//SMPACRQ   DD   DSN=SYS1.SMPACRQ,DISP=OLD
//SMPSCDS   DD   DSN=SYS1.SMPSCDS,DISP=OLD
//SMPPTS    DD   DSN=SYS1.SMPPTS,DISP=OLD
//SMPMTS    DD   DSN=SYS1.SMPMTS,DISP=OLD
//SMPSTS    DD   DSN=SYS1.SMPSTS,DISP=OLD
//SMPWRK1   DD   UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
                 DCB=BLKSIZE=3360
//SMPWRK2   DD   UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
                 DCB=BLKSIZE=3360
//SMPWRK3   DD   UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
                 DCB=BLKSIZE=3200
//SMPWRK4   DD   UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
                 DCB=BLKSIZE=3200
//SMPWRK5   DD   UNIT=SYSDA,SPACE=(CYL,(2,1,5)),DISP=(,DELETE),
                 DCB=BLKSIZE=7294
//SYSLIB    DD   DSN=SYS1.SMPMTS,DISP=OLD
//          DD   DSN=SYS1.MACLIB,DISP=OLD
(Include additional system macro library DD statements
here in the SYSLIB concatenation for assemblies)
//          DD   DSN=SYS1.AMACLIB,DISP=OLD
//          DD   DSN=SYS1.AMODGEN,DISP=OLD
(Include additional distribution macro library DD statements
here in the SYSLIB concatenation for assemblies)
//MACLIB    DD   DSN=SYS1.MACLIB,DISP=OLD
//LINKLIB   DD   DSN=SYS1.LINKLIB,DISP=OLD
(Include additional DD statements here for target system
library data sets containing modules, macros, or source modules)
//AOSB3     DD   DSN=SYS1.AOSB3,DISP=OLD
//AOSC5     DD   DSN=SYS1.AOSC5,DISP=OLD
//ASAMPLIB  DD   DSN=SYS1.ASAMPLIB,DISP=OLD
//AMACLIB   DD   DSN=SYS1.AMACLIB,DISP=OLD
//AMODGEN   DD   DSN=SYS1.AMODGEN,DISP=OLD
//AGENLIB   DD   DSN=SYS1.AGENLIB,DISP=OLD
(Include additional DD statements here for distribution
library data sets containing modules, macros, or source modules)
//SMPCNTL   DD   DDNAME=SYSIN
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,DELETE)
//SYSUT2    DD   UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,DELETE)
//SYSUT3    DD   UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,DELETE)
//SYSUT4    DD   UNIT=SYSDA,SPACE=(TRK,(1,1)),DISP=(,DELETE)
(Include additional DD statements here for any required
LKLIB and TXLIB data sets)
```

Notes:

- For a RECEIVE procedure, add an SMPPTFIN DD statement to describe the input containing SYSMODs.
- If SYSMODs being received are in relative file format, add an SMPTLIB DD statement to define storage devices to be used.
- The SMPACDS DD statement is not required for an APPLY, RECEIVE, or REJECT procedure.
- The SMPCDS DD statement is not required for an ACCEPT, RECEIVE, or REJECT procedure.
- The SMPMTS DD statement is required for modifications to macros not residing in a target system library.
- The SMPSTS DD statement is required for modifications to source modules not residing in a target system library.
- The SYSLIB DD statement concatenation is required if SMP performs an assembly as a result of a modification to a macro or source module.
- The DD statements for target system data sets are not required for a RECEIVE, REJECT, or ACCEPT procedure.
- The DD statements for distribution library data sets are not required for a RECEIVE, REJECT, or APPLY procedure.

Including the Required System Programs

SMP requires that you supply the following system programs. If these programs are not available, SMP is terminated. These programs are:

- Assembler (See OS/VS and DOS/VS Assembler Language)
- Linkage Editor (See OS/VS Linkage Editor and Loader)
- IEBCOPY (See OS/VS Utilities)
- IEBUPDTE (See OS/VS Utilities)
- IEHIOSUP, for VS1 only (See OS/VS Utilities)

- IMASPZAP (See OS/VS1 Service Aids or OS/VS2 System Programming Library: Service Aids)

System Levels

You must ensure that SMP uses the correct levels of programs. For example, an incompatible linkage editor may cause a module to be placed incorrectly into a library.

For VS1, IEHIOSUP is a system-dependent utility program that is critical to the processing of SYSMODs. SMP attempts to ensure that the correct level of IEHIOSUP is used by the following algorithm:

- If you specified a substitute name for IEHIOSUP in the IOSUPNAME subentry of the PTS SYSTEM entry, SMP executes that program, if it resides in the running system's LINKLIB, or is present in a library specified in the JOBLIB or STEPLIB DD statement.
- If the LINKLIB DD statement is present, SMP searches for IEHIOSUP on that library:
  - If the search fails, SMP processing is terminated.
  - If the search is successful, that version of IEHIOSUP is used instead of the version on the running system.
- If the LINKLIB DD statement is not present, SMP searches for IEHIOSUP on the data set(s) specified in the STEPLIB or JOBLIB DD statements, if either or both were specified, or the running system's LINKLIB.
  - If the search fails, SMP processing is terminated.
  - If the search is successful, the appropriate version of IEHIOSUP is used.

Writing Messages to SMPLOG

The LOG control statement enables you to write user-originated messages to the SMPLOG data set. You can write to the SMPLOG data set to provide a record of SMP operations or other records that you determine are appropriate. The messages are date and time stamped. See the 'LOG Control Statement' in Chapter 7 for further information.

Controlling SMP Processing

There are four methods to control SMP processing as follows:

- Establishing Return Code Threshold Values

  SMP checks the values of the return codes after the invocation of system programs to determine the success of those programs. If the return code value is higher than that considered successful, the SYSMOD or SYSMODs being processed, or perhaps the SMP function itself, is terminated. You may override the default return code values by changing the appropriate subentries in the PTS SYSTEM entry. The 'UCL SYS statement' in Chapter 7 describes the defaults for the various system programs invoked by SMP.

- Specifying the RC Keyword on Control Statements

  Many of the control statements have an RC keyword that allows you to override normal SMP return code processing for most functions. The control statements in Chapter 7 describe the syntax and use of the RC keyword. In general, the following describes the actions that take place when the RC keyword is specified:

  - If a specified function returns a code greater than the code specified in the RC keyword, SMP bypasses processing of the control statement.

  - If all specified functions have returned codes less than or equal to the codes specified in the RC keyword, SMP continues processing of the control statement.

  - For functions not specified in the RC keyword operand list, their return code values do not affect processing.

- Reseting the Return Codes

  The RESETRC control statement allows you to reset the previous highest return code from an SMP function to zero. This permits you to code other control statements in the CNTL input stream without specifying the RC keyword. This is useful when the control statements following the RESETRC control statement are not dependent upon the successful completion of the control statements that precede the RESETRC control statement. The RESETRC control statement does not affect the overall highest return code encountered during SMP processing; that is, the highest return code encountered will be the return code of SMP itself and is shown in message HMA205. A further description is found in 'The RESETRC control statement' in Chapter 7.

- User Exit Routines

  You can control the processing of SMP by providing user-written exit routines. These are described later in this chapter in 'User-Written Exit Routines.'

Directories in Storage

SMP performance can vary significantly, depending on the number of directory entries in the ACDS and CDS, and the number of new entries being added to those directories. Therefore, the functions that affect those directories have processing options as follows:

- Processing of the directory in read-only mode in storage - This is accomplished by specifying the DIS(READ) operand on the control statement. The appropriate directory is read into storage at the beginning of the function so that subsequent I/O operations to locate entries result in an in-storage search. This decreases the amount of time spent waiting for the entry image to be returned by the I/O operation from the direct access storage device. However, any changes to existing directory entries, deletions of existing entries, and additions of new entries is accomplished as encountered by I/O operations to the directory.

- Processing of the directory in read/write mode in storage - This is accomplished by specifying the DIS(WRITE) operand on the control statement. This mode provides the in-storage search provided by read-only mode. In addition, SMP does not update the directory entries on the data set until the SMP function completes. SMP changes the in-storage copy of the directory as it encounters each operation to delete, add, or replace an entry. This further decreases the wait time for I/O operations that update the directory in the data set.

- Processing the directory in the data set - This is accomplished by specifying the DIS(NO) operand on the control statement. In this mode, the directory is not read into storage for either read or write operations, thus resulting in wait time for all I/O operations. However, this option should be used in two cases:

- When the number of entries being processed is small, the time saved by not performing I/O operations to read and write directory entries individually is offset by the time necessary to read in the directory and possibly write it back out.

- When the amount of storage necessary to hold the directory is not available.

The SMP control statements that allow the DIS operand are ACCEPT, APPLY, JCLIN, RESTORE, and UCLIN. If the DIS operand is not specified, the default for these functions is READ, except for JCLIN, which has a default of WRITE. The WRITE option gives the best performance. However, since the directory is not updated on the data set until the function completes, data set integrity cannot be ensured if the function does not complete before a system failure. The topic 'Errors Related to Directory-in-Storage Processing' in Chapter 5 describes this problem in detail.

Special SYSMOD Processing Considerations

Some SYSMODs require special processing. When this is the case, the documentation supplied with the SYSMOD tells you what to do. As a general rule, you should read the documentation that accompanies SYSMODs even if no special processing is required.

Applying SYSMODs to Stage I SYSGEN Macros

Some manual intervention and special packaging is required when you apply SYSMODs to Stage I SYSGEN macros. The following are hints for processing these SYSMODs:

- Process the SYSMOD containing the modification to Stage I SYSGEN macros. You can invoke the ACCEPT function only at this point in time; that is, specify 'ACCEPT SELECT(sysmodid) NOAPPLY.', where "sysmodid" is the name of the SYSMOD modifying the Stage I SYSGEN macros.

- Execute a Stage I SYSGEN job to create a new Stage I output tape.

- For VS1, execute the mutually exclusive module PROCs, where applicable, to resolve possible conflicts. See the topic 'Applying SYSMODs After Partial SYSGEN' later in this chapter.

- Execute the SMP JCLIN function using the newly created Stage I output tape.

- Process all SYSMODs that specified the SYSMOD that modified the Stage I SYSGEN macros as a prerequisite. If you have not applied the latter SYSMOD, it must be processed concurrently with the others.

This special processing ensures that the modules that are assembled during SYSGEN and the changes to load module structure are reflected in the CDS before applying the set of SYSMODs that depend on the Stage I SYSGEN macro modifications.

Applying SYSMODs After Partial SYSGEN (VS/1 only)

You may encounter problems in VS1 when applying SYSMODs after you have performed a partial system generation (an I/O device generation or a nucleus generation.) Depending on the options specified during the original system generation and those specified during the partial generation, you may have caused mutually exclusive pairs of distribution library modules to be created in the target system. To resolve these conflicts, you may have to do the following before executing the SMP JCLIN function:

- Delete both module entries for each mutually exclusive pair from the CDS.

- If an I/O device generation was performed, use the SMPIO procedure to delete the conflicting entries from the CDS.

- If a nucleus generation was performed, use the SMPNUC procedure to delete the conflicting entries from the CDS.

SMPIO and SMPNUC are SMP procedures that reside in SYS1.PROCLIB. You can use the operator START command or JCL to invoke them, as described below:

Using the START Command

You can use the following START command to invoke SMPIO or SMPNUC:

```
S [SMPIO | SMPNUC],id,aaa,vvv,,dsn=ddd
```

where

id

specifies the partition number.

aaa

specifies the device address or device type of the CDS. The default is SYSDA.

vvv

specifies the volume serial number of the CDS device. This parameter is required.

ddd

specifies the data set name of the CDS. The default is SMPCDS.

Using JCL

You can use the following JCL to invoke SMPIO or SMPNUC:

```
//     EXEC [SMPIO | SMPNUC]
//IEFPROC.IEFRDER DD VOL=SER=vvv,UNIT=aaa,DSN=ddd
```

where

vvv

specifies the volume serial number of the CDS device. This parameter is required.

aaa

specifies the device address or device type of the CDS. The default is SYSDA.

ddd

specifies the data set name of the CDS. The default is SMPCDS.

Note: Although the SMPLOG DD statement specifies SYSOUT=A, you may override it to specify the LOG data set.

User-Written Exit Routines

You can write a user exit routine that is invoked by SMP during processing. The name of the user exit routine is HMASMUXD. It is a separate load module that is not link edited with HMASMP. A dummy version of HMASMUXD is supplied with the modules of SMP and is copied to the target system library LINKLIB. You replace this module with your user exit routine of the same name. During SMP initialization, HMASMUXD will be loaded via the LOAD macro and invoked via the CALL macro at the appropriate places during SMP processing.

If you chose to place your exit routine in a library other than LINKLIB, you must ensure that it is an authorized library.

The function of HMASMUXD is to define the user exits to be invoked by SMP during processing. Currently, only one user exit can be defined in HMASMUXD, and it is described in 'User Exit 1' in this chapter.

Module HMASMUXD (User Exit Determinator)

Since you must replace module HMASMUXD in LINKLIB, the following information is provided to help you write HMASMUXD.

The module must be coded using standard linkage conventions. The register values at invocation must be the same when the module returns to SMP with the exception of registers 0, 1, and 15. The registers should be saved in an area with backward and forward save area chains.

Module HMASMUXD is passed the address of a parameter list in general register 1. A mapping macro HMASMUXP, is provided for this parameter list in SYS1.MACLIB. The parameter list is mapped as follows:

Field Name	Location	Description
UXPUXNUM	1 - 2	User exit number (binary number)
	3 - 4	Unused
UXPUXNAM	5 - 12	User exit name
UXPUXAD	13 - 16	Address of user exit
UXPFUNCT	17 - 24	SMP function name
UXPPRMAD	25 - 28	Address of user exit parameter list
UXPLOJAD	29 - 32	Address of work area common to user exits
UXPLOEAD	33 - 36	Address of work area for this exit
UXPCTBAD	37 - 40	Reserved
UXPMODAD	41 - 44	Reserved

SMP passes the user exit number in field UXPUXNUM and module HMASMUXD determines if that user exit is to be activated. You can use one of two methods to activate the user exit. The name of the exit module can be placed in field UXPUXNAM or the address of the exit routine can be placed in field UXPUXAD. If the name is passed back, field UXPUXAD must be binary zeroes. If the address is passed back, field UXPUXNAM must be blank. If field UXPUXNAM is not blank and the field UXPUXAD is not binary zeroes, SMP issues an error message and terminates. If field UXPUXNAM is blank and field UXPUXAD is binary zeroes, SMP assumes that the user exit does not exist.

When the name of the user exit is passed back to SMP, the user exit module must exist as a load module in LINKLIB or in a data set defined by the STEPLIB or JOBLIB DD statements, if present. SMP issues a LOAD macro for the user exit module, a CALL macro to invoke the user exit, and a DELETE macro to remove the user exit when no longer required. If the user exit applies to the total SMP function, it will be loaded during initialization and deleted during termination. If the user exit applies to a specific SMP function, it will be loaded during initialization of the function and deleted at termination of the function.

When the address of the user exit routine is passed back to SMP, the user exit is invoked by a CALL macro. With this method, it is the user's responsibility to issue the LOAD macro for the user exit module unless it is part of module HMASMUXD.

When module HMASMUXD returns to SMP, general register 15 must contain one of the following values:

- 0 - Exit information supplied or ignored
- 12 - Terminate function
- 16 - Terminate SMP

If any other value is returned, SMP issues an error message and terminates.

The HMASMUXD module supplied with SMP does not modify the parameter list. It returns a value of 0.

User Exits

When a user exit is invoked, general register 1 contains the address of the parameter list HMASMUXP as shown above. The UXPLOJAD field is used to pass information between user exits; however, this field is not currently used because only one exit is supported. The UXPLOEAD field is used within an exit to pass information when the next call is made to the exit. This field is not referenced by SMP.

When the user exit returns to SMP, general register 15 must contain a valid return code, which is defined for each exit. If a value is returned that is not valid, SMP issues an error message and terminates.

User Exit 1

This exit routine allows you to scan the SYSMOD data in the SMPPTFIN data set. This exit performs the same function as the HMASMEXT module supported in previous versions of SMP. This user exit is called User Exit 1.

See 'RECEIVE Processing' in Chapter 3 for information on when this exit is called.

When the exit is called, the fields in the parameter list have the following values:

- UXPUXNUM - "1" - user exit number
- UXPFUNCT - "RECEIVE" - the RECEIVE function
- UXPPRMAD - address of 81 byte buffer area containing input record from PTFIN. The first character will be set to X'04' at End-Of-File, otherwise it will be X'00'. Charaters 2-81 will contain the PTFIN input record that is next to be processed by RECEIVE.

When the exit returns to SMP, one of the following values must be returned in general register 15:

- 0 - Continue normal processing
- 4 - Invalid
- 8 - Stop processing the SYSMOD. RECEIVE processing will not receive this SYSMOD, but records from the SYSMOD continue to be passed to the user exit.
- 12 - Stop RECEIVE processing
- 16 - Stop SMP processing
- 20 - Insert a record after the current one in the buffer. The exit is reinvoked without reading from PTFIN after the contents of the buffer area are processed. The exit routine returns data that is to be part of the SYSMOD being read in the buffer area. When no more data is to be placed in the buffer, the exit clears the buffer area and returns a 0 in register 15.
- 24 - Delete record in buffer area

User Exit 2

This exit routine allows the user to control the X37 RETRY function of SMP Release 4 . The exit is called after SMP has determined that a RETRY can be attempted. (Generally a RETRY is considered to be a compress of the target datset followered by a re-invocation of the failing UTILITY. Note that the dump for the failure has been cancelled When the user exit is called.)

When the exit is called, the fields in its imput parameter list have the following values:

- UXPUXNUM - '2' - user exit number
- UXPFUNCT - 'APPLY' 'ACCECPT' or 'RESTORE' - SMP function being performed.
- UXPPRMAD - address of 25 byte parameter list which is mapped as follows:

FIELD NAME	LOCATION	DISCRIPTION
UX002DDN	1-8	Target DDNAME on which the B, D, or E37-04 occurred.
UX002PGM	9-16	The program name of the UTILITY invoked which caused the failure.
UX002ACH	17-19	The abend code encountered in Hexadecimal. (same format as
UX002RCH	20-20	The abend code reason in Hexadecimal
UX002ACP	21-23	The abend code in printable EBCDIC
UX002RCP	24-25	The abend reason code in printable EBCDIC

When the exit returns to SMP, one of the following values must be returned in general register 15:

- 0 - Continue normal processing
- 12 - Stop the SMP function (APPLY, ACCEPT, or RESTORE) processing; perform no RETRY.
- 16 - Stop SMP processing; perform no RETRY
- 20 - Perform modified RETRY processing. Re-Invoke the failing UTILITY but do not compress the failing dataset.

Any other return code is invalid and is converted to a return code of 12.

User Modifications

You can use SMP to perform user modifications that:

- Modify existing system elements, such as load modules, object modules, source modules, and macros
- Add new load modules to your target system

- Add modules to existing load modules in your target system

Modifying Existing Elements

You can modify existing system elements by performing the following steps:

- Use the SMP modification control statements to define the modifications you wish to make. These are described in Chapter 8. You should use ++USERMOD as the header modification control statement.

- Place the modification control statements in the SMPPTFIN data set. You may use any of the packaging techniques described in Chapter 2.

- Use the SMP control statements to incorporate the modifications in your target system and to update the CDS.

User Modification Examples

The following four examples illustrate the use of IMASPZAP to perform modifications to distribution library modules, load modules, modules, and CSECTs within the modules. The examples assume that the load module structures are:

```
Load Module Name                 Load Module Name
  Module Name                      Module Name
    CSECT Name                       CSECT Name

LMODA                            LMODB
  MOD1                             MOD1
    CSECT1                           CSECT1
    CSECT2                           CSECT2
    CSECT3                           CSECT3
  MOD2                             MOD2
    MOD2                             MOD2
  MOD3
    MOD3
  MOD4
    MOD4
```

The examples assume the use of the cataloged procedure described in 'SMP Cataloged Procedure' earlier in this chapter. The appropriate DD statements for defining the target system and distribution libraries have been added to the procedure.

Example 1

Control section CSECT2 in module MOD1, which is in both LMODA and LMODB, is to be modified in both load modules.

```
//SMPCNTL   DD  *
  RECEIVE.
  APPLY S(MYMOD01).
/*
//SMPPTFIN  DD  *
++USERMOD(MYMOD01).
++VER(Z038) FMID(FXY1000).
++ZAP(MOD1).
  NAME CSECT2
  VER 000D FF
  REP 000D FE
/*
```

Example 2

Control section MOD3 in module MOD3, which is in LMODA, is to be modified.

```
//SMPCNTL   DD  *
  RECEIVE.
  APPLY S(MYMOD02).
/*
//SMPPTFIN  DD  *
++USERMOD(MYMOD02).
++VER(Z038) FMID(FXY1000).
++ZAP(MOD3).
  NAME MOD3
  VER 000A 00
  REP 000A FF
/*
```

Example 3

Control section CSECT2 in module MOD1, which is in LMODA and LMODB, is to be modified in LMODB only.

```
//SMPCNTL   DD  *
  RECEIVE.
  APPLY S(MYMOD03).
/*
//SMPPTFIN  DD  *
++USERMOD(MYMOD03).
++VER(Z038) FMID(FXY1000).
++ZAP(MOD1).
  NAME LMODB CSECT2
  VER 0000 00
  REP 0000 FF
/*
```

Example 4

Control section CSECT3 in module MOD1, which is in LMODA and LMODB, is to be modified with an EXPAND-type request.

```
//SMPCNTL   DD  *
  RECEIVE.
  APPLY S(MYMOD04).
/*
//SMPPTFIN  DD  *
++USERMOD(MYMOD04).
++VER(Z038) FMID(FXY1000).
++ZAP(MOD1).
  EXPAND CSECT3(4)
  NAME CSECT3
  VER 000D FF
  REP 000D FE
/*
```

Example 5

The following example shows how to add new load modules to your target system. Alternative methods involve executing the SMP JCLIN function prior to applying the modifications; however, this example requires only a single invocation of SMP and is the recommended method.

This example assumes the use of the cataloged procedure described earlier in this chapter. The procedure is named SMPPROC and resides in SYS1.PROCLIB. The appropriate DD statements for SYS1.USERLIB and SYS1.SVCLIB are not included in this procedure; you must supply them. Optionally, you may update the cataloged procedure to include DD statements for these libraries prior to invoking SMP.

The set of elements to be added to the target system include:

- Load modules USERSVC1 and USERSVC2 in SYS1.SVCLIB
- Load module USERTWO in SYS1.LINKLIB
- Modules USERSVC1, USERSVC2, IEFUSERA, and IEFUSERB
- Macros USERMACA and USERMACB in SYS1.MACLIB
- Assembler input text for module IEFUSERA

The JCL input data that describes the assembler step for IEFUSERA, the link edit step for load module USERTWO, and the copy step for USERSVC1 and USERSVC2 are placed in the user modification itself following the ++JCLIN modification control statement.

```
//ADDMYMOD JOB   1,'MYNAME',MSGLEVEL=1,CLASS=A
//STEPA     EXEC SMPPROC
//SVCLIB    DD   DSN=SYS1.SVCLIB,DISP=OLD
//USERLIB   DD   DSN=SYS1.USERLIB,DISP=OLD
//SMPCNTL    DD  *
  RECEIVE.
  APPLY S(MYMOD05) RC(RECEIVE=04).
  ACCEPT S(MYMOD05) USERMODS RC(APPLY=04).
/*
//SMPPTFIN  DD   DATA,DLM='$$'
++USERMOD(MYMOD05).
++VER(Z038) FMID(FXY1000).
++JCLIN.
//MYJOB     JOB  1,'MYNAME',MSGLEVEL=1,CLASS=A
//STEP1     EXEC PGM=ASMBLR
//SYSPUNCH DD    DSN=USER.OBJPDS(IEFUSERA),DISP=OLD
//SYSIN     DD   *
  PRINT ON,NODATA
  USERMACA PARM1,PARM2
  COPY USERMACB
  END
/*
//STEP2     EXEC PGM=IEWL,PARM='RENT'
//SYSLMOD   DD   DSN=SYS1.LINKLIB,DISP=OLD
//USERLIB   DD   DSN=SYS1.USERLIB,DISP=OLD
//SYSPUNCH DD    DSN=USER.OBJPDS,DISP=OLD
//SYSIN     DD   *
  INCLUDE SYSPUNCH(IEFUSERA)
  INCLUDE USERLIB(IEFUSERB)
  ENTRY USERONE
  NAME USERTWO(R)
/*
//STEP3     EXEC PGM=IEBCOPY
//USERLIB   DD   DSN=SYS1.USERLIB,DISP=OLD
//SVCLIB    DD   DSN=SYS1.SVCLIB,DISP=OLD
//SYSIN     DD   *
  COPY INDD=USERLIB,OUTDD=SVCLIB
  SELECT MEMBER=(USERSVC1,USERSVC2)
/*
++MAC(USERMACA) TXLIB(MACTXLIB) SYSLIB(MACLIB) DISTLIB(AMACLIB).
++MAC(USERMACB) TXLIB(MACTXLIB) SYSLIB(MACLIB) DISTLIB(AMACLIB).
++MOD(IEFUSERB) TXLIB(MODTXLIB) DISTLIB(USERLIB) LEPARM(RENT).
++MOD(USERSVC1) TXLIB(MODTXLIB) DISTLIB(USERLIB) LEPARM(RENT).
++MOD(USERSVC2) TXLIB(MODTXLIB) DISTLIB(USERLIB) LEPARM(RENT).
$$
```

During APPLY processing, the following updating will occur:

- By processing the JCL input data following the ++JCLIN modification control statement, SMP creates the following CDS entries:
  - An ASSEM entry for IEFUSERA
  - MAC entries for USERMACA and USERMACB with a GENASM subentry for IEFUSERA
  - An LMOD entry for USERTWO
  - MOD entries for IEFUSERA and IEFUSERB with LMOD subentries for USERTWO
  - MOD and LMOD entries for USERSVC1 and USERSVC2
- SMP places macros USERMACA and USERMACB in SYS1.MACLIB.
- By processing the ++MAC modification control statements for USERMACA and USERMACB, SMP assembles IEFUSERA.
- SMP link edits modules IEFUSERA and IEFUSERB to form load module USERTWO and places the load module in SYS1.LINKLIB.
- SMP link edits modules USERSVC1 and USERSVC2 individually and places the resultant load modules in SYS1.SVCLIB.
- SMP assigns MYMOD05, the SYSMOD-ID of the user modification, as the value of the RMID subentries for the affected MAC and MOD entries.

During ACCEPT processing, SMP performs the following updates:

- Macros USERMACA and USERMACB are placed in the distribution library SYS1.AMACLIB.
- Modules IEFUSERB, USERSVC1, and USERSVC2 are link edited and placed in the distribution library SYS1.USERLIB.
- MAC and MOD entries for the macros and modules defined in the element modification control statements are created with an RMID subentry value of MYMOD05.

Chapter 5: SMP Diagnostic Techniques

This chapter provides general diagnostic techniques that you can use to correct errors that might occur during SMP processing. The information should enable you to:

- Detect error conditions
- Resolve shortages of storage
- Resolve shortages of direct access storage
- Correct errors related to directory-in-storage processing

In addition, SMP STAE processing is described.

Detecting Error Conditions

If you encounter unexpected or incomplete results from the execution of SMP control statements, use the following procedures to determine the cause of the problem and the correct recovery techniques to use:

- Examine the return codes contained in the SMPOUT data set. Starting with the final code (that is, the one returned by the failing job step), trace backwards through the data set in search of the SMP function return codes that caused the job step return code. Remember that a single return code can be the product of multiple errors.

  See 'Diagnostic Messages' in Chapter 10 for a description of how return codes are reflected in the severity code of an SMP message.

  The job step return code issued for SMP is equal to the highest return code generated by all SMP functions within that step. The job step return codes are:

  00 SMP processing completed successfully and without errors.

  04 SMP processing completed but warning messages are

issued.

08 SMP processing completed, but processing errors occurred and processing terminated for at least one system modification. Check for SYSMODs that were processed but have the ERROR indicator set in the CDS or ACDS.

12 SMP processing terminated for at least one SMP function.

16 SMP processing terminated because of a severe error.

For specific return codes for each of the SMP functions, see the return codes for each respective control statement in Chapter 7.

- Check for any return code contingencies that you may have coded using the RC operand on the SMP control statements. The RC operand allows you to specify the maximum acceptable return codes from specified SMP control statements in order to bypass normal SMP return code processing. If a specified control statement returns a code exceeding the maximum specified in the RC operand, the control statement that contains the RC operand is not executed and issues a return code of 12.

  For example, if you specify RC(RECEIVE=04) on the APPLY control statement, and the RECEIVE control statement returns a code of 08, APPlY processing is not performed, and a return code of 12 results.

  For further information about the RC operand, refer to the discussion of the RC operand for each respective control statement in Chapter 7.

- As you trace back through the return codes, examine SMPOUT for error and warning messages issued with the return codes. Use the information supplied by the messages to help you interpret the meaning of the return codes.

- Check the SYSPRINT data set for information about the success or failure of the system programs invoked by SMP functions.

- Issue 'LIST LOG' to display the contents of the LOG data set. This log is cumulative and should be examined for the impact of prior SMP runs on the current problem.

For more details on the LIST control statement, see Chapter 7.

The status of a SYSMOD is indicated in the SMP reports and messages that are normally produced. However, if this output is not available, use the following techniques to obtain the data:

- Issue 'LIST CDS SYSMOD' to obtain the status of any SYSMODs applied or restored but suspected of being in error. Check to see if the ERROR indicator is set for those SYSMODs during APPLY and RESTORE.

- Issue 'LIST ACDS SYSMOD' to obtain the status of any SYSMODs accepted or restored but suspected of being in error. Check to see if the ERROR indicator is set for those SYSMODs.

- Issue 'LIST PTS SYSMOD' to obtain the status of SYSMODs on the PTS that are received or rejected.

  Specific error recovery for each of the SMP functions is in the Error Recovery section for each respective control statement in Chapter 7.

Resolving Shortages of Storage

If an SMP message indicates that there is insufficient storage for processing, do the following:

- Allocate a larger region or partition size and execute the job step again.

- Remove one of the control statement operands that causes storage to be used, such as the XREF keyword on the LIST control statement, and execute the command again.

Resolving Shortages of Direct Access Storage

This section outlines methods to:

- Prevent shortages of direct access storage

- Recover from shortages of direct access storage

Preventing Shortages of Direct Access Storage

You can prevent shortages of direct access storage using the following methods:

- After system generation, list the VTOCs of the target system library and distribution library (DLIB) volumes. If you notice any target library or DLIB data sets that have little free space, you can reallocate them and then copy these data sets into larger data sets to prevent future space problems.

  See Chapter 4 for more information about initial allocation of system and SMP data sets.

- Use the COMPRESS keyword on the ACCEPT, APPLY, REJECT or RESTORE control statement to recover space in target libraries and DLIBs during SMP processing. In order to have the COMPRESS function executed during ACCEPT, APPLY, REJECT, or RESTORE processing, that function must process a system modification.

  See the ACCEPT, APPLY, REJECT and RESTORE control statements in Chapter 7 for detailed information on the COMPRESS keyword.

- Periodically list the VTOC to check the amount of space that is left in the data sets that you have already compressed. If you discover that a compressed data set is running out of space, you can allocate a new, larger data set and copy the old data set into the new one.

- Every time that you issue the APPLY control statement for modifications that cause a re-link edit of IEANUC01 and (1) you specify a new value for 'n' in the NUCID(n) operand, or (2) the NUCID has been preset in the CDS SYSTEM entry, you cause an additional copy of the nucleus load module (IEANUC0n) to be saved.

  For example, if you issue APPLY and specify NUCID(3), a copy of the current load module is saved as IEANUC03; if you issue APPLY and specify NUCID(7), a copy of the current load module is saved as IEANUC07.

  If you use the NUCID keyword on the APPLY control statement, you must ensure that the NUCLEUS data set is large enough to hold the number of copies of IEANUC0n that you create, where 'n' is the number of copies.

- If you use JCLIN to define to SMP your own modules assembled with your own macros, SMP scans the assembler and linkage editor JCL to create macro, assembly modules, and load module entries in the CDS for your modules. If the SYSIN data to be assembled is large, consider one of the following to conserve space in the CDS:

    1) Include the assembler COPY statement as part of the assembly SYSIN to obtain large amounts of data from SYSLIB at assembly time. This reduces the size of the assembly data stored in the CDS.

       See the examples of adding new load modules and module entries to the CDS in Chapter 4.

    2) To eliminate the creation of ASSEM entries in the CDS, process your macro modifications using the ASSEM and DISTSRC keywords on the ++MAC, ++MACUPD or ++UPDTE modification control statement. SMP performs the macro modification and assembles the modules defined in the ASSEM keyword using assembly data from the library specified by the DISTSRC keyword.

       See the ++MAC, ++MACUPD, and ++UPDTE modification control statements in Chapter 8 for more details about the ASSEM and DISTSRC keywords.

Recovering from Shortages of Direct Access Storage

You can recover from many shortages of direct access storage using the following methods:

- If an SMP function fails because of insufficient space, check to see if the COMPRESS keyword is allowed and was specified for that function. If the COMPRESS keyword is valid, and if it was not specified the last time, rerun the SMP function with COMPRESS.

  Note that, although you can compress the CDS and ACDS, you cannot compress them using SMP because SMP might be maintaining in-storage copies of their directories. You must use a standard system program to perform the compression.

- To obtain additional space in the LOG data set, use one of the following techniques:

  1) Allocate a new LOG data set and create a backup copy of the old LOG data set, retaining it according to your normal recovery procedures.

  2) Create a backup copy of the old LOG and retain it according to your normal recovery procedures. The next time you run SMP functions, indicate DISP=OLD for SMPLOG; this will overlay the contents of the old LOG that you saved.

     Note that you must return to DISP=MOD the next time you execute SMP or you will continue to overlay the LOG every time that SMP functions are executed.

- To obtain additional space on the CDS or ACDS, the target libraries, or the distribution libraries, allocate a new, larger data set and copy the old, out-of-space data set into the new one.

- To obtain additional space in the MTS, PTS, or STS data sets, use one of the following techniques:

  1) Issue ACCEPT, REJECT, or RESTORE for non-accepted SYSMODs, if any. Specify the COMPRESS operand with a value of 'SMPMTS', 'SMPPTS', or 'SMPSTS' with the next ACCEPT, REJECT, or RESTORE function you execute.

  2) If no SYSMODs are candidates for an ACCEPT, REJECT or RESTORE, allocate a new, larger data set and copy the old, out-of-space data set into the new one.

- If an out-of-space condition (system ABEND B37 or D37) occurs on the LOG data set during the execution of LIST LOG, any subsequent attempt to issue LIST LOG or any other SMP control statement will result in the same abnormal termination.

  Since the LOG function records every SMP control statement, it will attempt to write that control statement to an already full data set. Use a utility program such as IEBGENER to copy SMPLOG to a larger data set.

Errors Related to Directory-in-Storage Processing

When the DIS(WRITE) operand is specified on a control statement, the CDS and the ACDS can be updated in an in-storage mode. When this is done, an indicator is set in the appropriate SYSTEM entry, indicating that the data set might now be at a level below the actual status of the target system or DLIBS. This condition occurs if SMP abnormally terminates prior to rewriting the data set directory.

When this condition occurs, SMP issues a warning message at the next invocation. You should reissue the command that was executed during the last invocation. While this might result in some modifications being reprocessed, it will also ensure that the data set is updated to the correct status.

The second error condition that might occur as a result of the DIS(WRITE) operand is an error during the rewrite of the directory. Prior to starting the directory rewrite, an indicator is set on in the SYSTEM entry indicating that the data set directory is no longer usable. If the rewrite fails because of an I/O error or an abnormal termination that SMP STAE cannot recover from, the indicator is left on.

At the next invocation of SMP, this indicator is checked, and, if it is on, the data set is considered unusable and SMP processing terminates. The only recovery from this type of error is to restore the data set using a previously saved copy.

SMP STAE Processing

The SMP STAE routine gets control whenever an ABEND occurs to perform the following processing:

- It issues message HMA432 to inform you that STAE processing is in effect.
- The CDS or ACDS directory entries are written if the DIS option is used to perform any updates that may have occurred as a result of SYSMOD processing.
- Completion processing for processed or partially processed SYSMODs is done and completion messages are issued.

- SYSMODs that were in process when the ABEND occurred, but that were not completed, are marked with the ERROR status.

- The reports that are normally produced by the function that was in process are produced.

- Control is passed to the supervisor for termination processing; no attempt is made to retry processing.

You can examine the reports and the dump, if any, to correct the problem and resubmit the job.

Reports that notify you, in summary format, of the outcome of SMP processing are produced for the RECEIVE, APPLY, RESTORE, and ACCEPT functions. These reports will appear in the SMPRPT output data set when the SMPRPT DD card is present in the JCL statements used to execute SMP. Otherwise, the reports will appear in the SMPOUT data set.

RECEIVE Output Data

Message Output from RECEIVE

As a result of RECEIVE processing, SMP writes messages interspersed with copies of the modification control statements of the SYSMODs processed to the SMPOUT data set. By analyzing these messages you can:

- Determine syntax and construction errors within a SYSMOD
- Determine the point at which an I/O error occurred
- Find information pertaining to SYSMODs for non-received conditions
- Find detailed information on the loading of Relfile data sets to the SMPTLIB volume(s)
- Determine which SYSMODs were not present in the SMPPTFIN data set that were specified in the SELECT operand list.

The RECEIVE SUMMARY Report

When SYSMODs are processed by RECEIVE, SMP produces a RECEIVE SUMMARY REPPORT on the SMPRPT dataset.

The "RECEIVE SUMMARY REPORT" lists those SYSMODs processed from the SMPPTFIN dataset. The SYSMODs included in the report depend upon the user specification of SELECT, EXCLUDE, or MASS. In SELECT mode, the report contains information for only those SYSMODs which were explicitly selected. In EXCLUDE mode, the report contains information for all SYSMODs in SMPPTFIN except those explicitly excluded and those which were previously RECEIVED. In MASS mode, the report contains information for all SYSMODs in SMPPTFIN except those which were previously RECEIVED.

Four fields are present on each line of the report for a SYSMOD:

- Field 1 - The SYSMOD ID (7 character identifier)
- Field 2 - STATUS (RECEIVED or NOT RECEIVED)
- Field 3 - SYSMOD Type (FUNCTION, PTF, APAR, or USERMOD)
- Field 4 - Additional information (see below)

Additional information (Field 4 of the report line) may appear as follows:

1) ALREADY RECEIVED - The SYSMOD was not received because the SYSMOD was found on the SMPPTS dataset as RECEIVED. This information appears only if the SYSMOD was explicitly selected.

   User considerations: You must delete the SYSMOD from the PTS using the REJECT control statement before receiving the SYSMOD.

2) I/O ERROR - The SYSMOD was not received because of an I/O error on an SMP dataset.

   User considerations: Investigate and correct the cause of the I/O error. If the I/O error occurred while reading a Relfile data set or writing to an SMPTLIB data set, the ERROR indicator is set in the PTS SYSMOD entry.

3) NO APPLICABLE ++VER - The SYSMOD was not received because no ++VER modification control statement(s) was found which applied to the system release (SREL) and/or FMID the SMPPTS System entry.

   This information appears only if the SYSMOD was explicitly selected.

User considerations: Ensure that the SYSMOD is required in your environment.

Ensure that the correct PTS data set is used if multiple environments are maintained by different PTS data sets. A list of all environments controlled by a PTS can be obtained using the LIST PTS SYS control statement.

If the PTS SYSTEM entry does not contain the SREL subentry required by the SYSMOD, it can be added using the UCLIN PTS function, with the UCL ADD SYS SREL statement.

If the PTS SYSTEM entry does not contain the FMID subentry required by the SYSMOD, it can be added using the the UCLIN PTS function with the UCL ADD SYS FMID statement, or by receiving the function SYSMOD specified in the FMID operand.

You can use the BYPASS operand on the RECEIVE control statement to bypass the FMID verification checks.

4) RELFILE PROCESS ERROR - The SYSMOD was not received because of an error attempting to allocate a dataset on the SMPLIB volume or during the IEBCOPY invocation to load the unloaded relfile to an SMPTLIB dataset.

User considerations: Ensure that the volume(s) referenced by the SMPTLIB DD statement contain enough direct access space to fulfill the requirement.

Adjust the SMP space requested by changing the DSSPACE parameter in the PTS SYSTEM entry using the UCLIN PTS function.

Check the results of the IEBCOPY invocation using the SYSPRINT data set or the IEBCOPY substitute for SYSPRINT output. An error condition is reported if the return code passed to SMP by IEBCOPY is not 0.

5) RELFILE NOT PROCESSED - The SYSMOD was not received because of a previous error which terminated RECEIVE processing before the RELFILEs for this SYSMOD could be loaded.

6) NOT FOUND ON SMPPTFIN - The SYSMOD was not received because it was not found on the SMPPTFIN dataset. This information appears only if the SYSMOD was explicitly selected.

7) SYNTAX/CONSTRUCTION - The SYSMOD was not received due to a syntax or construction error.

8) SMPTLIB DATASETS LOADED - The SYSMOD received had elements supplied in IEBCOPY unloaded files which were loaded to the SMPTLIB volume.

9) USER EXIT - The SYSMOD was not received due to the return code passed to SMP from the user exit routine.

   Check the results of the IEBCOPY invocation using the SYSPRINT data set or the IEBCOPY substitute for SYSPRINT output. An error condition is reported if the return code passed to SMP by IEBCOPY is not 0.

10) SYNTAX/CONSTRUCTION - An unknown operand was encountered on a modification control statement or a SYSMOD construction error was found.

   User considerations: This SYSMOD cannot be received. The SYSMOD modification control statements should be corrected by those responsible for construction.

   If the SYSMOD specifies the FILES and RELFILE operands incorrectly, subsequent SYSMODs in the SMPPTFIN data set are not received. The reason is described as "RELFILE CONSTRUCTION".

```
        DATE 78.069  TIME 10:26:42
             RECEIVE SUMMARY REPORT
SYSMOD-ID       TYPE       RELF  STATUS       NOT RECEIVED REASON
UZ89900         FUNCTION   YES   NOT RECVD    ALREADY RECEIVED
AZ00154         APAR       NO    NOT RECVD    FMID CHECK FAILURE
MQ00014         USERMOD    NO    RECEIVED
MQ00015         USERMOD    NO    NOT RCVD     SYNTAX ERROR
UZ03246         PTF        NO    NOT RECVD    USER EXIT
```

Figure 12. RECEIVE SUMMARY REPORT

APPLY, RESTORE and ACCEPT Output Data

Message Output from APPLY, RESTORE, and ACCEPT

When SYSMODs are processed by APPLY, RESTORE, and ACCEPT, messages appear in the SMPOUT data set. By analyzing these messages you can:

- Determine when SYSMODs are successfully processed
- Determine conditions that caused SYSMOD processing failure
- Determine the point at which an I/O error occurred.

Report Output from APPLY, RESTORE, and ACCEPT

SMP generates four reports for SYSMODs processed by APPLY and ACCEPT. Two reports can be generated for RESTORE processing. However, these reports are not produced when a function SYSMOD is selected for processing but is terminated prior to updating any target system or distribution libraries. By analyzing these reports you can:

- Determine those SYSMODS successfully processed and the libraries that were updated
- Determine those SYSMODS not processed because of error conditions encountered in related SYSMODs
- Determine which modifications to elements are regressed by SYSMODs processed by APPLY or ACCEPT
- Determine which SYSMODs were deleted from the CDS or ACDS as a result of applying or accepting a function SYSMOD with a DELETE operand in its ++VER modification control statement.

When the CHECK operand is specified on the APPLY or ACCEPT control statement, the reports indicate what will happen during actual processing of the SYSMODs. This "dry run" capability can save you valuable time by detecting error conditions that will occur if actual updates are done. For RESTORE processing, the CHECK mode can be useful in providing information about the SYSMODs that must be selected for RESTORE processing along with those specified in the SELECT operand list.

The SYSMOD STATUS Report

This report summarizes the processing that occurred for every selected SYSMOD. The SYSMODs are listed in alphanumeric sequence (see Figure 13).

The fields in the report are as follows:

- SYSMOD - The identifier of the system modification
- STATUS - Describes what has happened to the SYSMOD. The possible values of this field are as follows:

  1) APPLIED, ACCEPTED, or RESTORED - The SYSMOD was successfully processed.

  2) NOGO - The SYSMOD was not processed prior to any updates. The reason for the NOGO condition can be that a related SYSMOD has an error. The message output should be checked to determine the cause of the error.

  3) ERROR - The SYSMOD was terminated while SMP was updating the libraries. The reason for the ERROR condition can be that a related SYSMOD has an error. The message output should be checked to determine the cause of the error. This condition does not appear when the CHECK operand is specified.

  4) DELETED - The SYSMOD was explicitly or implicitly deleted.

  5) INCMPLT - A function SYSMOD was terminated causing termination of the SMP function. SYSMODs with this status may or may not be processable. Subsequent processing after correcting the cause of the function SYSMOD being terminated will determine the processing status of these SYSMODs.

     When this status is present, no ELEMENT SUMMARY report is produced.

- TYPE - The system modification type (APAR, FUNCTION, PTF, or USERMOD).

- FMID - For function SYSMODs, the SYSMOD-ID of the function; for service SYSMODs, the SYSMOD-ID of the owning function.

- REQUISITE SYSMODS - Lists every SYSMOD that is a requisite of the SYSMOD. The lists are preceded by the type of requisite as follows:

  1) IFREQ - The SYSMODs are conditional requisites of the SYSMOD, defined by its associated ++IF modification control statements or, if the SYSMOD is a function, defined by previously processed SYSMODs.

  2) PRE - The SYSMODs are prerequisites of the SYSMOD, defined by the PRE operand in its ++VER modification control statement.

  3) REQ - The SYSMODs are requisites of the SYSMOD, defined by the REQ operand in its ++VER modification control statement.

If a dash (-) appears next to a listed SYSMOD, that SYSMOD has NOGO status. This may mean that the SYSMOD is not available for processing.

If an asterisk (*) appears next to a listed SYSMOD, that SYSMOD has NOGO status, but the appropriate option was specified in the BYPASS operand list on the APPLY or ACCEPT control statement. This means that if the SYSMOD is not available for processing, the SYSMOD that has specified it as a requisite is considered processable.

```
    DATE 78.001  TIME 09:25:47/HMASMP LVL 04.00 SMPRPT OUTPUT
SYSMOD STATUS REPORT FOR APPLY CHECK PROCESSING

  NOTE:  '-' INDICATES THE REQUISITE SYSMOD CONDITION IS NOT SATISFIED
         '*' INDICATES THE NON SATISFIED REQUISITE SYSMOD CONDITION IS BYPASSED

SYSMOD   TYPE      STATUS  FMID     REQUISITE SYSMODS

AZ00124  APAR      APPLIED GXY1000  PRE     UZ00010

GXY1000  FUNCTION  APPLIED GXY1000

HXY1010  FUNCTION  APPLIED GXY1000  PRE     UZ00010

UZ00010  PTF       APPLIED GXY1000

UZ00012  PTF       APPLIED GXY1000  PRE     UZ00010

UZ00014  PTF       APPLIED GXY1000  IFREQ   UZ00015
                                    PRE     UZ00010

UZ00015  PTF       APPLIED HXY1010  REQ     UZ00014

XY10001  USERMOD   APPLIED GXY1000  IFREQ   XY10101

XY10101  USERMOD   APPLIED HXY1010
```

Figure 13. The SYSMOD STATUS REPORT

The ELEMENT SUMMARY Report

This report describes the status of the libraries that were updated for each module, macro, or source module (see Figure 14). The report is not generated when all SYSMODs selected for processing are terminated prior to any element selection.

The fields in the report are as follows:

- ELEM TYPE - The element type: MAC, MOD, SRC, or S/ZAP.
- ELEMENT NAME - The element name.
- ELEM STATUS - Describes what has happened to the element. The possible contents of this field are as follows:

  1) APPLIED, ACCEPTED, or RESTORED - The element was successfully processed.

  2) BYPASS - An error was detected while performing MODID checks, but the ID option was specified in the BYPASS operand. The element was processed.

  3) DELETED - The element was selected and deleted. The DELETE operand was specified on the element modification control statement.

  4) DLIB ER - The value in the DISTLIB operand on the element modification control statement does not match the DISTLIB subentry value in the element entry on the ACDS/CDS. The element is not processed and the SYSMOD will have NOGO status.

  5) ID ERR - An error was detected while performing MODID checks. Check messages on SMPOUT to determine error. The element was not processed.

  6) NOGO - The element was not processed. The SYSMOD STATUS field will contain either NOGO or ERROR. If ERROR status is indicated, the element may have been processed. Check the messages in SMPOUT for status of the library in which the element resides.

7) NOT SEL - Multiple versions of the same element were being processed concurrently. This version of the element was not selected because there was a superior version.

8) SRC SEL - The module was not selected since the source module with the same name was assembled and the resultant object text used instead of the module text supplied with the SYSMOD. This status will only appear if the element type is MOD or S/ZAP.

- CURRENT FMID - The FMID that appears in the CDS element entry for APPLY or RESTORE, or the ACDS element entry for ACCEPT, when processing completes. This will only appear if the element is successfully processed.

- CURRENT RMID - The RMID that appears in the CDS element entry for APPLY or RESTORE, or the ACDS element entry for ACCEPT, when processing completes. This will only appear if the element is successfully processed.

- MAC/SRC SYSLIB - The name of the target system library when TYPE is MAC or SRC. This field contains SMPMTS for macros that do not have a target system library, and SMPSTS for source modules that do not have a target system library. This field is not present for ACCEPT processing.

- MAC/SRC DISTLIB - The name of the distribution library when TYPE is MAC or SRC. This field is not present for APPLY or RESTORE processing.

- DISTSRC LIBRARY - The distribution library of the source module to be assembled when the element type is MAC and ASSEM NAMES are specified.

- ASSEM NAMES - A list of SRC and/or ASSEM modules assembled as a result of a macro modification. ASSEM modules do not exist for ACCEPT processing.

- LOAD MOD - A list of load modules that were link edited and/or copied using the module named in the ELEMENT NAME field. This field is not present for ACCEPT processing.

- LMOD SYSLIB - The name(s) of target system libraries that contained the load module named in the LOAD MOD field and that were updated during APPLY or RESTORE processing. This field is not present for ACCEPT processing.

- MOD DISTLIB - The name of the distribution library. This field is not present for APPLY processing.

- SYSMOD NAME - The identifier of the SYSMOD(s) that modify the element specified in the ELEMENT NAME field.

- SYSMOD STATUS - The status of the SYSMOD specified in the SYSMOD NAME field. The possible values are the same as in the SYSMOD STATUS report.

DATE 78.001 TIME 09:25:47/HMASMP LVL 04.00 SMPRPT OUTPUT
ELEMENT SUMMARY REPORT FOR APPLY CHECK PROCESSING

ELEM TYPE	ELEMENT NAME	ELEMENT STATUS	CURRENT FMID	CURRENT RMID	MAC/SRC SYSLIB	DISTSRC LIBRARY	ASSEM NAMES	LOAD MOD	---LMOD SYSLIB----	SYSMOD NAME	SYSMOD STATUS
MAC	MACRO1	APPLIED APPLIED	HXY1010	HXY1010	MACLIB01		ASSEM1 ASSEM2 MOD001 MOD002			HXY1010 GXY1000	APPLIED APPLIED
MAC	MACRO2	APPLIED APPLIED	GXY1000	GXY1000	MACLIB01		ASSEM1 MOD002			GXY1000 UZ00014	APPLIED APPLIED
MAC	MACRO3	APPLIED	HXY1010	HXY1010	MACLIB01		ASSEM2 MOD003			HXY1010	APPLIED
SRC	MOD001	APPLIED APPLIED APPLIED NOT SEL NOT SEL NOT SEL NOT SEL	HXY1010	HXY1010	SMPSTS					HXY1010 UZ00015 XY10101 GXY1000 UZ00010 UZ00014 XY10001	APPLIED APPLIED APPLIED APPLIED APPLIED APPLIED APPLIED
SRC	MOD002	APPLIED APPLIED	GXY1000	GXY1000	SMPSTS					GXY1000 UZ00010	APPLIED APPLIED
SRC	MOD003	APPLIED	HXY1010	HXY1010	SMPSTS					HXY1010	APPLIED
MOD	MOD001	SRC SEL SRC SEL NOT SEL NOT SEL NOT SEL	HXY1010	XY10101						HXY1010 UZ00015 GXY1000 UZ00010 UZ00014	APPLIED APPLIED APPLIED APPLIED APPLIED
MOD	MOD002	APPLIED APPLIED	GXY1000	GXY1000						GXY1000 UZ00010	APPLIED APPLIED
MOD	MOD003	APPLIED	HXY1010	HXY1010						HXY1010	APPLIED

Figure 14. The ELEMENT SUMMARY Report

The SYSMOD REGRESSION Report

This report describes regressions of previous modifications to elements by SYSMODs that were processed by APPLY or ACCEPT and the BYPASS(ID) operand is specified (see Figure 1). It is not produced for RESTORE processing or when no regressions have been detected by the Element Selection routines. SMP detects regression by the presence of SYSMOD-IDs in the RMID and/or UMID subentries of the element entries on the CDS or ACDS that were not specified in the PRE or SUP operand lists of the ++VER modification control statements of the regressing SYSMODs. If no SYSMODs were regressed, the report consists of the single message "NO SYSMODS REGRESSED."

The following describes the fields within the report:

- REGRESSING SYSMOD - The identifier of the SYSMOD that caused regression of the element(s) listed in the COMMON ELEMENTS fields.

- REGRESSED SYSMOD - A list of SYSMODs that had previously or concurrently modified the element(s) listed in the COMMON ELEMENTS fields. Because these SYSMODs were not specified in the PRE or SUP operands of the ++VER modification control statement of the regressing SYSMOD, it is possible that the existing modifications to the element(s) were lost.

- COMMON ELEMENTS TYPE and NAME - A list of elements modified by the regressing SYSMOD.

- OTHER POTENTIALLY REGRESSED SYSMODS - A list of SYSMODs superseded by the regressed SYSMOD that were not superseded by the regressing SYSMOD. This list may include the SYSMOD-IDs of APARs that were fixed (superseded) by the regressed SYSMOD that were not included in the regressing SYSMOD.

  A regression occurs when one of the following conditions is true:

  - The regressing SYSMOD did not specify the SYSMOD-ID of a UMID subentry in the element entry in the PRE or SUP operand lists of the ++VER modification control statement and the element is being replaced.

  - The regressing SYSMOD did not specify the SYSMOD-ID of the RMID subentry in the element entry in the PRE or SUP operand lists of the ++VER modification control statement and the element is being replaced.

    In either of these cases, the regressing SYSMOD is not processed unless BYPASS(ID) is specified on the APPLY or ACCEPT statement. It is not recommended that you bypass the ID verification checks unless you are sure that the previous modifications were not regressed, or if you intend to modify the element to include the regressed modification.

  A regression is possible if one of the following conditions is true:

  - The regressing SYSMOD did not specify the SYSMOD-ID of the RMID subentry in the element entry in the PRE operand list of the ++VER modification control statement and the element is being updated.

- The regressing SYSMOD did not specify the SYSMOD-ID of a UMID subentry in the element entry in the PRE or SUP operand lists of the ++VER modification control statement and the element is being updated.

  In either of these cases, the regressing SYSMOD is processed, but the results are unpredictable. Therefore, the resulting update should be carefully checked to ensure that previous modifications were not regressed.

```
      DATE 78.001  TIME 09:25:47/HMASMP LVL 04.00 SMPRPT OUTPUT
SYSMOD REGRESSION REPORT FOR APPLY CHECK PROCESSING

REGRESSING     REGRESSED    COMMON ELEMENTS    OTHER POTENTIALLY
SYSMOD         SYSMOD       TYPE   NAME        REGRESSED SYSMODS

UZ00099        UZ00001      MODULE HMABD123    AZ00050  AZ00051  AZ00052

               UZ00002      MACRO  HMAMAC01    AZ00055
                            MODULE HMABD456

               UZ00003      MODULE HMABD789    AZ00056  AZ00057
                            MODULE HMABD012
                            MODULE HMABD345

UZ00111        UZ00004      MODULE HMABD678    AZ00058  AZ00059  AZ00060
                            MODULE HMABD987
                            MODULE HMABD124
                            MODULE HMABD135
```

Figure 15. The SYSMOD REGRESSION Report

The DELETED FUNCTION Report

This report describes the SYSMODs that are deleted when SYSMODs containing the DELETE operand in their ++VER modification control statements are processed (see Figure 16). It is not produced for RESTORE processing or when no DELETE processing has occurred.

The fields in the report are as follows:

- SYSMOD CAUSING THE DELETION - The identifier of the SYSMOD containing the DELETE operand in its ++VER modification control statement.

- DELETED THE FOLLOWING SYSMODS - The type of SYSMOD and SYSMOD-ID of each SYSMOD that was deleted. The SYSMOD-IDs for each type of SYSMOD (FUNCTION, PTF, APAR, USERMOD) are listed from left to right following the TYPE column value. All PTFs, APARs, and USERMODs that are listed in the TYPE column belong to the function SYSMOD listed immediately above them. When TYPE is specified as "FUNCTION", the SYSMOD field value can be one of the following:

1) A SYSMOD-ID only - The SYSMOD was installed on your system or distribution libraries and was specified in the DELETE operand list of the ++VER modification control statement for the deleting SYSMOD.

2) A SYSMOD-ID followed by "FMID(sysmod-id)" - The SYSMOD was implicitly deleted. The FMID operand specifies the SYSMOD-ID of a function SYSMOD that appears earlier in the report that is also deleted. This SYSMOD is considered a dependent or feature level function.

3) A SYSMOD-ID followed by "NOT PREVIOUSLY INSTALLED" - The SYSMOD was specified in the DELETE operand list of the ++VER modification control statement for the deleting SYSMOD, but was not installed on your system or distribution libraries.

4) A SYSMOD-ID followed by "PREVIOUSLY DELETED" - The SYSMOD was specified in the DELETE operand list of the ++VER modification control statement for the deleting SYSMOD, but was previously deleted by another function SYSMOD.

SYSMODs that appear as deleted may remain as entries on the CDS or ACDS because they are specified in the SUP operand list of the deleting function SYSMOD or another SYSMOD processed concurrently.

```
     DATE 78.001  TIME 09:25:47/HMASMP LVL 04.00 SMPRPT OUTPUT
DELETED FUNCTION REPORT FOR APPLY CHECK PROCESSING

DELETING       DELETED SYSMODS
SYSMOD         TYPE       SYSMOD

FYZ3000        FUNCTION   FYZ1000
               PTF        UZ00111  UZ00123  UZ00135
               USERMOD    MY11111

               FUNCTION   GYZ1010  FMID (FYZ1000)
               PTF        UZ00112  UZ00124  UZ00136
               USERMOD    MY11112

               FUNCTION   GYZ1020  FMID (GYZ1010)
               PTF        UZ00142  UZ00164
               APAR       AZ12345

               FUNCTION   FYZ2000  NOT PREVIOUSLY INSTALLED
```

Figure 16. The DELETED FUNCTION Report

Chapter 7: SMP Control Statements

To carry out its functions, SMP has five major control statements (RECEIVE, REJECT, APPLY, RESTORE, and ACCEPT) as well as supporting control statements. Any number of each type of control statement can be coded in an SMP job step.

This chapter describes the SMP control statements in the following alphabetical order:

- ACCEPT - modifies distribution libraries
- APPLY - modifies target system libraries
- ENDUCL - identifies the end of update control language (UCL) statements
- JCLIN - creates or updates CDS entries
- LIST - lists the contents of SMP data sets
- LOG - writes messages to LOG data set
- RECEIVE - places SYSMODs in the PTS data set for subsequent processing by APPLY and ACCEPT
- REJECT - deletes SYSMODs from the PTS data set
- RESETRC - resets return codes from SMP functions
- RESTORE - removes modifications from target system libraries
- UCL - update control language statements used to describe update processing to be done by the UCLIN function.
- UCLIN - used in conjunction with the UCL and ENDUCL statements to update SMP data sets.
- UNLOAD - used to punch CDS or ACDS data in UCLIN format.

A detailed explanation of the processing that takes place for the ACCEPT, APPLY, JCLIN, RECEIVE, REJECT, RESTORE and UCLIN control statements is found in Chapter 3.

Each control statement is described in the following format:

Introduction and Description: The name of the control statement followed by a brief description of the function performed by the statement.

Syntax: Gives the syntax of the control statement. See "Appendix A: Rules for Coding SMP Statements" and "Appendix B: Syntax Notation Conventions" for more information on syntax rules.

Operands: Describes the function of each operand that can be coded with the control statement.

DDnames: Lists the ddnames that must be defined for the control statement.

Programming Considerations: Describes any special considerations and notes applicable to the control statement.

Return Codes: Presents a summary of the possible return code values along with the reasons for each possible return code.

Error Recovery: Gives, where applicable, a brief description of error recovery procedures.

Examples: Presents at least one coding example of the control statement. SMP does not require or suggest that the ddnames used in the examples be used in a particular user installation.

The ACCEPT Control Statement

The SMP ACCEPT control statement places SYSMODs into the distribution libraries (DLIBs) or permanent user libraries. Any number of ACCEPT statements can be included in an SMP job step. Once ACCEPT processing completes, SMP cannot remove the SYSMOD.

ACCEPT Syntax

```
ACCEPT [{SELECT | GROUP | EXCLUDE}
      (sysmodid[,sysmodid]...)]
      [APARS]
      [ASSEM]
      [BYPASS(option[,option]...)]
      [CHECK]
      [COMPRESS({ALL | ddname[,ddname]...})]
      [DIS( READ | NO | WRITE )]
      [NOAPPLY]
      [USERMODS]
      [RC(function=code[,function=code]...)]
      [RETRY(YES | NO)]
      .
```

ACCEPT Operands

SELECT(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be placed into the DLIBs or permanent user libraries. This operand can also be specified as 'S'.

GROUP(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be placed into the DLIBs or permanent user libraries. This operand can also be specified as 'G'. Any requisite and prerequisite SYSMODs are automatically included in the processing (including any requisites and prerequisites of the requisite and prerequisite SYSMODs).

EXCLUDE(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs not to be placed into the DLIBs or permanent user libraries. This operand can also be specified as 'E'.

Note: When none of the above operands is specified, then all SYSMODs that have not been accepted and are otherwise eligible for processing are selected. See the NOAPPLY operand for variations with no operands and the EXCLUDE operand.

APARS
specifies that APAR SYSMODs are to be included where applicable. If this operand is not specified, no APAR SYSMODs are selected for ACCEPT processing.

ASSEM
specifies that SYSMODs that contain both source text and object text for the same modules are to have the source text assembled to replace the object text.

BYPASS(option[,option]...)
specifies conditions that might normally result in the termination of SYSMODs are to be ignored. The options are as follows:

- ID - specifies that error conditions detected during ID checking of the RMID and UMID fields in the element entries on the ACDS should not cause termination of the SYSMODs.

- PRE - specifies that missing prerequisite SYSMODs should not cause termination of the SYSMODs for which they are needed.

- REQ - specifies that missing requisite SYSMODs should not cause termination of the SYSMODs for which they are needed.

- IFREQ - specifies that missing conditional requisite SYSMODs should not cause termination of the SYSMODs for which they are needed.

CHECK
specifies that ACCEPT processing of SYSMODs should not actually update libraries and SMP data sets. Instead, only the following processing is performed:

- Testing for error conditions, with the exception of those that might occur during the updating of the libraries, before accepting the SYSMODs.

- Reporting on libraries that could be updated during ACCEPT processing.

- Reporting on SYSMODs that are or will be regressed during ACCEPT processing.

Note: If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored; no compression is performed.

COMPRESS({ALL | ddname[,ddname]...})
specifies one or more partitioned data sets to be compressed. This operand can be specified as 'C'. Only the partitioned data sets affected by ACCEPT processing are compressed by specifying 'ALL'.

Note: 1. If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored and no compression is performed.

2. The SMPACDS and SMPCDS data sets cannot be compressed. If specified, they are ignored..

DIS(READ | NO | WRITE)
specifies that the SMPACDS directory is to be in storage during processing.

READ is the default; it causes the directory to be in storage in read only mode. Updates to the directory entries are stowed as they occur.

NO specifies that the directory is not to be in storage during processing. All reading of directory entries is done from the data set itself, and updates to the directory entries are stowed as they occur.

WRITE specifies that the directory is to be in storage for both reading and updating. Updates to the directory entries are performed on the in storage copy as they occur; the entire directory is written to the data set when ACCEPT processing completes.

Note: If DIS(NO) is specified with the CHECK operand, it is ignored and DIS(READ), the default value, is used. The directory entries are not updated in CHECK mode.

NOAPPLY
specifies that the SYSMODs have bypassed APPLY processing and are to be placed directly into the DLIBs or permanent user libraries.

Note: If the NOAPPLY operand is specified, then the CDS data set is not required during ACCEPT processing. When the NOAPPLY, SELECT, and GROUP operands are not specified, then only those SYSMODs that have been applied will be selected for ACCEPT processing. The specification of the EXCLUDE operand is considered a mode of mass ACCEPT processing. The specification of the SELECT or GROUP operand is not effected by the NOAPPLY operand; that is, those SYSMODs specified in the SELECT or GROUP operand list are selected for ACCEPT processing even though they may not have been applied and the NOAPPLY operand is not specified.

USERMODS
specifies that USERMOD SYSMODs are to be included where applicable. If this operand is not specified, USERMOD SYSMODs are not selected for ACCEPT processing.

| RETRY(YES | NO)
where 'YES' indicates that SMP is to attempt a RETRY for each utility failure during the function. 'NO' indicates that no RETRY is to be attempted. 'YES' is the default mode of operation if the RETRY keyword is not specified and a DDname list is available.

RC(function=code[,function=code]...)
specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the ACCEPT statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, ACCEPT processing is bypassed and ACCEPT terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.
- If all specified SMP functions return codes less than or equal to their indicated codes, ACCEPT is executed.
- Previous processing by any SMP function not specified on the RC operand has no effect on the current ACCEPT processing.

ACCEPT DDnames

distlib	(one for each different distribution library to be updated)
lklib	(one for each different LKLIB operand value on ++MOD modification control statements, if any)
SMPACDS	(required)
SMPACRQ	(required)
SMPCDS	(required unless the NOAPPLY operand is specified on the ACCEPT control statement)
SMPCNTL	(required)
SMPLOG	(required)
SMPMTS	(required unless the NOAPPLY operand is specified on the ACCEPT control statement, or the SAVESTS or SAVEMTS indicators are set on in the CDS)
SMPOUT	(required)
SMPPTS	(required)
SMPRPT	(optional)
SMPSCDS	(required unless the NOAPPLY operand is specified on the ACCEPT control statement)
SMPSTS	(required unless the NOAPPLY operand is specified on the ACCEPT control statement, or the SAVESTS or SAVEMTS indicators are set on in the CDS)
SMPTLIB	(required if the modifications were loaded to temporary libraries during RECEIVE)
SMPWRK1	(required)
SMPWRK2	(required)
SMPWRK3	(required)
SMPWRK4	(required)
SYSLIB	(required)
SYSPRINT	(required)
SYSUT1	(required)
SYSUT2	(required)
SYSUT3	(required)
SYSUT4	(required for RETRY only)
txlib	(one for each different TXLIB operand value on element modification control statements, if any)

ACCEPT Programming Considerations

1) CAUTION: SMP cannot remove a SYSMOD from the target system after ACCEPT processing.

2) To prevent direct access space problems during ACCEPT processing, COMPRESS should be specified. Note, however, that use of the COMPRESS option might increase processing time significantly.

3) A data set can be specified in the COMPRESS operand list even if it is not affected by any modification in the same ACCEPT pass.

4) COMPRESS does not process keyed or unmovable data sets.

5) The COMPRESS function should not be performed on a running operating system; an alternate system should be used to apply the service or function.

6) When COMPRESS is specified, all elements that are being replaced in DLIBs being compressed are deleted before the compression. Macro elements are not deleted during compression processing before they are replaced, since termination of the SYSMOD containing the macro would cause termination of the SYSMOD's that required the macro for assemblies.

7) SYSMODs have the ACCEPT and ERROR status indicators set in their entries on the ACDS before any updating of elements in the distribution libraries. If processing is unsuccessful, the ERROR indicator remains on with the ACCEPT indicator. The ERROR indicator means that the SYSMOD is not completely accepted, although all the updates might have been done. This condition occurs when a SYSMOD has a requisite relationship with another SYSMOD that did not process successfully. Review the SMPOUT and SYSPRINT output from the ACCEPT processing that failed to determine the cause of error. Use the LIST control statement with the ERROR operand to list SYSMOD entries in the ACDS to determine if the ERROR indicator is set.

8) When modules are link edited into the distribution libraries, external references might be unresolved; therefore, ignore message IEW0461.

9) The ddnames required by ACCEPT for DLIBs can be found in the output of the ACCEPT CHECK function. DD statements must be included in the job step that uses these ddnames to point to the appropriate libraries. Typically, the ddnames used for distribution libraries are usually the lowest level qualifiers of the data set names (that is, AOS12 for SYS1.AOS12).

10) When SYSMODs that had contained TXLIB or LKLIB operands are to be accepted, DD statements must be supplied for each of the ddnames specified in these operand lists.

11) The GROUP and SELECT operands cause ACCEPT processing to try to process the selected SYSMODs, even though they have been previously accepted successfully. If GROUP is specified, only those requisite SYSMODs that have not been successfully processed by ACCEPT are selected for processing.

12) Use the DIS(NO) option only when the number of SYSMODs and their elements is small or when the tradeoff between storage utilization and performance has to be made in favor of storage.

13) The DIS(NO) option should not be used if the previous SMP control statement was ACCEPT or UCLIN specified without the DIS(NO) option and the same directory is to be used.

ACCEPT Return Codes

00 ACCEPT processing completed successfully and without errors.

04 ACCEPT processing completed, but there are possible error or warning messages.

ACCEPT invoked a system program to perform some work and the system program returned a non zero, but still acceptable, return code. One of the following system programs could generate this return code:

- Assembler (ASMBLR)
- IEBCOPY - invoked to copy one or more modules, macros, or source modules, or to compress a data set
- IEBUPDTE - invoked to update or replace source modules or macros
- IMASPZAP - invoked to perform a ZAP operation
- Linkage editor (IEWL)

The affected SYSMOD entries have the ACCEPT status indicator set in the ACDS.

08 ACCEPT processing completed, but processing errors were encountered. At least one SYSMOD had its processing terminated. The possible error conditions are:

1) ACCEPT invoked a system program to perform some work and the system program returned a non zero and unacceptable return code. One of the following system programs could generate this return code:

   - Assembler (ASMBLR)
   - IEBCOPY - invoked to copy one or more modules, macros, or source modules
   - IEBUPDTE - invoked to update or replace source modules or macros
   - IMASPZAP - invoked to perform a ZAP operation
   - Linkage editor (IEWL)

   The affected SYSMOD entries have the ACCEPT and ERROR status indicators set in the ACDS.

2) SMP encountered an error while scanning IMASPZAP control statements. Check SMPOUT output for error messages to determine the cause of the problem. The affected SYSMOD entry has the ACCEPT and ERROR indicators set, although no update has been done to the module unless an EXPAND linkage editor control statement was included in the modification. In this case, the module has been link edited to expand its size in the distribution library.

3) An IMASPZAP VERIFY REJECT was encountered by the IMASPZAP program. Check SYSPRINT output for error messages to determine the cause of the problem. The affected SYSMOD entry has the ACCEPT and ERROR indicators set, although no update has been done to the module unless an EXPAND linkage editor control statement was included in the modification. In this case, the module has been link edited to expand its size in the distribution library.

4) A DD statement was missing. ACCEPT did not process any SYSMOD that required the missing DD statement.

5) A SYSMOD specified in the SELECT or GROUP operand list has an entry in the ACDS that indicates that it has been superseded by another SYSMOD.

6) A TXLIB or LKLIB member cannot be found. The affected SYSMOD entry has the ACCEPT and ERROR status indicators set in the ACDS.

7) A SYSMOD specified in the SELECT or GROUP operand list was not found on the PTS.

8) PEMAX was too small to process one or more SYSMOD entries and/or selected element entries being modified. If the latter situation is true, the affected SYSMOD entries might have the ACCEPT and ERROR status indicators set in the ACDS. Check SMPOUT output for error messages to determine which SYSMODs and/or elements were affected.

9) An error occurred while attempting to open a target system or distribution library. The affected SYSMOD entry might have the ACCEPT and ERROR status indicators set in the ACDS.

12 ACCEPT processing terminated. The possible error conditions are:

1) A function SYSMOD was selected for processing and subsequently terminated before any updating of distribution libraries.

2) No SYSMODs met ACCEPT specifications.

3) A GETMAIN failure occurred during ACCEPT processing.

4) An error occurred while opening or closing an SMP data set.

5) A syntax error was detected in the ACCEPT control statement.

6) The ACCEPT control statement was not processed because a previous control statement returned a non acceptable return code.

7) A DD statement was missing.

16 A severe error was encountered and SMP processing was terminated. The possible error conditions are:

1) IEBCOPY, invoked to compress a data set, returned a non acceptable code. ACCEPT was not executed, but the elements within the subject SYSMODs that were candidates for replacement may have been deleted from the appropriate distribution libraries.

Note: The distribution libraries might be unusable. Examine the IEBCOPY output to determine the status of the data set when IEBCOPY failed.

2) A severe error occurred while accessing an SMP data set.

3) An error occurred while writing a message.

ACCEPT Error Recovery

After completion or abnormal termination of the ACCEPT function, examine SMPOUT and SYSPRINT to determine the relative success of the function. Note that partially applied SYSMODs have the ACCEPT and ERROR status indicators set in the SYSMOD entries on the ACDS. Examine the reports if they have been produced.

You must rerun ACCEPT for a SYSMOD that failed during a previous ACCEPT. After an ACCEPT fails, SMP does not allow any other function other than ACCEPT to be performed on that PTF. If you remove the ERROR status indicator in the ACDS SYSMOD entry and attempt a subsequent RESTORE which will use some or all of the copies of the elements in the distribution libraries supposedly updated or replaced by that SYSMOD, unpredictable results should be expected. The following processing takes place:

- All linkage editor processing is repeated.
- All IEBCOPY processing is repeated.
- All macro and source module updating is repeated.
- All assemblies are repeated.
- All IMASPZAP processes are repeated. However, if any IMASPZAP process completed through the IMASPZAP REPLACE stage, or if any IMASPZAP process produced an IMASPZAP VERIFY REJECT in the previous ACCEPT, this rerun of ACCEPT will also fail. To correct this problem:
  - Use the utility IEBPTPCH to obtain the IMASPZAP control cards from the PTS for the modules involved in the SYSMOD by punching the SYSMOD.
  - REJECT the SYSMOD from the PTS.

- Correct any IMASPZAP modification processed that caused a VERIFY REJECT.

- RECEIVE and ACCEPT the SYSMOD as corrected.

If an out-of-space condition occurs on any library during ACCEPT processing, see "Resolving Direct Access Storage Shortage Problems" in Chapter 5 for information on how to handle the problem. Then rerun ACCEPT with CHECK to determine the appropriate actions.

The APPLY Control Statement

The SMP APPLY control statement places SYSMODs into the target system libraries. Any number of APPLY statements can be included in an SMP job step. APPLY processing does not change the distribution libraries (DLIBS) or permanent user libraries; the SYSMODs can be removed by restoring to the current level of these libraries using the RESTORE control statement.

APPLY Syntax

```
APPLY     [{SELECT | GROUP | EXCLUDE}
            (sysmodid[,sysmodid]...)]
          [ASSEM]
          [BYPASS(option[,option]...)]
          [CHECK]
          [COMPRESS({ALL | ddname[,ddname]...})]
          [DIS( READ | NO | WRITE )]
          [NOJCLIN[(sysmodid[,sysmodid]...)]]
          [NUCID(n)]
          [RC(function=code[,function=code]...)]
          [RETRY(YES | NO)]
          •
```

APPLY Operands

SELECT(sysmodid[,sysmodid]...)
specifies one or more SYSMODs to be placed into the target system libraries. This operand can also be specified as 'S'.

GROUP(sysmodid[,sysmodid]...)
specifies one or more SYSMODs to be placed into the target system libraries. This operand can also be specified as 'G'. Any requisite and prerequisite SYSMODs are automatically included in the processing (including any requisites and prerequisites of the requisite and prerequisite SYSMODs).

EXCLUDE(sysmodid[,sysmodid]...)
specifies one or more SYSMODs not to be placed into the target system libraries. This operand can also be specified as 'E'.

Note: If none of the above operands is specified, then all SYSMODs that have not been accepted and are otherwise eligible for processing are selected.

ASSEM
specifies that SYSMODs that contain both source text and object text for the same modules are to have the source text assembled to replace the object text.

BYPASS(option[,option]...)
specifies that conditions that might normally result in the termination of SYSMODs are to be ignored. The options are as follows:

- ID - specifies that error conditions detected during ID checking of the RMID and UMID fields in the element entries on the CDS should not cause termination of the SYSMODs.

- PRE - specifies that missing prerequisite SYSMODs should not cause termination of the SYSMODs for which they are needed.

- REQ - specifies that missing requisite SYSMODs should not cause termination of the SYSMODs for which they are needed.

- IFREQ - specifies that missing conditional requisite SYSMODs should not cause termination of the SYSMODs for which they are needed.

CHECK
specifies that APPLY processing of SYSMODs should not actually cause libraries and SMP data sets to be updated. Instead, only the following processing is performed:

- Testing for error conditions, with the exception of those that can occur during the updating of the libraries, before applying the SYSMODs.

- Reporting on libraries that would be updated during APPLY processing.

- Reporting on SYSMODs that are or will be regressed during APPLY processing.

Note: If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored and no compression is performed.

COMPRESS({ALL | ddname[,ddname]...})
specifies one or more ddnames of partitioned data sets to be compressed. This operand can be specified as 'C'. Only the partitioned data sets affected by APPLY processing are compressed by specifying 'ALL'.

Note: 1. If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored and no compression is performed.

2. The SMPACDS and SMPCDS data sets cannot be compressed. If either or both of these ddnames is specified, they are ignored.

DIS(READ | NO | WRITE)
specifies that the SMPCDS directory is to be in storage during processing.

READ is the default. It causes the directory to be in storage in read only mode. Updates to the directory entries are stowed as they occur.

NO specifies that the directory is not to be in storage during processing. All reading of directory entries is done from the data set itself and updates to the directory entries are stowed as they occur.

WRITE specifies that the directory is to be in storage for both reading and updating. Updates to the directory entries are performed on the in-storage copy as they occur, and the entire directory is written to the data set when APPLY processing completes.

Note: If DIS(NO) is specified with the CHECK operand, it is ignored and DIS(READ), the default value, is used. The directory entries are not updated in CHECK mode.

NOJCLIN[(sysmodid[,sysmodid]...)]
specifies that inline JCLIN processing for all or specified SYSMODs is to be omitted.

NUCID(n)
Specifies the digit at the end of the IEANUCOn module under which the current nucleus is to be saved during APPLY processing. This operand overrides the NUCID operand specified when the CDS SYSTEM entry was created. The overriding is done only for the APPLY statement that contains this parameter.

| RETRY(YES | NO)
where 'YES' indicates that SMP is to attempt a RETRY for each utility failure during the function. 'NO' indicates that no RETRY is to be attempted. 'YES' is the default mode of operation if the RETRY keyword is not specified

and a DDname list is available.

RC(function=code[,function=code]...)
specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE,

REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the APPLY statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, APPLY processing is bypassed and APPLY terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.

- If all specified SMP functions return codes less than or equal to their indicated codes, APPLY is executed.

- Previous processing by any SMP function not specified on the RC operand has no effect on the current APPLY processing.

APPLY DDnames

distlib	(for macro and source libraries if there are no corresponding macro or source target libraries and the modification being applied is an update)
lklib	(one for each different LKLIB operand value on ++MOD modification control statements, if any)
SMPCDS	(required)
SMPCNTL	(required)
SMPCRQ	(required)
SMPLOG	(required)
SMPMTS	(required)
SMPOUT	(required)
SMPPTS	(required)
SMPRPT	(optional)
SMPSCDS	(required)
SMPSTS	(required)
SMPTLIB	(required if the modifications were loaded to temporary libraries during RECEIVE)
SMPWRK1	(required)
SMPWRK2	(required)
SMPWRK3	(required)
SMPWRK4	(required)
SMPWRK5	(required)
SYSLIB	(required)
SYSPRINT	(required)

SYSUT1 (required)
SYSUT2 (required)
SYSUT3 (required)
SYSUT4 (required for RETRY)
txlib (one for each different TKLIB operand value on element modification control statements, if any)
tgtlib (one for each target system library being updated)

APPLY Programming Considerations

1) To prevent direct access space problems during APPLY processing, COMPRESS should be specified. Note, however, that use of the COMPRESS option might increase processing time significantly.

2) A data set can be specified in the COMPRESS operand list even if it is not affected by any modification in the same APPLY pass.

3) COMPRESS will not process keyed or unmovable data sets.

4) The COMPRESS function should not be performed on a running operating system; an alternate system should be used to apply the service or function.

5) If SYSMODs selected for APPLY processing replace modules or source modules that were copied to target system data sets at SYSGEN, and the COMPRESS operand is specified on the APPLY control statement for those data sets, the modules or source modules are deleted during APPLY compression processing before they are replaced by elements in the SYSMODs selected for APPLY. Macro elements are not deleted during compression processing before they are replaced, since termination of the SYSMOD containing the macro would cause the termination of other SYSMOD's that require the macro for assemblies.

6) SYSMODs have the APPLY and ERROR status indicators set in their entries on the CDS before any updating of elements in the target system libraries. If processing is unsuccessful, the ERROR indicator remains on with the APPLY indicator. The ERROR indicator means that the SYSMOD is not completely applied, although all the updates may have been done. This condition occurs when a SYSMOD has a requisite relationship with another SYSMOD that did not process successfully. Review the SMPOUT and SYSPRINT output from the APPLY processing that failed to determine the cause of error. Use the

LIST control statement with the ERROR keyword to list SYSMOD entries in the CDS to determine if the ERROR indicator is set.

7) The ddnames required by APPLY for target libraries can be found in the output from the APPLY CHECK function. DD statements must be included in the job step that uses these ddnames to point to the appropriate libraries. Typically, the ddnames used for target libraries are usually the lowest level qualifiers of the data set names (that is, TCAMLIB for SYS1.TCAMLIB).

8) When SYSMODs that contain TXLIB or LKLIB operands are to be applied, DD statements must be supplied for each of the ddnames specified as values of these operands.

9) Nucleus backup capability is lost if the same NUCID is specified in two or more APPLY statements that affect the nucleus.

10) The saved nucleus is not used to replace the current nucleus restored during RESTORE processing. The saved nucleus is only used to provide you with an alternate nucleus for IPL in case an applied SYSMOD damaged the current nucleus. To provide room for link edits required when applying service, enough space should be allocated for the nucleus data set (SYS1.NUCLEUS) to hold at least three copies of the nucleus.

11) The GROUP and SELECT operands cause APPLY processing to try to process the selected SYSMOD(s) even though they have been previously applied successfully. If GROUP is specified, only those requisite SYSMODs that have not been successfully processed by APPLY are selected for processing.

12) The NOJCLIN operand can be used to circumvent processing of inline JCLIN when reapplying SYSMODs if JCLIN data would change the content of CDS entries that should not be changed. You should check the inline JCLIN carefully for a SYSMOD that is being reapplied. This checking is to ensure that processing of data will not change updates to CDS entries made after the SYSMOD was originally applied.

13) The DIS(NO) option should be used only when the number of SYSMODs and their elements is small or if the tradeoff between storage utilization and performance has to be made in favor of storage.

14) Specification of DIS(NO) when processing SYSMODs that have inline JCLIN might cause the processing time to increase significantly.

15) The DIS(NO) option should not be used when the previous SMP control statement was APPLY, RESTORE, JCLIN, or UCLIN specified without the DIS(NO) option and the same directory is to be used.

APPLY Return Codes

00 APPLY processing completed successfully and without errors.

04 APPLY processing completed, but there are possible error or warning messages. The possible error conditions are:

1) APPLY invoked a system program to perform some work and the system program returned a non zero, but still acceptable, return code. One of the following system programs could generate this return code:

- Assembler (ASMBLR)
- IEBCOPY - invoked to copy modules, macros, or source modules, or to compress a data set
- IEBUPDTE - invoked to update or replace source modules or macros
- IMASPZAP - invoked to perform a ZAP operation
- Linkage editor (IEWL)

The affected SYSMOD entries have the APPLY status indicator set in the CDS.

2) No assembler input could be found in either the CDS or the distribution library specified in the DISTSRC or ASMLIB operand list when APPLY attempted to reassemble a module because of a macro modification. The module was not reassembled, but the APPLY status indicator was set for the affected SYSMOD entries in the CDS.

08 APPLY processing completed, but processing errors were encountered. At least one SYSMOD had its processing terminated. The possible error conditions are:

1) APPLY invoked a system program to perform some work, and the system program returned a non zero and unacceptable return code. One of the following system programs could generate this return code:

   - Assembler (ASMBLR)
   - IEBCOPY - invoked to copy modules, macros, or source modules
   - IEBUPDTE - invoked to update or replace source modules or macros
   - IMASPZAP - invoked to perform a ZAP operation
   - Linkage editor (IEWL)

   The affected SYSMOD entries have the APPLY and ERROR status indicators set in the CDS.

2) SMP encountered an error while scanning IMASPZAP control statements. Check SMPOUT output for error messages to determine the cause of the problem. The affected SYSMOD entry has the APPLY and ERROR indicators set, although no update has been done to the module unless an EXPAND linkage editor control statement was included in the modification. In this case, the module was link edited to expand its size and the load module was replaced in the target system library.

3) An IMASPZAP VERIFY REJECT was encountered by program IMASPZAP. Check SYSPRINT output for error messages to determine the problem. The affected SYSMOD entry has the APPLY and ERROR indicators set, although no update will have been done to the module unless an EXPAND Linkage Editor control statement was included in the modification. In this case, the module will have been link edited to expand the size and the load module replaced in the target system library.

4) A DD statement was missing. APPLY did not process any SYSMOD that required the missing DD statement.

5) A SYSMOD specified in the SELECT or GROUP operand list has an entry in the CDS that indicates it has been superseded by another SYSMOD.

6) A TXLIB member cannot be found. The affected SYSMOD entry has the the APPLY and ERROR status indicators set in the CDS.

7) A SYSMOD specified in the SELECT or GROUP operand list was not found on the PTS.

8) PEMAX was too small to process one or more SYSMOD entries and/or selected element entries being modified. If the latter situation is true, the affected SYSMOD entries may have the APPLY and ERROR status indicators set in the CDS. Check SMPOUT output for error messages to determine which SYSMODs and/or elements were affected.

9) An error occurred while attempting to open a target system or distribution library. The affected SYSMOD entry may have the APPLY and ERROR status indicators set in the CDS.

12 APPLY processing terminated. The possible error conditions are:

1) A function SYSMOD was selected for processing and terminated before any updating of target system libraries.

2) No SYSMODs met APPLY specifications.

3) A GETMAIN failure occurred during APPLY processing.

4) An error occurred while opening or closing an SMP data set.

5) A syntax error was detected in the APPLY control statement.

6) The APPLY control statement was not processed because a previous control statement returned a non acceptable return code.

7) A DD statement was missing.

16 A severe error was encountered and SMP processing was terminated. The possible error conditions are:

1) IEBCOPY, invoked to compress a data set, returned a non acceptable return code. APPLY processing did not occur, but the elements within the subject SYSMODs that were candidates for replacement may have been deleted from the appropriate target system libraries.

   Note: The target system libraries might be unusable. Examine the IEBCOPY output to determine the status of the data set when IEBCOPY failed.

2) A severe error occurred while deleting modules from a target system library before compression of that data set.

   Note: The target system libraries might be unusable. Examine the IEBCOPY output to determine the status of the data set when IEBCOPY failed.

3) A severe error occurred while accessing an SMP data set.

4) An error occurred while writing a message.

5) A non acceptable return code was returned from IEHIOSUP.

APPLY Error Recovery

After completion or abnormal termination of the APPLY function, examine SMPOUT and SYSPRINT output to determine the relative success of the function. Note that partially applied SYSMODs have the APPLY and ERROR status indicators set in the SYSMOD entries on the CDS.

You can rerun APPLY for a SYSMOD that has failed by correcting any conditions that caused the SYSMOD to be terminated. If a SYSMOD that failed APPLY processing had inline JCLIN that was successfully processed , you should specify the NOJCLIN keyword with that SYSMOD-ID as an operand on the APPLY control statement for the subsequent reapplication. The following processing takes place:

- All linkage editor processing is repeated.
- All IEBCOPY processing is repeated.
- All macro and source updating is repeated.
- All assemblies are repeated.
- All IMASPZAP processes are repeated. However, if any IMASPZAP processing completed through the IMASPZAP REPLACE stage, or if any IMASPZAP process produced an IMASPZAP VERIFY REJECT in the previous APPLY, this rerun of APPLY will fail. To correct this problem:
  - Use the utility IEBPTPCH to obtain the IMASPZAP control cards from the PTS for the modules involved in the SYSMOD by punching the SYSMOD.
  - REJECT the SYSMOD from the PTS.
  - Correct any IMASPZAP processed that caused a VERIFY REJECT.
  - RECEIVE and APPLY the corrected SYSMOD.

If an out-of-space condition occurs on any library during APPLY processing, see "Resolving Direct Access Storage Shortage Problems" in Chapter 5 for information on how to handle the problem. Then rerun APPLY with CHECK to determine the appropriate actions.

The ENDUCL Control Statement

The SMP ENDUCL control statement identifies the end of the update control language (UCL) statements and signifies the end of UCLIN processing. ENDUCL must immediately follow the last UCL statement.

ENDUCL Syntax

```
ENDUCL  •
```

ENDUCL Operands

The ENDUCL control statement has no operands.

ENDUCL DDnames

See "UCLIN DDnames" under "The UCLIN Control Statement" later in this chapter.

ENDUCL Programming Considerations

The ENDUCL control statement must terminate the UCL statements.

ENDUCL Return Codes

See "UCLIN Return Codes" under "The UCLIN Control Statement" later in this chapter.

The JCLIN Control Statement

The JCLIN Control Statement reads in the Stage I output from system generation (or similar job step JCL) to create or update the CDS. Any number of JCLIN statements can be included in an SMP job step.

JCLIN Syntax

```
JCLIN [ASM({PGM=name | procname})]
      [COPY({PGM=name | procname})]
      [DIS( NO | READ | WRITE ]
      [LKED({PGM=name | procname})]
      [UPDATE({PGM=name | procname})]
      [RC(function=code[,function=code]...)]
      .
```

JCLIN Operands

ASM({PGM=name | procname})
specifies an additional assembler name or procedure name to replace the ones assumed by SMP, which are program names ASMBLR, IFOX00, IEUASM and procedure ASMS. See the 'JCLIN Programming Considerations'.

COPY({PGM=name | procname})
specifies an additional copy program name or procedure name to replace the one assumed by SMP, which is program name IEBCOPY.

DIS(NO | READ | WRITE)
specifies that the SMPCDS directory is to be in storage during processing.

NO specifies that the directory is not to be in storage during processing. All reading of directory entries is done from the data set itself and updates to the directory entries are stowed as they occur.

READ specified that the directory is to be in storage for read only mode. Updates to the directory are stowed as they occur.

WRITE specifies that the directory is to be in storage for both reading and updating. Updates to the directory entries are performed on the in-storage copy as they occur; the entire directory is written to the data set when JCLIN processing completes. This is the default mode.

LKED({PGM=name | procname})
specifies an additional linkage editor name or procedure name to replace those assumed by SMP, which are program names IEWL and HEWL, and procedure name LINKS. See 'JCLIN Programming Considerations'.

UPDATE({PGM=name | procname})
specifies an additional update program name or procedure name to replace the one assumed by SMP, which is program name IEBUPDTE.

Note: This operand is used only to ensure that the SYSIN data for the update program is not processed because it can contain JCL statements.

RC(function=code[,function=code]...)
specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the JCLIN statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, JCLIN processing is bypassed and JCLIN terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.

- If all specified SMP functions return codes less than or equal to their indicated codes, JCLIN is executed.

- Previous processing by any SMP function not specified on the RC operand has no effect on the current JCLIN processing.

JCLIN DDnames

```
SMPCDS      (required)
SMPCNTL     (required)
SMPJCLIN    (required)
SMPLOG      (required)
SMPOUT      (required)
```

JCLIN Programming Considerations

1) The input for JCLIN must be free of JCL errors or other syntax errors and must be a job stream similar to that used for system generation.

2) If you specify the ASM operand, you must specify the assembly steps using the program names and procedure names as they appear in the input job stream. For example:

```
PGM=IFOX00

PGM=IEUASM

PGM=ASMBLR

PGM=name

ASMS

or procname
```

   Indicate the member name (modname) of the assembled object module as:

```
   MOD=modname (when using ASMS or procname)

   //SYSPUNCH DD DSN=library(modname),...

or //SYSPUNCH DD DSNAME=library(modname),...
```

3) If you specify the LKED operand, you must specify the linkage editor steps as they appear in the input job stream. For example:

PGM=IEWL

PGM=HEWL

PGM=name

LINKS

or procname

Indicate the data set name (libname) for the output of the linkage editor as:

NAME=libname (when using LINKS or procname)

//SYSLMOD DD DSN=index.libname,...

or //SYSLMOD DD DSNAME=index.libname,...

<u>Note</u>: For overlay structures only, all CSECTs must be explicitly defined with linkage editor INSERT control cards, including cards for those CSECTs within the root segment.

4) If you specify the COPY operand, you must specify the program name or procedure names as they appear in the input job stream. For example:

PGM=IEBCOPY

PGM=name

or procname

Specify the COPY statement as:

COPY INDD=ddname1,OUTDD=ddname2

- where ddname1 and ddname2 are the lowest level qualifiers on the data set names on the respective DD cards (for example, INDD=CI505 and OUTDD=LINKLIB).

5) The SYSIN data set must be the last data set specified in each job step in the SMPJCLIN input.

6) After a complete system generation, the ACDS must be copied, using IEBCOPY, to the new CDS before JCLIN processing. This ensures that the initial CDS entries match those of the ACDS and contain entries that are not created by JCLIN processing.

7) After a partial system generation (that is, a device generation), the output of Stage I must be input to JCLIN processing to ensure that:

   - Module, macro, and load module entries in the CDS are updated.

   - New assembler entries are stored with the new assembler input in the CDS.

   - Linkage editor control statements for load module entries are replaced except for linkage editor CHANGE and REPLACE control statements that were carried over to the updated version.

8) Specification of DIS(NO) or DIS(READ) when processing JCLIN might cause the processing time to increase significantly.

9) The DIS(NO) option should not be used when the previous SMP control statement was APPLY, ACCEPT, RESTORE, JCLIN, or UCLIN specified without the DIS(NO) option and the same directory is to be used.

10) If you are superseding a previous SYSMOD that contained JCLIN, and the SYSMOD that you are processing requires that JCLIN, then it must also be included in the new SYSMOD.

11) The linkage editor control statement IDENTIFY should not be used as input for JCLIN.

JCLIN Return Codes

00 JCLIN processing completed successfully and without errors.

04 JCLIN processing completed, but a premature end of file was encountered for the JCLIN input data set.

08 JCLIN processing terminated because a syntax error was encountered in the JCLIN input.

12 JCLIN processing terminated. The possible error conditions are:

1) A syntax error existed in the JCLIN control statement.

2) Not enough storage was available.

3) Directory space was exceeded on the CDS.

4) PEMAX was too small to process one or more entries on the CDS.

5) A DD statement was missing.

6) The JCLIN control statement was not processed because a previous control statement returned a non acceptable return code.

16 A severe error was encountered and SMP processing was terminated.

JCLIN Error Recovery

If an error occurs in the JCLIN data set, examine SMPOUT output to determine the job, job step, and record that caused the error. Correct the problem and rerun JCLIN. If the DIS keyword was not specified or was specified with the NO or READ options, all jobs, steps, and records up to the point of the error have been processed and the appropriate updates were made to the CDS. The JCLIN rerun repeats the updates that have occurred.

If the error occurred in your user-specified JCLIN input data set, see Chapter 2 for further information.

If an out-of-space condition occurred on the SMPCDS during JCLIN processing, see "Resolving Direct Access Storage Problems" in Chapter 5 for information on how to handle the problem, and then rerun JCLIN.

The LIST Control Statement

The SMP LIST control statement enables you to request a listing on SMPOUT or, optionally, on SMPLIST of:

- All data or selected data from the ACDS, ACRQ, CDS, CRQ, PTS and SCDS data sets.
- The contents of the LOG data set.

The listings can be used to determine the status of your system and the success of the processing performed. Any number of LIST statements can be included in an SMP job step.

LIST Syntax

The syntax shown below includes the operands that cause each type of data set to be listed. Because the options differ for each type, the syntax and operands for each specific type of data set are shown in the sections that follow.

```
LIST [ACDS |
      ACRQ |
      CDS |
      CRQ |
      LOG |
      PTS |
      SCDS]
      [option[,option]...]
      .
```

LIST Operands

There is no default data set; one of the following must be specified.

ACDS
: specifies that all or selected information from the ACDS is to be printed.

ACRQ
: specifies that all or selected information from the ACRQ is to be printed.

CDS
: specifies that all or selected information from the CDS is to be printed.

CRQ
: specifies that all or selected information from the CRQ is to be printed.

LOG
: specifies that the contents of the LOG data set are to be printed.

PTS
: specifies that all or selected information from the PTS is to be printed.

SCDS
: specifies that all or selected information from the SCDS is to be printed.

option
: specifies the options that you need for the ACDS, ACRQ, CDS, CRQ, LOG, PTS,or SCDS operand. For the syntax and explanations of the options, see the descriptions that follow for each data set type.

If you list a data set without specifying any options, the listing produced, by default, contains all of the information for each option that can be listed for that data set with the exception of the XREF information. XREF information must be explicitly requested.

LIST ACDS Syntax

```
LIST ACDS

     [XREF]

     [MAC[(macname[,macname]...)]]

     [MOD[(modname[,modname]...)]]

     [SRC[(srcname[,srcname]...)]]

     [SYSMOD[(sysmodid[,sysmodid]...)]
        [APAR] [DELETE]  [ERROR] [FUNCTION]  [NOAPPLY] [NOSUP]
        [PTF] [SUP] [USERMOD]]

     [SYS]

     .
```

LIST ACDS Operands

ACDS
specifies that all or selected information from the ACDS is to be printed.

If XREF is the only operand specified, all MAC, MOD, SRC, SYS, and SYSMOD information is listed, as well as the XREF information.

XREF
specifies that SMP is to generate the following additional information as part of the listing for each MAC, MOD, SRC, and SYSMOD entry:

MAC	SYSMOD history
MOD	SYSMOD history
SRC	SYSMOD history and macro cross reference
SYSMOD	SYSMODs that reference the listed SYSMOD in NPRE, PRE, REQ, or SUP operands of their ++VER modification control statements and in REQ operands of their ++IF modification control statements.

Descriptions of the additional information provided appears in the MAC, MOD, SRC and SYSMOD operand descriptions. You should be aware that SMP uses extra time and more storage to generate the additional data requested by the XREF keyword.

MAC[(macname[,macname]...)]
specifies that information for all MAC entries or the specified MAC entries is to be listed. This information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the macro.

UMID
a list of the SYSMOD-IDs for SYSMODs that updated the macro.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID or 'UCLIN' and the last type of update made to the entry.

GENASM
a list of the ASSEM and SRC entries that are reassembled when this macro is changed.

SYSMOD HISTORY
the SYSMOD-ID, type, and status for each SYSMOD that contains a ++MAC, ++MACUPD, or ++UPDTE modification control statement for the macro. This information is produced only when you specify the XREF keyword.

See Figure 20 for an example of output from LIST CDS MAC XREF, which contains the same type of information as the output from LIST ACDS MAC XREF.

MOD[(modname[,modname]...)]
specifies that information for all MOD entries or the specified MOD entries is to be listed. This information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the module.

UMID
a list of SYSMOD-IDs for the SYSMODs that updated the module.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID or 'UCLIN' and the last type of update made to the entry.

LMODS
a list of the load modules that include the module.

SYSMOD HISTORY
the SYSMOD-ID, type, and status for each SYSMOD that contains a ++MOD or ++ZAP modification control statement for the module. This information is produced only when you specify the XREF keyword.

See Figure 21 for an example of output from LIST CDS MOD XREF, which contains the same type of information as the output from LIST ACDS MOD XREF.

SRC[(srcname[,srcname]...)]
specifies that information for all SRC entries or the specified SRC entries is to be listed. This information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the source module.

UMID
a list of the SYSMOD-IDs for SYSMODs that updated the source module.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID or 'UCLIN' and the last type of update made to the entry.

MACROS
a list of the MAC entries with a GENASM subentry for the source module. This information is produced only when you specify the XREF keyword.

SYSMOD HISTORY
the SYSMOD-ID, type, and status for each SYSMOD that contains a ++SRC or ++SRCUPD modification control statement for the source module. This

information is produced only when you specify the XREF keyword.

See Figure 22 for an example of output from the LIST CDS SRC XREF, which contains the same type of information as the output from the LIST ACDS SRC XREF.

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that information for all SYSMOD entries or the specified SYSMOD entries is to be listed. This information includes:

TYPE
the type of SYSMOD ('APAR', 'FUNCTION', 'PTF', 'USERMOD', or SUPERSEDED').

FMID
the SYSMOD-ID from the ++VER or ++FUNCTION modification control statement.

JCLIN
an indicator that there is inline JCLIN within the SYSMOD.

IF MCS
an indicator that there are ++IF modification control statement within the SYSMOD.

STATUS

'BYP' if the SYSMOD was accepted using the BYPASS keyword
'ERR' if the SYSMOD was not successfully accepted
'REC' if the SYSMOD was received
'APP' if the SYSMOD was applied
'ACC' if the SYSMOD was accepted
'RGN' if the SYSMOD was accepted.

DATE/TIME
the date and time stamps for RECEIVE, ACCEPT, and UCLIN processing for the SYSMOD.

LASTSUP
the last SYSMOD processed that superseded this SYSMOD.

SREL, DELETE, PRE, NPRE, REQ, SUP, and VERSION
the contents of the keyword lists from the ++VER modification control statement used by ACCEPT processing.

MAC, MACUPD, MOD, SRC, SRCUPD, SZAP, and XZAP
the names from element modification control statements included in the SYSMOD.

RMAC, RMACUPD, RMOD, RSRC, RSRCUPD, RSZAP, and RXZAP
: the names from element modification control statements included in the SYSMOD that represent regressed modifications. A regression occurs when a subsequent SYSMOD did not specify this SYSMOD in the PRE or SUP operand of its ++VER modification control statement.

ASSEM
: the names of modules to be assembled as a result of macro or source changes contained in a SYSMOD.

SUPBY
: a list of SYSMODs that supersede this SYSMOD; that is, SUP is specified in their ++VER modification control statements.

DELBY
: a SYSMOD that deletes this SYSMOD; that is, DELETE is specified in its ++VER modification control statement.

IFREQBY
: a list of SYSMODs that specify this SYSMOD as a requisite SYSMOD (REQ) in a ++IF modification control statement. This information is produced only when you specify the XREF keyword.

NPREBY
: a list of SYSMODs that specify this SYSMOD as negative prerequisites (NPRE) in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

PREBY
: a list of SYSMODs that specify this SYSMOD as prerequisite SYSMODs (PRE) in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

REQBY
: a list of SYSMODs that specify this SYSMOD as requisite SYSMODs (REQ) in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

VERSIONBY
: a list of SYSMODs that specify this SYSMOD as a versioned SYSMOD in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

See Figure 23 for an example of output from LIST CDS

SYSMOD XREF, which contains the same type of information as the output from LIST ACDS SYSMOD XREF.

You can restrict the selection of SYSMOD entries to be listed by specifying the SYSMOD operand with one or more of the following operands. For example, if you specify 'LIST ACDS SYSMOD ERROR.', SMP lists all of the SYSMOD entries in the ACDS that have the ERROR indicator set on.

If you specify more than one operand, SMP combines the operands into one logical request. For example, if you specify 'LIST ACDS SYSMOD APAR PTF ERROR SUP.', SMP lists all of the APAR and PTF entries that have the ERROR indicator set on and that are superseded. Specifying both SUP and NOSUP at the same time causes a syntax error.

APAR
: specifies that APAR SYSMODs are to be listed.

DELETE
: specifies that function SYSMODs that have been deleted from the CDS by other function SYSMODs are to be listed. This operand can be abbreviated as 'DEL'.

ERROR
: specifies that SYSMODs that have the ERROR indicator set are to be listed. This operand can be abbreviated as 'ERR'.

FUNCTION
: specifies that all function SYSMODs are to be listed. This operand can be abbreviated as 'FUNC'.

NOAPPLY
: specifies that SYSMODs that have been received and accepted, but not applied are to be listed. Both the CDS and the ACDS data sets must be available when NOAPPLY is coded. A SYSMOD is considered applied when the SYSMOD entry exists on the CDS with the ERROR status indicator set off. This operand can be abbreviated as 'NOAPP'.

NOSUP
: specifies that only SYSMODs that have not been superseded are to be listed.

: This operand is mutually exclusive with the SUP operand. Specification of both causes a syntax error.

PTF
specifies that all PTF SYSMODs are to be listed.

SUP
specifies that only superseded SYSMODs are to be listed.

This operand is mutually exclusive with the NOSUP operand. Specification of both causes a syntax error.

USERMOD
specifies that all USERMOD type SYSMODs are to be listed. This operand can be abbreviated as 'USER'.

SYS
specifies that system information, such as the default NUCID, system type and release, and the identifier of the ACDS, is to be listed.

See Figure 24 for an example of output from LIST CDS SYS, which contains the same type of information as output from LIST ACDS SYS.

LIST ACDS Exception Reports

There are two possible exception reports from LIST ACDS processing; the LIST MASS SUMMARY REPORT FOR SMPACDS and the LIST SELECT SUMMARY REPORT FOR SMPACDS. The reports are produced at the end of your LIST output if any of the following exceptions are found:

- If you list the ACDS without specifying any other operands, and there are no MAC, MOD, SRC,or SYSMOD entries, then 'xxxx ENTRIES NOT FOUND' where 'xxxx' is MAC, MOD, SRC,or SYSMOD, appears in the LIST MASS SUMMARY REPORT FOR SMPACDS. See Figure 31 for an example of output from LIST MASS SUMMARY REPORT FOR SMPPTS, which contains the same type of information as output from LIST MASS SUMMARY REPORT FOR SMPACDS.

- If you list the ACDS and specify the MAC, MOD, SRC,or SYSMOD operand but do not qualify the operand with a particular entry name, and no MAC, MOD, SRC or SYSMOD entries are found, then 'xxxx ENTRIES NOT FOUND' where 'xxxx' is MAC, MOD, SRC, or SYSMOD appears in the LIST MASS SUMMARY REPORT FOR SMPACDS.

- If you list the ACDS, specify the MAC, MOD, SRC, or SYSMOD operand and qualify the operand with a particular entry name, but that entry name is not found, then 'xxx yyyyyyyy' where 'xxx' is MAC, MOD, SRC or SYSMOD and 'yyyyyyyy' is the entry name appears in the LIST SELECT SUMMARY REPORT FOR SMPACDS.

- If you list all the ACDS SYSMOD entries, qualified by the APAR, DELETE, ERROR, FUNCTION, NOAPPLY, NOSUP, PTF, SUP or USERMOD keywords, and that type of entry is not found, then 'SYSMOD ENTRIES NOT FOUND' appears in the LIST MASS SUMMARY REPORT FOR SMPACDS'

- If you list specific ACDS SYSMOD entries, qualified by the APAR, DELETE, ERROR, FUNCTION, NOAPPLY, NOSUP, PTF, SUP or USERMOD keywords, and that specific entry is not found, then 'SYSMOD xxxxxxxx' where 'xxxxxxxx' is the SYSMOD-ID appears in the LIST SELECT SUMMARY REPORT FOR SMPACDS as not found. In addition, the entries that you selected can be considered ineligible for the following reasons:

keyword	exception
APAR	the SYSMOD is not an APAR
DELETE	the DELETE status is not on
ERROR	the ERROR status is not on
FUNCTION	the SYSMOD is not a function
NOAPPLY	the SYSMOD exists in the CDS
NOSUP	the SYSMOD is superseded
PTF	the SYSMOD is not a PTF
SUP	the SYSMOD is not superseded
USERMOD	the SYSMOD is not a user modification

LIST ACRQ Syntax

```
LIST ACRQ

     [SYSMOD[(sysmodid[,sysmodid]...)]]

     [FMID[(sysmodid[,sysmodid]...)]]

     .
```

LIST ACRQ Operands

ACRQ
specifies that all or selected information from the ACRQ is to be listed. If no other operands are specified, all SYSMOD entries and all FMID entries (without requisite SYSMOD information) are to be listed.

See Figure 25 for an example of output from LIST CRQ, which contains the same type of information as output from LIST ACRQ.

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that all SYSMOD entries or selected SYSMOD entries from the ACRQ are to be listed. The information listed includes, for each SYSMOD entry, the SYSMOD-ID specified in the FMID operand, and the requisite SYSMOD-IDs from the ++IF modification control statements.

See Figure 26 for an example of output from LIST CRQ SYSMOD, which contains the same type of information as output from LIST ACRQ SYSMOD.

FMID[(sysmodid[,sysmodid]...)]
specifies that all FMID entries or selected FMID entries on the ACRQ are to be listed with the requisite SYSMOD information from the corresponding SYSMOD entries. Information printed includes the SYSMOD-ID specified in the FMID operand, the IDs of the SYSMODs that reference the SYSMOD-ID specified in the FMID operand, and the IDs of the SYSMODs that must be ACCEPTed when the function SYSMOD specified as the FMID is accepted.

See Figure 27 for an example of output from LIST CRQ FMID, which contains the same type of information as output from LIST ACRQ FMID.

LIST CDS Syntax

```
LIST CDS

     [XREF]

     [ASSEM[(asmname[,asmname]...)]]

     [DLIB[(dlibname[,dlibname]...)]]

     [LMOD[(modname[,modname]...)]]

     [MAC[(macname[,macname]...)]]

     [MOD[(modname[,modname]...)]]

     [SRC[(srcname[,srcname]...)]]

     [SYSMOD[(sysmodid[,sysmodid]...)]
        [APAR] [DELETE] [ERROR]  [FUNCTION] [NOACCEPT] [NOSUP]
        [PTF] [SUP] [RESTORE] [USERMOD]]

     [SYS]

     •
```

LIST CDS Operands

CDS
: specifies that all or selected information from the CDS is to be printed.

  If XREF is the only operand specified, all ASSEM, DLIB, LMOD, MAC, MOD, SRC, SYS, and SYSMOD information is listed as well as the XREF information.

XREF
: specifies that SMP is to generate the following information as part of the listing for each ASSEM, LMOD, MAC, MOD, SRC and SYSMOD entry:

ASSEM	macro cross reference
LMOD	module cross reference
MAC	SYSMOD history
MOD	SYSMOD history
SRC	SYSMOD history and macro cross reference
SYSMOD	SYSMODs that reference the listed SYSMOD in NPRE, PRE, REQ, or SUP operands of their ++VER modification control statements and in REQ operands of their ++IF modification

control statements

Descriptions of the information provided appears in the ASSEM, LMOD, MAC, MOD, SRC and SYSMOD operand descriptions. You should be aware that SMP uses extra time and more storage to generate the additional data.

ASSEM[(asmname[,asmname]...)]
specifies that information for all ASSEM entries or the specified ASSEM entries is to be listed. This includes:

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

ASSEMBLER INPUT
the contents of each text card in the entry.

MACROS USED
a list of the MAC entries with the GENASM subentry for the assembler module. This information is produced only if you specify the XREF option.

Figure 17 is an example of the output from LIST CDS ASSEM XREF.

```
SMPCDS   ASSEMBLER ENTRIES
  NAME

ASSEM1    LAST UPDATE      = GXY1000  TYPE=ADD
          ASSEMBLER INPUT  = THIS IS THE FIRST LINE OF ASSEMBLER INPUT.  THE ENTIRE LINE IS PRINTED.
                             THIS IS THE SECOND LINE OF ASSEMBLER INPUT.  THE ENTIRE MEMBER IS PRINTED.
          MACROS USED      = MACRO1    MACRO2

ASSEM2    LAST UPDATE      = HXY1010  TYPE=REP
          ASSEMBLER INPUT  = THIS IS THE FIRST LINE OF ASSEMBLER INPUT.  THE ENTIRE LINE IS PRINTED.
                             THIS IS THE SECOND LINE OF ASSEMBLER INPUT.  THE ENTIRE MEMBER IS PRINTED.
          MACROS USED      = MACRO1    MACRO3
```

Figure 17. LIST CDS ASSEM XREF

DLIB[(dlibname[,dlibname]...)]
specifies that information for all DLIB entries or the specified DLIB entries are to be listed. The information includes:

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

SYSTEM LIBRARY
the names of the target system libraries.

Figure 18 is an example of output from LIST CDS DLIB.

```
SMPCDS    DLIB ENTRIES
  NAME

DLIB01     LAST UPDATE       = GXY1000   TYPE=ADD
           SYSTEM LIBRARY    = SYSLIB01  SYSLIB02

DLIB02     LAST UPDATE       = HXY1010   TYPE=ADD
           SYSTEM LIBRARY    = SYSLIB01  SYSLIB02

DLIBMAC1   LAST UPDATE       = GXY1000   TYPE=ADD
           SYSTEM LIBRARY    = MACLIB01

DLIBMAC2   LAST UPDATE       = HXY1010   TYPE=ADD
           SYSTEM LIBRARY    = MACLIB01
```

Figure 18. LIST CDS DLIB

LMOD[(modname[,modname]...)]
specifies that information for all LMOD entries or the specified LMOD entries are to be listed. The information includes:

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

SYSTEM LIBRARY
the names of the target system libraries.

LKED ATTRIBUTES
the parameters used to link edit the load module.

LKED CONTROL
the linkage editor control cards for the load module.

MODULES
a list of the MOD entries with the LMOD subentry for the load module. This information is produced only when you specify the XREF option.

Figure 19 is an example of output from LIST CDS LMOD XREF.

```
SMPCDS    LOAD MODULE ENTRIES
  NAME

LMOD1      LAST UPDATE       = HXY1010   TYPE=REP
           SYSTEM LIBRARY    = SYSLIB01  SYSLIB02
           LKED ATTRIBUTES   = NCAL      RENT
           LKED CONTROL      = ENTRY MOD001
           MODULES           = MOD001    MOD002     ASSEM1

MOD003     LAST UPDATE       = HXY1010   TYPE=ADD
           SYSTEM LIBRARY    = SYSLIB01  SYSLIB02
           LKED ATTRIBUTES   = NCAL      RENT
           LKED CONTROL      = ENTRY MOD003
           MODULES           = MOD003    ASSEM2
```

Figure 19. LIST CDS LMOD XREF

MAC[(macname[,macname]...)]
specifies that information for all MAC entries or the specified MAC entries are to be listed. The information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the macro.

UMID
a list of the SYSMOD-IDs for the SYSMODs that updated the macro.

DISTLIB
the distribution library name.

SYSLIB
the target system library name.

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

GENASM
a list of the ASSEM and SRC entries that are reassembled when this macro is changed.

SYSMOD HISTORY
the SYSMOD-ID, type, and status for each SYSMOD that contains a ++MAC, ++MACUPD, or ++UPDTE modification control statement for the macro. This information is produced only when you specify the XREF option.

Figure 20 is an example of output from LIST CDS MAC XREF.

```
SMPCDS   MACRO ENTRIES
  NAME

MACRO1     LAST UPDATE       = HXY1010   TYPE=REP
           LIBRARIES         = DISTLIB=DLIBMAC2  SYSLIB=MACLIB01
           FMID              = HXY1010
           RMID              = HXY1010
           GENASM            = ASSEM1     ASSEM2    MOD001    MOD002
           SYSMOD HISTORY    = SYSMOD  TYPE       DATE   MCS        ---------- STATUS ----------
                               GXY1000 FUNCTION  77.301  MAC                APP      ACC
                               HXY1010 FUNCTION  77.355  MAC                APP

MACRO2     LAST UPDATE       = GXY1000   TYPE=ADD
           LIBRARIES         = DISTLIB=DLIBMAC1  SYSLIB=MACLIB01
           FMID              = GXY1000
           RMID              = GXY1000
           UMID              = UZ00014
           GENASM            = ASSEM1     MOD002
           SYSMOD HISTORY    = SYSMOD  TYPE       DATE   MCS        ---------- STATUS ----------
                               GXY1000 FUNCTION  77.301  MAC                APP      ACC
                               UZ00014 PTF       77.357  MACUPD             APP

MACRO3     LAST UPDATE       = HXY1010   TYPE=ADD
           LIBRARIES         = DISTLIB=DLIBMAC2  SYSLIB=MACLIB01
           FMID              = HXY1010
           RMID              = HXY1010
           GENASM            = ASSEM2     MOD003
           SYSMOD HISTORY    = SYSMOD  TYPE       DATE   MCS        ---------- STATUS ----------
                               HXY1010 FUNCTION  77.345  MAC                APP
```

Figure 20. LIST CDS MAC XREF

MOD[(modname[,modname]...)]
specifies that information for all MOD entries or the specified MOD entries are to be listed. The information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the module.

UMID
a list of the SYSMOD-IDs for the SYSMODs that updated the module.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

LMODS
a list of the load modules that include the module.

SYSMOD HISTORY
the SYSMOD-ID, type, and status for each SYSMOD that contains a ++MOD or ++ZAP modification control statement for the module. This information is produced only when you specify the XREF option.

Figure 21 is an example of output from LIST CDS MOD XREF.

```
SMPCDS    MODULE ENTRIES
 NAME

MOD001     LAST UPDATE      = UZ00010   TYPE=ADD
           LIBRARIES        = DISTLIB=DLIB01
           FMID             = HXY1010
           RMID             = XY10101
           LMODS            = LMOD1
           SYSMOD HISTORY   = SYSMOD   TYPE      DATE    MCS       ---------- STATUS ----------
                              GXY1000  FUNCTION  77.301  MOD            APP         ACC
                              UZ00010  PTF       77.312  MOD            APP
                              AZ00124  APAR      77.318  SRCUPD         APP
                              HXY1010  FUNCTION  77.345  MOD            APP
                              UZ00012  PTF       77.338  MOD            APP
                              XY10001  USERMOD   77.345  SRCUPD         APP
                              UZ00014  PTF       77.357  MOD            APP
                              UZ00015  PTF       77.357  MOD            APP
                              XY10101  USERMOD   77.357  SRCUPD         APP
MOD002     LAST UPDATE      = UCLIN     TYPE=ADD
           LIBRARIES        = DISTLIB=DLIB01
           FMID             = GXY1000
           RMID             = UZ00014
           LMODS            = LMOD1
           SYSMOD HISTORY   = SYSMOD   TYPE      DATE    MCS       ---------- STATUS ----------
                              GXY1000  FUNCTION  77.301  MOD            APP         ACC
                              UZ00010  PTF       77.312  MOD            APP
MOD003     LAST UPDATE      = HXY1010   TYPE=ADD
           LIBRARIES        = DISTLIB=DLIB02
           FMID             = HXY1010
           RMID             = HXY1010
           LMODS            = MOD003
           SYSMOD HISTORY   = SYSMOD   TYPE      DATE    MCS       ---------- STATUS ----------
                              HXY1010  FUNCTION  77.345  MOD            APP
```

Figure 21. LIST CDS MOD XREF

SRC[(srcname[,srcname]...)]
specifies that information for all SRC entries or the specified SRC entries are to be listed. The information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the source module.

UMID
a list of the SYSMOD-IDs for the SYSMOD that updated the source module.

DISTLIB
the distribution library name.

SYSLIB
the target system library name.

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

MACROS
: a list of the MAC entries with a GENASM subentry for the source module. This information is produced only when you specify the XREF option.

SYSMOD HISTORY
: the SYSMOD-ID, type, and status for each SYSMOD that contains a ++SRC or ++SRCUPD modification control statement for the source module. This information is produced only when you specify the XREF option.

Figure 22 is an example of output from LIST CDS SRC XREF.

```
SMPCDS   SOURCE ENTRIES
 NAME

MOD001    LAST UPDATE      = HXY1010  TYPE=REP
          LIBRARIES        = DISTLIB=DLIBSRC2
          FMID             = HXY1010
          RMID             = HXY1010
          UMID             = UZ00015   XY10101
          MACROS USED      = MACRO1
          SYSMOD HISTORY   = SYSMOD  TYPE       DATE    MCS        ---------- STATUS ----------
                             GXY1000 FUNCTION  77.301  SRC              APP       ACC
                             UZ00010 PTF       77.312  SRCUPD           APP
                             AZ00124 APAR      77.318  SRCUPD           APP
                             UZ00012 PTF       77.338  SRCUPD           APP
                             HXY1010 FUNCTION  77.345  SRC              APP
                             XY10001 USERMOD   77.345  SRCUPD           APP
                             UZ00014 PTF       77.357  SRCUPD           APP
                             UZ00015 PTF       77.357  SRCUPD           APP
                             XY10101 USERMOD   77.357  SRCUPD           APP
MOD002    LAST UPDATE      = GXY1000  TYPE=ADD
          LIBRARIES        = DISTLIB=DLIBSRC1
          FMID             = GXY1000
          RMID             = GXY1000
          UMID             = UZ00010
          MACROS USED      = MACRO1    MACRO2
          SYSMOD HISTORY   = SYSMOD  TYPE       DATE    MCS        ---------- STATUS ----------
                             GXY1000 FUNCTION  77.301  SRC              APP       ACC
                             UZ00010 PTF       77.312  SRCUPD           APP
MOD003    LAST UPDATE      = HXY1010  TYPE=ADD
          LIBRARIES        = DISTLIB=DLIBSRC2
          FMID             = HXY1010
          RMID             = HXY1010
          MACROS USED      = MACRO3
          SYSMOD HISTORY   = SYSMOD  TYPE       DATE    MCS        ---------- STATUS ----------
                             HXY1010 FUNCTION  77.345  SRC              APP
```

Figure 22. LIST CDS SRC XREF

SYSMOD[(sysmodid[,sysmodid]...)]
: specifies that information for all SYSMOD entries or the specified SYSMOD entries is to be listed. The information includes:

TYPE
: the type of SYSMOD ('APAR', 'FUNCTION', 'PTF', 'USERMOD', or 'SUPERSEDED').

FMID
: the SYSMOD-ID from the ++VER or ++FUNCTION modification control statement.

JCLIN
: an indicator that there is inline JCLIN within the SYSMOD.

IF MCS
: an indicator that there are ++IF modification control statements within the SYSMOD.

STATUS
: a status indicator that contains one of the following:

  'BYP' if the SYSMOD was applied using the BYPASS keyword
  'ERR' if the SYSMOD was not successfully applied or restored
  'RGN' if the SYSMOD entry was copied from the ACDS
  'RES' if an attempt was made to restore the SYSMOD
  'REC' if the SYSMOD is received
  'APP' if the SYSMOD is applied
  'ACC' if the SYSMOD is accepted.

DATE/TIME
: the date and time stamps for RECEIVE, APPLY, ACCEPT, UCLIN, and RESTORE processing for this SYSMOD.

LASTSUP
: the last SYSMOD processed that superseded this SYSMOD.

SREL, DELETE, NPRE, PRE, REQ, SUP, and VERSION
: the contents of the keyword lists from the ++VER modification control statement used during APPLY processing.

MAC, MACUPD, MOD, SRC, SRCUPD, SZAP, and XZAP
: the names from element modification control statements included in the SYSMOD.

RMAC, RMACUPD, RMOD, RSRC, RSRCUPD, RSZAP, and RXZAP
: the names from element modification control statements included in SYSMODs that represent regressed modifications. A regression occurs when a subsequent SYSMOD did not specify this SYSMOD in the PRE or SUP operand list of its ++VER modification control statement.

ASSEM
: the names of modules to be assembled as a result of macro or source changes contained in the SYSMOD.

SUPBY

a list of the SYSMODs that supersede this SYSMOD; that is, SUP was specified in their ++VER modification control statements.

DELBY

a SYSMOD that deletes this SYSMOD; that is, DELETE was specified in its ++VER modification control statement.

IFREQBY

a list of SYSMODs that specify this SYSMOD as a requisite SYSMOD (REQ) in a ++IF modification control statement. This information is produced only when you specify the XREF keyword.

NPREBY

a list of SYSMODs that specify this SYSMOD as a negative prerequisite (NPRE) in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

PREBY

a list of SYSMODs that specify this SYSMOD as a prerequisite SYSMOD (PRE) in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

REQBY

a list of SYSMODs that specify this SYSMOD as a requisite SYSMOD in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

VERSIONBY

a list of SYSMODs that specify this SYSMOD as a versioned SYSMOD in a ++VER modification control statement. This information is produced only when you specify the XREF keyword.

Figure 23 is an example of output from LIST CDS SYSMOD XREF.

```
SMPCDS    SYSMOD ENTRIES
 NAME

AZ00123   TYPE               = SUPERSEDED
          SUPBY              = HXY1010    UZ00010

AZ00124   TYPE               = APAR
          STATUS             = REC  APP
          FMID               = GXY1000
          DATE/TIME REC      = 77.318 14:28:54
                    APP      = 77.318 14:30:25
          SREL      VER(001) = Z038
          PRE       VER(001) = UZ00010
          SRCUPD             = MOD001
          SUPBY(IN SYSMOD)=    HXY1010    UZ00012

AZ00136   TYPE               = SUPERSEDED
          SUPBY              = UZ00014    UZ00015

GXY1000   TYPE               = FUNCTION
          STATUS             = REC  APP   ACC
          FMID               = GXY1000
          DATE/TIME REC      = 77.301 12:18:35
                    APP      = 77.301 12:20:43
                    ACC      = 77.314 16:22:45
          JCLIN              = YES
          SREL      VER(001) = Z038
          MAC                = MACRO1     MACRO2
          MOD                = MOD001     MOD002
          SRC                = MOD001     MOD002

HXY1010   TYPE               = FUNCTION
          STATUS             = REC  APP
          FMID               = GXY1000
          DATE/TIME REC      = 77.345 10:13:46
                    APP      = 77.345 10:15:27
          JCLIN              = YES
          SREL      VER(001) = Z038
          PRE       VER(001) = UZ00010
          SUP       VER(001) = AZ00123    AZ00124
          MAC                = MACRO1     MACRO3
          MOD                = MOD001     MOD003
          SRC                = MOD001     MOD003

UZ00010   TYPE               = PTF
          STATUS             = REC  APP
          FMID               = GXY1000
          DATE/TIME REC      = 77.312 09:43:12
                    APP      = 77.312 09:46:15
          SREL      VER(001) = Z038
          SUP       VER(001) = AZ00123
          MOD                = MOD001     MOD002
          SRCUPD             = MOD001     MOD002
          PREBY       (XREF) = HXY1010    UZ00012    UZ00014

UZ00012   TYPE               = PTF
          STATUS             = REC  APP
          FMID               = GXY1000
          DATE/TIME REC      = 77.338 13:32:32
                    APP      = 77.338 13:35:43
          SREL      VER(001) = Z038
          PRE       VER(001) = UZ00010
          SUP       VER(001) = AZ00124
          MOD                = MOD001
          SRCUPD             = MOD001
          SUPBY(IN SYSMOD)=    UZ00014

UZ00014   TYPE               = PTF
          STATUS             = REC  APP
          FMID               = GXY1000
          DATE/TIME REC      = 77.357 15:12:56
                    APP      = 77.357 15:15:21
          SREL      VER(001) = Z038
          PRE       VER(001) = UZ00010
          SUP       VER(001) = AZ00136    UZ00012
          MACUPD             = MACRO2
          MOD                = MOD001
          SRCUPD             = MOD001
          REQBY       (XREF) = UZ00015

UZ00015   TYPE               = PTF
          STATUS             = REC  APP
          FMID               = HXY1010
          DATE/TIME REC      = 77.357 15:13:14
                    APP      = 77.357 15:15:21
          SREL      VER(001) = Z038
          REQ       VER(001) = UZ00014
          SUP       VER(001) = AZ00136
          MOD                = MOD001
          SRCUPD             = MOD001
          IFREQBY     (XREF) = UZ00014

XY10001   TYPE               = USERMOD
          STATUS             = REC  APP
          FMID               = GXY1000
          DATE/TIME REC      = 77.345 10:13:51
                    APP      = 77.345 10:15:37
          SREL      VER(001) = Z038
          SRCUPD             = MOD001

XY10101   TYPE               = USERMOD
          STATUS             = REC  APP
          FMID               = HXY1010
          DATE/TIME REC      = 77.345 10:14:02
                    APP      = 77.357 15:15:21
          SREL      VER(001) = Z038
          SRCUPD             = MOD001
          IFREQBY     (XREF) = XY10001
```

Figure 23. LIST CDS SYSMOD XREF

You can restrict the selection of SYSMOD entries to be printed by coding SYSMOD, followed by one or more of the following operands. For example, if you specify 'LIST CDS SYSMOD ERROR.', SMP lists all of the SYSMOD entries in the CDS that have the ERROR indicator set on.

If you specify more than one operand, SMP combines the operands into one logical request. For example, if you specify 'LIST CDS SYSMOD APAR PTF ERROR SUP.', SMP lists all of the APAR and PTF entries that have the ERROR indicator set on and that are superseded. Specifying both SUP and NOSUP at the same time causes a syntax error.

APAR
: specifies that APAR SYSMODs are to be listed.

DELETE
: specifies that function SYSMOD entries that have been deleted from the CDS by other function SYSMODs are to be listed. This operand can be abbreviated as 'DEL'.

ERROR
: specifies that SYSMODs that have the ERROR indicator set are to be listed. This operand can be abbreviated as 'ERR'.

FUNCTION
: specifies that all function SYSMODs are to be listed. This operand can be abbreviated as 'FUNC'.

NOACCEPT
: specifies that SYSMODs that have been received and applied, but not accepted in the ACDS are to be listed. Both the CDS and the ACDS data sets must be available when NOACCEPT is coded. A SYSMOD is considered accepted if the SYSMOD entry exists on the ACDS with the ERROR status indicator set off. This operand can be abbreviated as 'NOACC'.

NOSUP
: specifies that only SYSMODs that have not been superseded are to be listed.

  Note: This operand is mutually exclusive with the SUP operand. Specification of both causes a syntax error.

PTF
: specifies that all PTF SYSMODs are to be listed.

RESTORE
: specifies that SYSMODs that have the RESTORE

indicator set are to be listed. The ERROR indicator must also be on for this condition to be valid. This operand can be abbreviated as 'RES'.

SUP
specifies that only superseded SYSMODs are to be listed.

Note: This operand is mutually exclusive with the NOSUP operand. Specification of both is causes a syntax error.

USERMOD
specifies that all USERMOD SYSMODs are to be listed. This operand can be abbreviated as 'USER'.

SYS
specifies that system information, such as the default NUCID, system type and release, the SAVESTS and SAVEMTS indicators, and the identifier of the CDS, is to be listed.

Figure 24 is an example of output from LIST CDS SYS.

```
SMPCDS   SYSTEM ENTRY
 NAME

SYSTEM   OPTIONS        = CDSID=CDS1     SREL=Z038  NUCID=8  PEMAX=9999  SAVEMTS=YES  SAVESTS=NO
```

Figure 24. LIST CDS SYS

LIST CDS Exception Reports

There are two possible exception reports from LIST CDS processing; the LIST MASS SUMMARY REPORT FOR SMPCDS and the LIST SELECT SUMMARY REPORT FOR SMPCDS. The reports are produced at the end of your LIST output if any of the following exceptions are found:

- If you list the CDS without specifying any other operands, and there are no ASSEM, DLIB, LMOD, MAC, MOD, SRC, or SYSMOD entries, then 'xxxx ENTRIES NOT FOUND' where 'xxxx' is ASSEM, DLIB, LMOD, MAC, MOD, SRC, or SYSMOD appears in the LIST MASS SUMMARY REPORT FOR SMPCDS. See Figure 31 for an example of output from LIST MASS SUMMARY REPORT FOR SMPPTS which contains the same type of information as output from LIST MASS SUMMARY REPORT FOR SMPCDS.

- If you list the CDS and specify the ASSEM, DLIB, LMOD, MAC, MOD, SRC, or SYSMOD operand but do not qualify the operand with a particular entry name, and no ASSEM, DLIB, LMOD, MAC, MOD, SRC or SYSMOD entries are found, then 'xxxx ENTRIES NOT FOUND' where 'xxxx' is ASSEM, DLIB, LMOD, MAC, MOD, SRC, or SYSMOD appears in the LIST MASS SUMMARY REPORT FOR SMPCDS.

- If you list the CDS and specify the ASSEM, DLIB, LMOD, MAC, MOD, SRC, or SYSMOD operand and qualify the operand with a particular entry name, but that entry name is not found, then 'xxx yyyyyyyy' where 'xxx' is ASSEM, DLIB, LMOD, MAC, MOD, SRC or SYSMOD and 'yyyyyyyy' is the entry name appears in the LIST SELECT SUMMARY REPORT FOR SMPCDS.

- If you list all the CDS SYSMOD entries, qualified by the APAR, DELETE, ERROR, FUNCTION, NOACCEPT, NOSUP, PTF, SUP, RESTORE, or USERMOD keywords, and that type of entry is not found, then 'SYSMOD ENTRIES NOT FOUND' appears in the LIST MASS SUMMARY REPORT FOR SMPCDS.

- If you list specific ACDS SYSMOD entries, qualified by the APAR, DELETE, ERROR, FUNCTION, NOACCEPT, NOSUP, PTF, SUP, RESTORE, or USERMOD keywords and that specific entry is not found, then 'SYSMOD xxxxxxxx' where 'xxxxxxxx' is the SYSMOD-ID appears in the LIST SELECT SUMMARY REPORT FOR SMPCDS as not found. In addition, the entries that you selected can be considered ineligible for the following reasons:

keyword	exception
APAR	the SYSMOD is not an APAR
DELETE	the DELETE indicator is not set
ERROR	the ERROR indicator is not set
FUNCTION	the SYSMOD is not a function
NOACCEPT	the SYSMOD exists in the ACDS
NOSUP	the SYSMOD is superseded
PTF	the SYSMOD is not a PTF
RESTORE	the RESTORE indicator is not set
SUP	the SYSMOD is not superseded
USERMOD	the SYSMOD is not a user modification

LIST CRQ Syntax

```
LIST CRQ

     [SYSMOD[(sysmodid[,sysmodid]...)]]

     [FMID[(sysmodid[,sysmodid]...)]]

     .
```

LIST CRQ Operands

CRQ
specifies that all or selected information from the CRQ is to be listed. If no other operands are specified, all SYSMOD entries and all FMID entries (without requisite SYSMOD information) are to be listed.

Figure 25 is an example of output from LIST CRQ.

```
SMPCRQ    FMID ENTRIES
  FMID

HXY1010    SYSMODS = UZ00014    XY10001
```

Figure 25. LIST CRQ

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that all SYSMOD entries or selected SYSMOD entries from the CRQ are to be listed. The information listed includes, for each SYSMOD entry, the SYSMOD-ID specified in the FMID operand and the requisite SYSMOD-IDs from the ++IF modification control statements.

Figure 26 is an example of output from LIST CRQ SYSMOD.

```
SMPCRQ    SYSMOD ENTRIES
  SYSMOD    FMID

UZ00014    HXY1010    IFREQ = UZ00015
XY10001    HXY1010    IFREQ = XY10101
```

Figure 26. LIST CRQ SYSMOD

FMID[(sysmodid[,sysmodid]...)]
specifies that all FMID entries or selected FMID entries on the CRQ are to be listed with the requisite SYSMOD

information from the corresponding SYSMOD entries. The information listed includes the SYSMOD-ID specified in the FMID operand, the IDs of the SYSMODs that reference the SYSMOD-ID specified in the FMID operand, and the IDs of the SYSMODs that must be processed by APPLY when the function SYSMOD specified as the FMID is applied.

Figure 27 is an example of output from LIST CRQ FMID.

```
SMPCRQ   FMID/SYSMOD ENTRIES
 FMID     SYSMOD

HXY1010   UZ00014   IFREQ = UZ00015
          XY10001   IFREQ = XY10101
```

Figure 27. LIST CRQ FMID (HXY1010)

LIST LOG Syntax

```
LIST LOG [(from-date, to-date)]
```

•

LIST LOG Operands

LOG
: specifies that the contents of the LOG data set are to be listed.

(from-date,to-date)
: specifies the range of dates (from the from-date to the to-date) for which the data set is to be listed. The dates are specified as mm dd yy, where mm is the month (01-12), dd is the day (01-31) and yy is the year (00-99).

  If this operand is not specified, the contents of the entire LOG data set are listed.

LIST PTS Syntax

```
LIST PTS

     [MCS[(sysmodid[,sysmodid]...)]

     [SYSMOD[(sysmodid[,sysmodid]...)]
        [APAR]    [FUNCTION]   [NOACCEPT]   [NOAPPLY]   [PTF]
        [USERMOD]]

     [SYS]

     •
```

LIST PTS Operands

MCS[(sysmodid[,sysmodid]...)]
: specifies that the modification control statements for all MCS entries or the specified MCS entries are to be listed, including all comments.

  Figure 28 is an example of output from LIST PTS MCS.

```
SMPPTS    M.C.S. ENTRIES
  NAME

AZ00124   M.C.S. ENTRIES  = ++ APAR(AZ00124).
                            ++ VER(Z038) FMID(GXY1000) PRE(UZ00010).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
GXY1000   M.C.S. ENTRIES  = ++ FUNCTION(GXY1000) FILES(4).
                            ++ VER(Z038).
                            ++ JCLIN RELFILE(1).
                            ++ MAC(MACRO1) DISTLIB(DLIBMAC1) RELFILE(2).
                            ++ MAC(MACRO2) DISTLIB(DLIBMAC1) RELFILE(2).
                            ++ MOD(MOD001) DISTLIB(DLIB01) RELFILE(3).
                            ++ MOD(MOD002) DISTLIB(DLIB01) RELFILE(3).
                            ++ SRC(MOD001) DISTLIB(DLIBSRC1) RELFILE(3).
                            ++ SRC(MOD002) DISTLIB(DLIBSRC1) RELFILE(3).
HXY1010   M.C.S. ENTRIES  = ++ FUNCTION(HXY1010) FILES(4).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00123,AZ00124) PRE(UZ00010).
                            ++ JCLIN RELFILE(1).
                            ++ MAC(MACRO1) DISTLIB(DLIBMAC2) RELFILE(2).
                            ++ MAC(MACRO3) DISTLIB(DLIBMAC2) RELFILE(2).
                            ++ MOD(MOD001) DISTLIB(DLIB02) RELFILE(3).
                            ++ MOD(MOD003) DISTLIB(DLIB02) RELFILE(3).
                            ++ SRC(MOD001) DISTLIB(DLIBSRC2) RELFILE(3).
                            ++ SRC(MOD003) DISTLIB(DLIBSRC2) RELFILE(3).
UZ00010   M.C.S. ENTRIES  = ++ PTF(UZ00010).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00123).
                            ++ MOD(MOD001) DISTLIB(DLIB01).
                            ++ MOD(MOD002) DISTLIB(DLIB01).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
                            ++ SRCUPD(MOD002) DISTLIB(DLIBSRC1).
UZ00012   M.C.S. ENTRIES  = ++ PTF(UZ00012).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00124) PRE(UZ00010).
                            ++ MOD(MOD001) DISTLIB(DLIB01).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
UZ00014   M.C.S. ENTRIES  = ++ PTF(UZ00014).
                            ++ VER(Z038) FMID(GXY1000) SUP(AZ00136,UZ00012) PRE(UZ00010).
                            ++ IF FMID(HXY1010) THEN REQ(UZ00015).
                            ++ MOD(MOD001) DISTLIB(DLIB01).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
                            ++ MACUPD(MACRO2) DISTLIB(DLIBMAC1).
UZ00015   M.C.S. ENTRIES  = ++ PTF(UZ00015).
                            ++ VER(Z038) FMID(HXY1010) SUP(AZ00136) REQ(UZ00014).
                            ++ MOD(MOD001) DISTLIB(DLIB02).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC2).
XY10001   M.C.S. ENTRIES  = ++ USERMOD(XY10001).
                            ++ VER(Z038) FMID(GXY1000).
                            ++ IF FMID(HXY1010) THEN REQ(XY10101).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC1).
XY10101   M.C.S. ENTRIES  = ++ USERMOD(XY10101).
                            ++ VER(Z038) FMID(HXY1010).
                            ++ SRCUPD(MOD001) DISTLIB(DLIBSRC2).
```

Figure 28. LIST PTS MCS

Note: If both the MCS and SYSMOD entries for a selected set of SYSMODs are to be listed, the same SYSMOD-IDs must be specified in both the MCS and SYSMOD operands.

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that information for all or the specified SYSMOD entries are to be listed. This includes:

TYPE
the type of SYSMOD ('APAR', 'FUNCTION', 'PTF', or 'USERMOD').

DATE/TIME REC
the date and time stamp indicating when the SYSMOD was received.

STATUS
a status indicator that contains:

'BYP' if the SYSMOD was received using the BYPASS option
'APP' if the SYSMOD is applied to any target system
'ACC' if the SYSMOD is accepted into any DLIB
'ERR' if the SYSMOD was not completely received due to an error loading relative files.

APPLY CDSID
a list of the identifiers of any CDS on which the SYSMOD is applied.

ACCEPT ACDSID
a list of the identifiers of any ACDS on which the SYSMOD is accepted.

JCLIN
an indicator that there is inline JCLIN within the SYSMOD.

DSPREFIX
the user-specified high level data set name qualifier for files on SMPTLIB volumes for this SYSMOD.

SREL, DELETE, FMID, PRE, NPRE, REQ, SUP, and VERSION
the contents of the keyword lists from each processable ++VER modification control statement within the SYSMOD.

MAC, MACUPD, MOD, SRC, SRCUPD, and ZAP
the names from element modification control statements included in the SYSMOD.

Figure 29 is an example of output from LIST PTS SYSMOD.

```
SMPPTS     SYSMOD ENTRIES
  NAME

AZ00124    TYPE                = APAR
           STATUS              = APP
           DATE/TIME REC       = 77.318 14:28:54
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = GXY1000
           PRE       VER(001)  = UZ00010
           SRCUPD              = MOD001

GXY1000    TYPE                = FUNCTION
           STATUS              = APP   ACC
           DATE/TIME REC       = 77.301 12:18:35
           APPLY CDSID         = CDS1
           ACCEPT ACDSID       = ACDS1
           JCLIN               = YES
           DSPREFIX            = ZZ10
           SREL      VER(001)  = Z038
           MAC                 = MACRO1     MACRO2
           MOD                 = MOD001     MOD002
           SRC                 = MOD001     MOD002

HXY1010    TYPE                = FUNCTION
           STATUS              = APP
           DATE/TIME REC       = 77.345 10:13:46
           APPLY CDSID         = CDS1
           JCLIN               = YES
           DSPREFIX            = ZZ10
           SREL      VER(001)  = Z038
           FMID      VER(001)  = GXY1000
           PRE       VER(001)  = UZ00010
           SUP       VER(001)  = AZ00123    AZ00124
           MAC                 = MACRO1     MACRO3
           MOD                 = MOD001     MOD003
           SRC                 = MOD001     MOD003

UZ00010    TYPE                = PTF
           STATUS              = APP
           DATE/TIME REC       = 77.312 09:43:12
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = GXY1000
           SUP       VER(001)  = AZ00123
           MOD                 = MOD001     MOD002
           SRCUPD              = MOD001     MOD002

UZ00012    TYPE                = PTF
           STATUS              = APP
           DATE/TIME REC       = 77.338 13:32:32
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = GXY1000
           PRE       VER(001)  = UZ00010
           SUP       VER(001)  = AZ00124
           MOD                 = MOD001
           SRCUPD              = MOD001

UZ00014    TYPE                = PTF
           STATUS              = APP
           DATE/TIME REC       = 77.357 15:12:56
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = GXY1000
           PRE       VER(001)  = UZ00010
           SUP       VER(001)  = AZ00136    UZ00012
           MACUPD              = MACRO2
           MOD                 = MOD001
           SRCUPD              = MOD001

UZ00015    TYPE                = PTF
           STATUS              = APP
           DATE/TIME REC       = 77.357 15:13:14
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = HXY1010
           REQ       VER(001)  = UZ00014
           SUP       VER(001)  = AZ00136
           MOD                 = MOD001
           SRCUPD              = MOD001

XY10001    TYPE                = USERMOD
           STATUS              = APP
           DATE/TIME REC       = 77.345 10:13:51
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = GXY1000
           SRCUPD              = MOD001

XY10101    TYPE                = USERMOD
           STATUS              = APP
           DATE/TIME REC       = 77.345 10:14:02
           APPLY CDSID         = CDS1
           SREL      VER(001)  = Z038
           FMID      VER(001)  = HXY1010
           SRCUPD              = MOD001
```

Figure 29. LIST PTS SYSMOD

You can restrict the selection of SYSMOD entries to be listed by specifying the SYSMOD keyword followed by one or more of the following operands. If you specify 'LIST PTS SYSMOD NOAPPLY.', SMP lists all of the SYSMODs that have been received but not applied.

However, if you specify more than one operand, SMP combines the operands into one logical request. For example, if you specify 'LIST PTS SYSMOD APAR PTF NOAPPLY NOACCEPT.', SMP lists all APARs and PTFs that have been received but not applied or accepted.

APAR
: specifies that APAR SYSMODs are to be listed.

FUNCTION
: specifies that all function SYSMODs are to be listed. This operand can be abbreviated as 'FUNC'.

NOACCEPT
: specifies that SYSMODs that have been received but not accepted in the ACDS are to be listed. The ACDS data set must be available when NOACCEPT is coded. A SYSMOD is considered accepted when the SYSMOD entry exists on the ACDS with the ERROR status indicator set off. This operand can be abbreviated as 'NOACC'.

NOAPPLY
: specifies that SYSMODs that have been received but not applied in the CDS are to be listed. The CDS data set must be available when NOAPPLY is coded. A SYSMOD is considered applied when the SYSMOD entry exists on the CDS with the ERROR status indicator set off. This operand can be abbreviated as 'NOAPP'.

PTF
: specifies that all PTF SYSMODs are to be listed.

USERMOD
: specifies that all USERMOD SYSMODs are to be listed. This operand can be abbreviated as 'USER'.

SYS
: specifies that system information, such as the system releases and function modification IDs that pertain to the target system, space parameters for allocation of storage by SMP, data set prefix and SMP processing options, such as the PURGE and REJECT indicators, and assembler, linkage editor, compress, copy, update, IOSUP, and IMASPZAP programs, parameters and defaults, is to be listed.

If a subentry of the SYSTEM entry was never created using UCLIN, the subentry appears in the LIST output as the characters 'NULL'.

Figure 30 is an example of output from LIST PTS SYS.

```
SMPPTS   SYSTEM ENTRY
  NAME

SYSTEM     OPTIONS           = PAGELEN=60  PEMAX=9999  PURGE=YES  REJECT=NO
           DSSPACE           = (1,20,10)
           DSPREFIX          = ZZ10
           ASM    NAME       = ASMBLRQ
                  SYSPRINT   = ASMPRINT
                  RC         = 4
           LKED   NAME       = IEWLQ
                  SYSPRINT   = LKDPRINT
                  PARM       = 'DECK,XREF,LET'
           UPDAT  SYSPRINT   = UPDPRINT
           SREL              = Z038
           FMID              = GXY1000    HXY1010
```

Figure 30. LIST PTS SYS

LIST PTS Exception Reports

There are two possible exception reports from LIST PTS processing; the LIST MASS SUMMARY REPORT FOR SMPPTS and the LIST SELECT SUMMARY REPORT FOR SMPPTS. The reports are produced at the end of your LIST output if any of the following exceptions are found:

- If you list the PTS without specifying any other operands, and there are no MCS or SYSMOD entries, then 'xxxx ENTRIES NOT FOUND' where 'xxxx' is MCS or SYSMOD appears in the LIST MASS SUMMARY REPORT FOR SMPPTS.

  Figure 31 is an example of output from LIST MASS SUMMARY REPORT FOR SMPPTS.

```
LIST MASS SUMMARY REPORT FOR SMPPTS

SYSMOD   ENTRIES NOT FOUND
MCS      ENTRIES NOT FOUND
```

Figure 31. LIST MASS SUMMARY REPORT FOR SMPPTS

- If you list the PTS and specify the MCS or SYSMOD operand but do not qualify the operand with a particular entry name, and no MCS or SYSMOD entries are found, then 'xxxx ENTRIES NOT FOUND' where 'xxxx' is MCS or SYSMOD appears in the LIST MASS SUMMARY REPORT FOR SMPPTS.

- If you list the PTS, specify the MCS or SYSMOD operand and qualify the operand with a particular entry name, but that entry name is not found, then 'xxx yyyyyyyy' where 'xxx' is MCS or SYSMOD and 'yyyyyyyy' is the entry name appears in the LIST SELECT SUMMARY REPORT FOR SMPPTS.

- If you list all the PTS SYSMOD entries, qualified by the APAR, FUNCTION, NOACCEPT, NOAPPLY, PTF, or USERMOD keywords, and that type of entry is not found, then 'SYSMOD ENTRIES NOT FOUND' appears in the LIST MASS SUMMARY REPORT FOR SMPPTS.

- If you list specific PTS SYSMOD entries, qualified by the APAR, FUNCTION, NOACCEPT, NOAPPLY, PTF, or USERMOD keywords, and that specific entry is not found, then 'SYSMOD xxxxxxxx' where 'xxxxxxxx' is the SYSMOD-ID appears in the LIST SELECT SUMMARY REPORT FOR SMPPTS as not found. In addition, the entries that you selected can be considered ineligible for the following reasons:

keyword	exception
APAR	the SYSMOD is not an APAR
FUNCTION	the SYSMOD is not a function
NOACCEPT	the SYSMOD exists in the ACDS
NOAPPLY	the SYSMOD exists in the CDS
PTF	the SYSMOD is not a PTF
USERMOD	the SYSMOD is not a user modification

LIST SCDS Syntax

```
LIST SCDS

     [SYSMOD[(sysmodid[,sysmodid]...)]]

     .
```

LIST SCDS Operands

SCDS
  specifies that all entries or selected entries on the SCDS for SYSMODs that caused BACKUP entries to be created are to be listed. The information listed for each SYSMOD entry consists of the back-up versions of the ASSEM, DLIB, LMOD, MAC, MOD, or SRC entries that were changed by JCLIN, the MAC, MOD/LMOD, and SRC entries that were deleted by the DELETE keyword on the associated modification control statement, and the MOD entries that were modified by the LMOD operand on the ++MOD modification control statement.

SYSMOD[(sysmodid[,sysmodid]...)]
  specifies one or more SYSMODs whose BACKUP entries are to be listed. If this operand is not specified, all BACKUP entries on the SCDS are listed. If a SYSMOD is specified for which there is no BACKUP entry on the SCDS, it is listed in the LIST SELECT SUMMARY REPORT FOR SMPSCDS. For each SYSMOD with backup entries, the following information is included when appropriate:

  DATE/TIME APP
    the date and time stamps indicating when APPLY processing was performed for the SYSMOD.

  ASSEM (ADD)
    a list of any ASSEM entries created by inline JCLIN.

  LMOD (ADD)
    a list of any LMOD entries created by inline JCLIN.

  MAC (ADD)
    a list of any MAC entries created by inline JCLIN.

  MOD (ADD)
    a list of any MOD entries created by inline JCLIN.

SRC (ADD)
: a list of any SRC entries created by inline JCLIN.

DLIB (ADD)
: a list of any DLIB entries created by inline JCLIN.

ASSEM (UPDATE)
: a list of any ASSEM entries created by inline JCLIN.

LMOD (UPDATE)
: a list of any LMOD entries created by inline JCLIN.

MAC (UPDATE)
: a list of any MAC entries updated by inline JCLIN.

MOD (UPDATE)
: a list of any MOD entries updated by inline JCLIN or the LMOD operand on ++MOD modification control statements.

SRC (UPDATE)
: a list of any SRC entries updated by inline JCLIN.

DLIB (UPDATE)
: a list of any DLIB entries updated by inline JCLIN.

LMOD (DEL)
: a list of any LMOD entries deleted by the DELETE operand on a ++MOD modification control statement.

MAC (DEL)
: a list of any MAC entries deleted by the DELETE operand on a ++MAC modification control statement.

MOD (DEL)
: a list of any MOD entries deleted by the DELETE operand on a ++MOD modification control statement.

SRC (DEL)
: a list of any SRC entries deleted by the DELETE operand on a ++SRC modification control statement.

The BACKUP entry for each entry listed in an UPDATE or

DEL list is formatted as described earlier in this chapter under "LIST CDS Operands".

Figure 32 is an example of output from LIST SCDS.

```
SMPSCDS  BACKUP ENTRIES FOR HXY1010
  NAME

HXY1010   DATE/TIME APP    = 77.345 10:15:27
          LMOD   (ADD)     = MOD003
          DLIB   (ADD)     = DLIB02     DLIBMAC2
          ASSEM  (UPDATE)  = ASSEM2

ASSEM2    TYPE             = ASSEM
          LAST UPDATE      = GXY1000  TYPE=ADD
          ASSEMBLER INPUT  = THIS IS THE FIRST LINE OF ASSEMBLER INPUT FROM GXY1000
                             THIS IS THE 2ND   LINE OF ASSEMBLER INPUT FROM GXY1000
                             THIS IS THE THIRD LINE OF ASSEMBLER INPUT FROM GXY1000
```

Figure 32. LIST SCDS SYSMOD (HXY1010)

LIST DDnames

SMPACDS	(required if the ACDS operand is specified)
SMPACRQ	(required if the ACRQ operand is specified)
SMPCDS	(required if the CDS operand is specified)
SMPCNTL	(required)
SMPCRQ	(required if the CRQ operand is specified)
SMPLIST	(required if the LIST output is to be separate from SMPOUT)
SMPLOG	(required)
SMPOUT	(required)
SMPPTS	(required if the PTS operand is specified)
SMPSCDS	(required if the SCDS operand is specified)

LIST Programming Considerations

1) When you specify a set of entries to be listed for a data set and specify a list for any entry type, then all other entry types specified must also have a list. Otherwise, a syntax error occurs.

   For example, a syntax error occurs if you code:

   ```
   LIST CDS MAC(MAC001,MAC002) MOD.
   ```

because a list of modules is not specified with the MOD operand.

2) Coding the LIST control statement without any operands causes a syntax error.

3) Coding the LIST control statement with data set options but without the data set operand causes a syntax error. For example, 'LIST SYSMOD.' is in error because the data set to be listed is not specified.

4) Bit indicators in SYSTEM entries are listed as YES if the indicator is set and NO if it is not set. Bit indicators in the SYSMOD entries that are defined as status indicators appear in the listing with their associated abbreviation.

LIST Return Codes

00 LIST processing completed successfully and without errors.

04 LIST processing completed, but at least one requested item was not listed. The possible error conditions are:

1) An entry specified in the LIST control statement was not found on the data set being listed.

2) An entry specified in the LIST control statement was found but was not eligible, as requested. For example, the SYSMOD-ID requested in 'LIST SYSMOD(UZ00004) FUNCTION.' was found, but it was not a function SYSMOD.

3) PEMAX was too small to process a selected entry.

4) A DD statement was missing.

08 LIST processing terminated. The error condition is:

1) The LIST control statement was specified without any accompanying keywords.

12 LIST processing terminated. The possible error conditions are:

1) A syntax error occurred in the LIST control statement.

2) Not enough storage was available.

3) An invalid date range was specified in the LIST LOG control statement.

4) A DD statement was missing.

16 A severe error was encountered, and SMP processing terminated.

LIST Error Recovery

If an out-of-space condition occurs on SMPLOG during LIST LOG processing, see "Resolving Direct Access Storage Problems" in Chapter 5 for information on how to handle this problem and then rerun LIST.

The LOG Control Statement

The SMP LOG control statement writes user-specified messages to the LOG data set. Messages written to the LOG data set cannot exceed 250 characters. Any number of LOG statements can be included in an SMP job step.

LOG Syntax

```
LOG  (message)
      [RC (function=code[,function=code]...)]
```

•

LOG Operands

(message)
: specifies the text of the message. The entire message text must be enclosed in parentheses, and the length of the message cannot exceed 250 characters. If your message is longer than 250 characters, issue multiple LOG control statements.

  Any character can be specified in the message text. If parentheses are to be specified as part of the message text, make sure that they are not nested; that is, make sure that each left parenthesis specified as part of the message text is followed by a right parenthesis before another left parenthesis is specified.

RC(function=code[,function=code]...)
: specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the LOG statement, or a syntax error results.

  Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, LOG processing is bypassed and LOG terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.

- If all specified SMP functions return codes less than or equal to their indicated codes, LOG is executed.

- Previous processing by any SMP function not specified on the RC operand has no effect on the current LOG processing.

LOG DDnames

SMPCNTL (required)
SMPLOG (required)
SMPOUT (required)

LOG Programming Considerations

The LOG data set be a sequential data set. The data set should be defined in the DD statement as a data set that can be modified; for example:

```
//SMPLOG DD  DSN=SYS1.SMPLOG,DISP=MOD
```

LOG Return Codes

00 LOG processing completed successfully and without errors.

04 Unused

08 Unused

12 LOG processing terminated. The possible error conditions are:

1) A syntax error occurred in the LOG control statement.

2) A DD statement was missing.

3) The LOG control statement was not processed because a previous control statement returned a non acceptable return code.

16 An I/O error was encountered and processing was terminated.

The RECEIVE Control Statement

The RECEIVE control statement initiates SMP processing of a system modification (SYSMOD). Any number of RECEIVE statements can be coded in an SMP job step.

RECEIVE Syntax

```
RECEIVE [{SELECT | EXCLUDE} (sysmodid[,sysmodid]...)]
        [BYPASS(FMID)]
        [RC(function=code[,function=code]...)]
        •
```

RECEIVE Operands

SELECT(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be processed from the SMPPTFIN data set. This operand can also be specified as 'S'.

EXCLUDE(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be excluded from the processing of the SMPPTFIN data set. This operand can also be specified as 'E'.

Note: If neither of the above operands is specified, all SYSMODs in the SMPPTFIN data set are processed.

BYPASS(FMID)
: specifies that the function modification identifier (FMID) check is to be bypassed during the processing of the SYSMODs.

RC(function=code[,function=code]...)
: specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the RECEIVE statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, RECEIVE processing is bypassed and RECEIVE terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.

- If all specified SMP functions return codes less than or equal to their indicated codes, RECEIVE is executed.

- Previous processing by any SMP function not specified on the RC operand has no effect on the current RECEIVE processing.

RECEIVE DDnames

SMPCNTL	(required)
SMPLOG	(required)
SMPOUT	(required)
SMPPTFIN	(required)
SMPPTS	(required)
SMPRPT	(optional)
SMPTLIB	(required if the SMPPTFIN tape has relative files)
SYSPRINT	(required if the SMPPTFIN tape has relative files)
SYSUT1	(required)
SYSUT2	(required)
SYSUT3	(required)

RECEIVE Programming Considerations

1) RECEIVE processing causes space to be used on the SMPPTS; therefore, the SMPPTS data set should have a secondary allocation and be blocked for maximum efficiency. There is no restriction as to the maximum blocksize.

2) Use the messages issued by the RECEIVE function on the SMPOUT data set for the processing status of each SYSMOD. Use the LIST PTS function to report the complete set of SYSMODs received in the SMPPTS data set.

3) The messages "RECEIVE PROCESSING TERMINATED" and "RECEIVE PROCESSING COMPLETED" do not imply that every SYSMOD in the SMPPTFIN data set has been processed by RECEIVE.

4) When RECEIVE processing detects a syntax error on a modification control statement, processing of the SYSMOD terminates; however, syntax checking continues and all subsequent modification control statements are listed.

5) SYSMODs are received regardless of the status of any requisite, negative prerequisite or prerequisite SYSMODs.

6) A SYSTEM entry is required in the SMPPTS data set to determine if any SYSMODs are eligible to be received. The SYSTEM entry must have at least one system release (SREL). The SREL and FMID subentries of the SYSTEM entry are used for comparison with the SREL and FMID operands of the ++VER modification control statement to determine if a SYSMOD should be received. If no FMID operand is present on a ++VER modification control statement in the SYSMOD, the SYSMOD is received if the SREL check is positive and the header modification control statement is ++FUNCTION. The BYPASS operand can be specified to bypass the FMID check. Every SYSMOD must have at least one ++VER modification control statement.

RECEIVE Return Codes

00 RECEIVE processing completed successfully and without errors.

04 RECEIVE processing completed, but there are possible error or warning messages because:

1) The HMASMEXT user written exit routine took some action for at least one SYSMOD.

08 RECEIVE processing completed, but errors were encountered and processing terminated for at least one SYSMOD. The possible error conditions are:

1) A selected SYSMOD was not found on the SMPPTFIN data set and was not processed.

2) PEMAX was too small to process a SYSMOD.

3) A GETMAIN failure occurred while processing a SYSMOD resulting in the termination of processing for that SYSMOD.

4) The modification name specified on the ++MACUPD, ++SRCUPD, or ++UPDTE modification control statement was different from the modification name specified on the IEBUPDTE "./ CHANGE" control statement resulting in the termination of processing for the affected SYSMOD.

5) RECEIVE processing detected a syntax error in a SYSMOD while scanning the modification control statements for a SYSMOD in the SMPPTFIN data set resulting in the termination of processing for that SYSMOD. Syntax errors include validation check errors.

6) RECEIVE processing encountered an end-of-file on the SMPPTFIN data set during the processing of a SYSMOD resulting in the termination of processing for that SYSMOD.

7) A return code from the HMASMEXT user written exit routine required RECEIVE to stop processing a SYSMOD.

8) Two modification control statements within a SYSMOD referred to the same element resulting in the termination of processing for that SYSMOD.

12 RECEIVE processing terminated. The possible error conditions are:

1) A GETMAIN failure occurred that caused the termination of RECEIVE processing.

2) A return code from the HMASMEXT user written exit routine required RECEIVE processing to terminate.

3) A syntax error was detected in the RECEIVE control statement.

4) A failure occurred during STOW processing while attempting to place a SYSMOD or MCS entry on the PTS.

5) None of the SYSMODs specified in the SELECT operand list were found or no SYSMODs were found in the SMPPTFIN data set.

6) A DD statement was missing.

7) The RECEIVE control statement was not processed because a previous control statement returned a non acceptable return code.

16 A severe error was encountered and SMP processing was terminated. The possible error conditions are:

1) An I/O error occurred.

2) A return code from the HMASMEXT user written exit routine required the termination of all processing.

RECEIVE Error Recovery

If RECEIVE issued the message 'HMA344 SYSMOD nnnn SUCCESSFULLY RECEIVED', the SYSMOD was completely stored and the SYSMOD and MCS entries in the PTS have been created.

If you are unsure about the status of a SYSMOD, issue LIST PTS SYSMOD to obtain a listing of the SYSMODs on the PTS. If the PTS SYSMOD entry is present with the ERROR status indicated, the SYSMOD is not ready for processing by the APPLY and ACCEPT functions.

If you are still unsure if the SYSMOD was completely processed by RECEIVE, use the REJECT control statement to delete the SYSMOD. After correcting any conditions that might have caused problems during the previous RECEIVE pass, reissue RECEIVE for the SYSMOD.

If an out-of-space condition occurs on the SMPPTS during RECEIVE processing, see 'Resolving Direct Access Storage Problems' in Chapter 5 for information on handling the problem and rerun RECEIVE.

The REJECT Control Statement

The SMP REJECT control statement removes SYSMODs from the SMPPTS and deletes any temporary libraries loaded during RECEIVE processing for those SYSMODs selected for rejection. Any number of REJECT statements can be included in an SMP job step.

REJECT Syntax

```
REJECT  [{SELECT | EXCLUDE} (sysmodid[,sysmodid]...)]
        [COMPRESS({ALL | ddname[,ddname]...})]
|       [PURGE]
        [RC(function=code[,function=code]...)]
        .
```

REJECT Operands

SELECT(sysmodid[,sysmodid]...)
specifies one or more SYSMODs to be removed from the PTS data set. This operand can also be specified as 'S'.

Note: If SELECT is not specified, then only those SYSMODs that have never been processed by APPLY or ACCEPT are selected for processing.

EXCLUDE(sysmodid[,sysmodid]...)
specifies one or more SYSMODs that are not to be removed from the PTS data set. This operand can also be specified as 'E'.

COMPRESS({ALL | ddname[,ddname]...})
specifies one or more partitioned data sets to be compressed. This operand can be specified as 'C'. When 'ALL' is specified, only data sets affected by REJECT processing are compressed.

Note: The CDS and ACDS data sets cannot be compressed. If specified, they are ignored.

| PURGE
When PURGE is coded on the REJECT statement, all SYSMOD's found on the ACDS (NOT 'IN ERROR') are removed from the PTS.

Note: If SELECT or EXCLUDE are to be usd with the PURGE option, the rules as stated for PURGE apply to the

SELECTED or EXCLUDED SYSMODS.

RC(function=code[,function=code]...)
specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When

specified, the RC operand must be the last operand on the REJECT statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, REJECT processing is bypassed and REJECT terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.

- If all specified SMP functions return codes less than or equal to their indicated codes, REJECT is executed.

- Previous processing by any SMP function not specified on the RC operand has no effect on the current REJECT processing.

REJECT DDnames

```
SMPCNTL  (required)
SMPLOG   (required)
SMPOUT   (required)
SMPPTS   (required)
SMPTLIB  (required if modifications were loaded to
          temporary libraries during RECEIVE)
SYSPRINT (required if COMPRESS is specified)
SYSUT1   (required)
SYSUT2   (required)
SYSUT3   (required)
```

REJECT Programming Considerations

1) A data set is specified as a COMPRESS operand value even if it is not affected by the REJECT process.

2) The compress function does not process keyed or unmovable data sets.

3) The compress function should <u>not</u> be performed if the target system libraries of the running operating system are eligible or selected for compression.

4) Processing time can increase significantly when the COMPRESS operand is specified.

5) If a function SYSMOD that has been neither applied nor accepted is rejected, the FMID subentry for its SYSMOD-ID is deleted from the PTS SYSTEM entry.

REJECT Return Codes

00 REJECT processing completed successfully and without errors.

04 REJECT processing completed, but there might be possible error or warning messages. The possible error conditions are:

1) IEBCOPY, invoked to compress a data set, returned an acceptable but non-zero return code.

2) A SYSMOD specified in the SELECT operand list was processed by APPLY or ACCEPT. The SYSMOD is not processed by REJECT.

08 REJECT processing completed, but errors were encountered and processing terminated for at least one SYSMOD. The possible error is:

1) PEMAX was too small to process a SYSMOD entry on the PTS and that SYSMOD has not been rejected.

12 REJECT processing terminated. The possible error conditions are:

1) No SYSMODs met REJECT specifications.

2) GETMAIN failed during REJECT processing.

3) An error occurred while opening or closing an SMP data set.

4) SMP detected a syntax error in the REJECT control statement.

5) A DD statement was missing.

6) The REJECT control statement was not processed because a previous control statement returned a non acceptable return code.

16 A severe error was encountered and SMP processing terminated. The possible error conditions are:

1) IEBCOPY, invoked to compress a data set, returned a non-acceptable return code. REJECT processing did not occur.

2) A severe error occurred while accessing an SMP data set.

3) An error occurred while writing a message.

REJECT Error Recovery

If a failure occurs during REJECT processing, issue the REJECT control statement for those SYSMODs that were not successfully rejected. If a function SYSMOD is being rejected, check the PTS SYSTEM entry to see if the FMID subentry for that SYSMOD-ID has been deleted. If it was not and should have been; that is, the SYSMOD was never applied or accepted), use the UCLIN function to delete the FMID subentry.

The RESETRC Control Statement

The RESETRC control statement resets the return code values previously returned by other functions invoked by SMP control statements. Any number of RESETRC statements can be included in an SMP job step.

RESETRC Syntax

```
RESETRC •
```

RESETRC Operands

There are no operands for this statement.

RESETRC DDnames

SMPCNTL	(required)
SMPLOG	(required)
SMPOUT	(required)

RESETRC Programming Considerations

1) Use of this control statement should be carefully analyzed. The statement should not be placed in the SMPCNTL input stream in front of statements that have a dependency on the processing results of the preceding statements.

2) When you are executing SMP in an interactive environment, you can use this statement after the completion of other statements when the function invoked by the previous statements did not complete successfully, but other functions need to be invoked. An alternative method is to specify the RC operand on any subsequent control statements, which can be cumbersome.

3) The RESETRC function does not affect the maximum return code value returned by SMP when it terminates execution. This value is always set to the highest value returned by any of the functions invoked during execution.

RESETRC Return Codes

The RESETRC control statement does not have any return codes.

The RESTORE Control Statement

The SMP RESTORE control statement removes SYSMODs processed by APPLY from target system libraries. Any number of RESTORE control statements can be coded in an SMP job step.

RESTORE Syntax

```
RESTORE {SELECT | GROUP}(sysmodid[,sysmodid]...)
        [BYPASS(ID)]
        [CHECK]
        [COMPRESS({ALL | ddname[,ddname]...})]
        [DIS( READ | NO | WRITE )]
        [RC(function=code[,function=code]...)]
        [RETRY(YES | NO)]
        .
```

RESTORE Operands

SELECT(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be restored on the target system libraries. This operand can also be specified as 'S'.

GROUP(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be restored on the target system libraries. If you specify GROUP, any other SYSMOD that references a specified SYSMODs as a requisite or prerequisite is also included in RESTORE processing. This operand can also be specified as 'G'.

  Other SYSMODs than those specified on the SELECT or GROUP operands might be required to synchronize the system with the level of the DLIBs. If you specify SELECT mode, you must explicitly specify all related SYSMODs.

BYPASS(ID)
: specifies that error conditions detected during ID checking of the FMID, RMID and UMID in the element entries on the CDS and/or the ACDS finding error conditions should not cause termination of any SYSMODs.

CHECK
: specifies the RESTORE processing of SYSMODs should not actually update libraries and SMP data sets. Instead, the following processing is performed:

testing for error conditions that can occur before restoring the SYSMODs.

reporting on libraries that would be updated during RESTORE processing.

Note: If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored; no compression is performed.

COMPRESS({ALL | ddname[,ddname]...})
specifies one or more ddnames of partitioned data sets to be compressed. This operand can also be specified as 'C'. Only the partitioned data sets affected by RESTORE processing are compressed by specifying 'ALL'.

Note: If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored; no compression is performed. The SMPACDS and SMPCDS data sets cannot be compressed. If specified, they are ignored.

DIS(READ | NO | WRITE)
specifies that the SMPCDS directory is to be in storage during processing.

READ is the default; it causes the directory to be in storage in read only mode. Updates to the directory entries are stowed as they occur.

NO specifies that the directory is not to be in storage during processing. All reading of directory entries is done from the data set itself and updates to the directory entries are stowed as they occur.

WRITE specifies that the directory is to be in storage for both reading and updating. Updates to the directory entries are performed on the in- storage copy as they occur; the entire directory is written to the data set when RESTORE processing completes.

Note: If DIS(NO) is specified with the CHECK operand, it is ignored and DIS(READ), the default value, is used.

RC(function=code[,function=code]...)
specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the RESTORE statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, RESTORE processing is bypassed and RESTORE terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.
- If all specified SMP functions return codes less than or equal to their indicated codes, RESTORE is executed.
- Previous processing by any SMP function not specified on the RC operand has no effect on the current RESTORE processing.

| RETRY(YES | NO)
where 'YES' indicates that SMP4 is to attempt a RETRY for each utility failure during the function. 'NO' indicates that no RETRY is to be attempted. 'YES' is the default mode of operation if the RETRY keyword is not specified and a DDname list is available.

RESTORE DDnames

distlib	(one for each library containing copies of the elements being restored)
SMPACDS	(required)
SMPCDS	(required)
SMPCNTL	(required)
SMPCRQ	(required)
SMPLOG	(required)
SMPMTS	(required)
SMPOUT	(required)
SMPPTS	(required)
SMPRPT	(optional)
SMPSCDS	(required)
SMPSTS	(required)
SMPTLIB	(if modifications were loaded to temporary libraries during RECEIVE and the REJECT indicator in the PTS SYSTEM entry is set on)
SMPWRK1	(required)
SMPWRK2	(required)
SMPWRK3	(required)
SMPWRK4	(required)
SYSLIB	(required)
SYSPRINT	(required)

SYSUT1	(required)
SYSUT2	(required)
SYSUT3	(required)
SYSUT4	(required for RETRY)
tgtlib	(one for each target system library to be restored)

RESTORE Programming Considerations

1) If the REJECT indicator is set off in the PTS SYSTEM entry, then a successfully restored SYSMOD is not deleted from the PTS.

2) SYSMOD entries on the CDS have the ERROR and RESTORE status indicators set on before the target system libraries are updated. If processing fails during the updating, these indicators will remain on and the updating for these entries is not completed. After you determine the cause of the termination, you can process these SYSMODs again by specifying them as operand values of the SELECT operand on the RESTORE control statement.

3) The ddnames for target system and distribution libraries can be determined by specifying the CHECK operand on the RESTORE control statement. The ddnames are listed in the ELEMENT SUMMARY report the SMPRPT data set.

4) If a compress of affected data sets is not performed before or during RESTORE processing, out of space conditions can occur in the target system libraries. As a rule, compressing libraries on a running operating system should be avoided and an alternate system should be used in its place. The COMPRESS option can not process keyed or unmovable data sets. The data sets eligible for compressing are any target system libraries affected by the SMP job step (that is, the data set defined on any DD statement that specifies a partitioned data set that is not an SMP data set). Processing time might increase significantly if the COMPRESS operand is specified on the RESTORE control statement.

   During COMPRESS processing for RESTORE, target system elements that were copied during SYSGEN, reside in data sets specified in the COMPRESS operand, and are affected by SYSMODs specified for RESTORE are deleted before the compression.

5) RESTORE processing does not replace the nucleus with the saved copy.

6) When a selected SYSMOD contains an element that was added to the system by that SYSMOD, RESTORE processing deletes that element from all target system libraries in which it is found and deletes the corresponding element entry (that is, the MAC, MOD, or SRC entry) from the CDS.

7) When a selected SYSMOD contains an element that was deleted from the system by that SYSMOD, RESTORE processing reintroduces that element to the target system with the corresponding element entry copied from the SCDS data set.

8) Use the DIS(NO) option only when the number of SYSMODs and their elements is small or if when the tradeoff between storage utilization and performance has to be made in favor of storage.

9) The DIS(NO) option should not be used if the previous SMP control statement was APPLY, ACCEPT, RESTORE, JCLIN, or UCLIN specified without the DIS(NO) option and the same directory is to be used.

10) You do not have to use SMP to recover after a failure. You have the option of restoring your system and the distribution libraries via system and DLIB restore tapes. In this situation, ensure that the CDS, MTS, ACDS, PTS, and STS are also restored to their previous levels.

RESTORE Return Codes

00 RESTORE processing completed successfully and without errors.

04 RESTORE processing completed, but there are possible error or warning messages. The possible error conditions are:

1) RESTORE invoked a system program to perform some work and the system program returned a non zero but still acceptable return code. One of the following system programs could have generated this return code:

- Assembler (ASMBLR)
- IEBCOPY - invoked to copy a module, macro, or source module, or to compress a data set
- IEBUPDTE - invoked to update a macro or source module
- Linkage editor (IEWL)

The affected SYSMOD is restored and is marked RESTORE in the CDS.

2) During RESTORE processing, assembly input for a selected module was not found on the CDS.

08 RESTORE processing completed, but processing errors were encountered resulting in the termination of at least one SYSMOD. The possible error conditions are:

1) RESTORE invoked a system program to perform some work and the system program returned a non zero but still acceptable return code. One of the following system programs could have generated this return code:

- Assembler (ASMBLR)
- IEBCOPY - invoked to copy a module, macro, or source module
- IEBUPDTE - invoked to update a macro or source module
- Linkage editor (IEWL)

The affected SYSMOD entries have the RESTORE and ERROR status indicators set in the CDS.

2) A DD statement was missing. RESTORE did not process any SYSMOD that requires the missing DD statement.

3) A SYSMOD selected for RESTORE processing was never processed by APPLY, but a SYSMOD entry exists that was created by the processing of another SYSMOD that superseded it. RESTORE processing did not affect the superseded and superseding SYSMODs.

4) RESTORE processing requires an element entry that cannot be found on the CDS. RESTORE processing terminated for all affected SYSMODs.

5) PEMAX was too small to process a selected SYSMOD or element entry.

6) A SYSMOD selected for RESTORE has already been processed by ACCEPT. RESTORE did not process the affected SYSMOD.

7) During RESTORE processing an error occurred while opening a required data set. RESTORE processing was terminated for all affected SYSMODs.

12 RESTORE processing terminated. The possible error conditions are:

1) No SYSMODs met RESTORE specifications.

2) GETMAIN failed during RESTORE processing.

3) An error occurred while opening an SMP data set.

4) A syntax error was detected in the RESTORE control statement.

5) A DD statement was missing.

6) The RESTORE control statement was not processed because a previous control statement returned a non acceptable return code.

16 A severe error was encountered and SMP processing was terminated. The possible error conditions are:

1) IEBCOPY, invoked to compress a data set, returned a non acceptable return code. RESTORE processing did not occur, but the modules within the subject SYSMODs that were candidates for replacement during RESTORE were deleted from the appropriate target system libraries.

<u>Note</u>: The target system libraries might be unusable; that is,the system or some of its components might not be executable.

2) A severe error occurred while deleting members from a target system library before compression of that library.

   Note: The target system libraries might be unusable; that is, the system or some of its components might not be executable.

3) An error occurred while writing a message.

4) A severe error occurred while accessing an SMP data set.

5) A non acceptable return code was returned from IEHIOSUP.

RESTORE Error Recovery

After the RESTORE function completes, examine SMPOUT and SYSPRINT output to determine the relative success of the function. Note that partially restored SYSMOD entries have the RESTORE and ERROR status indicators set in the CDS.

You should rerun RESTORE for a SYSMOD that failed during previous RESTORE processing. The following processing takes place:

- All linkage editor processing is repeated.
- All IEBCOPY processing is repeated.
- All assemblies are repeated.
- All IEBUPDTE processing is repeated.

If an out-of-space condition occurs on any library during RESTORE processing, see 'Resolving Direct Access Storage Shortage Problems' in Chapter 5 for information on how to handle the problem; rerun RESTORE.

The UCLIN Control Statement

The UCLIN control statement specifies, by means of an operand, the SMP data set whose entries are to be updated by processing the update control language (UCL) statements that follow the UCLIN control statement in the SMPCNTL input data set. The UCLIN statement must be followed by one or more UCL statements and an ENDUCL statement. There can be more than one UCLIN statement in the SMPCNTL input data set.

The UCL processing function is provided as a means to correct data present in entries on SMP data sets and to define parameters used in processing. For most entries, the data that is present is due to the processing of modification control statements in SYSMODs by the RECEIVE, APPLY, RESTORE, ACCEPT, and JCLIN functions. If these SYSMODs are processed correctly, there should seldom be a need to invoke UCL processing.

UCLIN Syntax

```
UCLIN [ACDS |
       ACRQ |
       CDS |
       CRQ |
       MTS |
       PTS |
       SCDS |
       STS]
       [DIS( READ | NO | WRITE )]
       [RC(function=code[,function=code]...)]
       .
```

UCLIN Operands

Note: There is no default data set; one of the following must be specified.

ACDS
Specifies that succeeding UCL statements apply to the SMPACDS data set.

ACRQ
Specifies that succeeding UCL statements apply to the SMPACRQ data set.

CDS
: Specifies that succeeding UCL statements apply to the SMPCDS data set.

CRQ
: Specifies that succeeding UCL statements apply to the SMPCRQ data set.

MTS
: Specifies that succeeding UCL statements apply to the SMPMTS data set.

PTS
: Specifies that succeeding UCL statements apply to the SMPPTS data set.

SCDS
: Specifies that succeeding UCL statements apply to the SMPSCDS data set.

STS
: Specifies that succeeding UCL statements apply to the SMPSTS data set.

DIS(<u>READ</u> | NO | WRITE)
: specifies a directory in storage option. The directory used is dependent on the data set being updated and only has meaning if ACDS, ACRQ, CDS, or CRQ is specified.

  READ is the default and it causes the directory to be in storage in read only mode. Updates to the directory entries are stowed as they occur.

  NO specifies that the directory is not to be in storage during processing. All reading of directory entries is done from the data set itself and updates to the directory entries are stowed as they occur.

  WRITE specifies that the directory is to be in storage for both reading and updating. Updates to the directory entries are performed on the copy in storage as they occur and the entire directory is written to the data set when UCLIN processing completes.

RC(function=code[,function=code]...)
: specifies one or more SMP functions with associated return codes to enable you to bypass normal SMP return code processing. The function specified must be one of the following: ACCEPT, APPLY, JCLIN, LIST, LOG, RECEIVE, REJECT, RESTORE or UCLIN. The code specified must be a decimal number that is greater than or equal to 0 and less than 16. The code specified cannot equal 16. When specified, the RC operand must be the last operand on the UCLIN statement, or a syntax error results.

Specifying the RC operand causes the following return code processing to occur:

- If any specified function returns a code greater than its specified code, UCLIN processing is bypassed and UCLIN terminates with a return code of 12. The default codes are 8 or greater from UCLIN and JCLIN, and 12 or greater from all other functions.
- If all specified SMP functions return codes less than or equal to their indicated codes, UCLIN is executed.
- Previous processing by any SMP function not specified on the RC operand has no effect on the current UCLIN processing.

UCLIN DDnames

```
SMPACDS  (required if ACDS specified as operand)
SMPACRQ  (required if ACRQ specified as operand)
SMPCDS   (required if CDS specified as operand)
SMPCNTL  (required)
SMPCRQ   (required if CRQ specified as operand)
SMPMTS   (required if MTS specified as operand)
SMPLOG   (required)
SMPOUT   (required)
SMPPTS   (required if PTS specified as operand)
SMPSCDS  (required if SCDS specified as operand)
SMPSTS   (required if STS specified as operand)
```

UCLIN Programming Considerations

1) Each UCLIN control statement must be followed by at least one UCL statement.

2) The ENDUCL control statement must terminate the UCL statements.

3) Use the DIS(NO) option only when the number of updates to entries is small or when the tradeoff between storage utilization and performance has to be made in favor of storage.

4) For performance reasons, the DIS(NO) option should not be specified if the previous SMP control statement was APPLY, ACCEPT, RESTORE, JCLIN, or UCLIN specified without the DIS(NO) option and the same directory is to be used.

5) If you change the MOD, MAC, or SRC entry and the entry that results has an FMID and no RMID, the FMID value becomes the RMID value.

UCLIN Return Codes

00 UCLIN processing completed successfully and without errors.

04 UCLIN processing completed, but with unexpected results:

1) End-of-file was encountered in the SMPCNTL data set before an ENDUCL control statement was processed.

2) No UCL statement followed the UCLIN control statement.

08 UCLIN processing completed with errors. The possible error conditions are:

1) A syntax error was detected in at least one UCL input statement.

2) At least one UCL statement does not meet conditions for update.

12 UCLIN processing terminated:

1) A syntax error was detected the UCLIN control statement.

2) No processing occurred due to an unacceptable return code from a previous function.

3) A DD statement was missing for a required data set.

16 A severe error was encountered; SMP processing was terminated.

UCLIN Error Recovery

If a failure occurs when processing a UCL statement, follow the actions recommended in the Programmer Response section for the message describing the failure.

If the DIS(WRITE) option was specified on the UCLIN control statement and the failure occurred during the rewrite of the CDS or ACDS directory entries, see Directory-in-Storage Related Errors in Chapter 5.

The UCL Statements

UCL statements are used to add, delete, and modify entries in the ACDS, ACRQ, CDS, CRQ, MTS, PTS, SCDS, and STS data sets. A UCL statement must be preceded, in the SMPCNTL data set, by another UCL statement or by a UCLIN control statement that defines the SMP data set against which the succeeding UCL statements are to operate. A UCL statement must be followed by another UCL statement or an ENDUCL control statement.

UCL Syntax

```
{ ADD | DEL | REP }
      { ASSEM(name) |
        DLIB(name) |
        FMID(name) |
        LMOD(name) |
        MAC(name) |
        MOD(name) |
        PTF(name) |
        SRC(name) |
        SYS |
        SYSMOD(name) }
      [option[,option]...]
      .
```

UCL Operands

ADD

specifies that new data is to be added to an existing entry or that a new entry is to be created on the ACDS, ACRQ, CDS, CRQ, or PTS.

For ADD operations, either the entry specified must not exist or, if it does, the subentries specified within the entry must not be present and the indicators specified within the entry must be in reset state. If any of these conditions is false, then a message is issued indicating the invalid condition and the update to the entry, subentry, or indicator is not done.

If the above verification succeeds, then the following updating is done:

If the entry is being created, all subentries are set to the specified values and indicators placed in set state. For example:

```
ADD MOD(XYZ) DISTLIB(AOS99).
```

creates a MOD entry for module XYZ.

Subentries are added to the existing entry using the specified values. For example:

```
ADD MOD(XYZ) UMID(UZ12345,UZ13579).
```

adds two UMID subentries to MOD entry XYZ.

Indicators are placed in set state in the existing entry. For example:

```
ADD SYSMOD(UZ12345) RESTORE.
```

sets the RESTORE indicator in SYSMOD entry UZ12345.

DEL
specifies that an entry is to be deleted or, within an entry, subentries are to be deleted and indicators placed in reset state. Valid data sets are the ACDS, ACRQ, CDS, CRQ, MTS, PTS, SCDS, or STS. Only entries can be deleted from the MTS, SCDS, and STS data sets.

For DEL operations, the specified entry must exist, subentries within an entry must exist and contain the same data as is specified in the operand or be unconditionally deleted, and indicators must be in set state. If any of these conditions is false, then a message is issued indicating the invalid condition and the update is not done.

If the above verification succeeds, the following updating is done:

If the only operand specified is the entry type with name, the entry is deleted from the data set. For example:

```
DEL SYSMOD(UZ12345).
```

deletes the SYSMOD entry for UZ12345.

For subentries, either the individual subentry is deleted, the specified list of subentries is deleted, or all subentries of the same type are deleted. An unconditional delete of a single subentry or all subentries of the same type is done if the operand name is followed by a pair of parentheses with no value. For example:

```
DEL MAC(ABC) UMID( ).
```

deletes all UMID subentries in the MAC entry for macro ABC. The parentheses may be contiguous, such as (), or be separated by any number of blanks, such as (   ).

Indicators are placed in reset state within the entry. For example:

```
DEL SYSMOD(UZ12345) ERROR.
```

resets the ERROR indicator in the SYSMOD entry for UZ12345.

REP
specifies that subentries are to be replaced and indicators placed in set state in an existing entry or, if the entry did not exist, it is to be created using the criteria for ADD operations. The valid data sets are the ACDS, ACRQ, CDS, CRQ, or PTS.

For REP operations, if the entry did not exist or a subentry within an existing entry did not exist, then a message indicating that the entry or subentry did not exist is issued and that an ADD operation is assumed. This message is issued only once per entry or subentry. All processing from this point on follows the rules for ADD.

If the subentry exists within an existing entry, then all subentries of the same type are replaced with the values specified in the operand. For example:

```
REP SYSMOD(UZ12345) SUPING(AZ11111,AZ11122).
```

replaces all SUPING subentries in the SYSMOD entry for UZ12345.

Indicators within an existing entry are placed in set state. For example:

```
REP SYSMOD(UZ12345) ERROR.
```

sets the ERROR indicator in the SYSMOD entry for UZ12345.

ASSEM(name)
specifies an ASSEM entry. ASSEM entries only exist on the CDS. The only valid operation is DEL with no optional operands.

DLIB(name)
specifies a DLIB entry. DLIB entries only exist on the CDS. ADD, DEL, and REP operations of any form are permitted.

FMID(name)
specifies a FMID entry. FMID entries exist on the ACRQ and CRQ. ADD, DEL, and REP operations of any form are permitted.

LMOD(name)
specifies a LMOD entry. LMOD entries only exist on the CDS. ADD, DEL, and REP operations are permitted, however, an LMOD entry cannot be created with UCLIN process.

MAC(name)
specifies a MAC entry. MAC entries exist on the ACDS, CDS, and MTS. ADD, DEL, and REP operations of any form are permitted for ACDS and CDS MAC entries. For MAC entries on the MTS, the only valid operation is DEL with no optional operands.

MOD(name)
specifies a MOD entry. MOD entries exist on the ACDS and CDS. ADD, DEL, and REP operations of any form are permitted.

PTF(name)
specifies a SYSMOD entry. PTF is equivalent to the SYSMOD operand and is included for compatibility with UCL statements processable by previous versions of SMP for operations on the ACDS and CDS only. The syntax and operands are described in the UCL SYSMOD statement section.

SRC(name)
specifies a SRC entry. SRC entries exist on the ACDS, CDS, and STS. ADD, DEL, and REP operations of any form are permitted for ACDS and CDS SRC entries. For SRC entries on the STS, the only valid operation is DEL with no optional operands.

SYS
specifies a SYSTEM entry. A SYSTEM entry exists on the ACDS, CDS, and PTS. ADD, DEL, and REP operations of any form are permitted.

SYSMOD(name)
specifies a SYSMOD entry. SYSMOD entries exist on the ACDS, ACRQ, CDS, CRQ, PTS, and SCDS. ADD, DEL, and REP operations of any form are permitted for ACDS, ACRQ, CDS, and ACDS SYSMOD entries. For SYSMOD entries on the SCDS, the only valid operation is DEL with no optional operands. For SYSMOD entries on the PTS, an ADD or REP operation cannot create an entry and a DEL operation cannot delete an entry.

[option[,option]...]
specifies the options that are available for each ASSEM, DLIB, FMID, LMOD, MAC, MOD, PTF, SRC, SYS, and SYSMOD operand. The syntax and explanation of these options are described on the following pages.

UCL ASSEM Syntax

```
        { ADD | DEL | REP } ASSEM (name)
|       ASMIN
        ASSEMBLER INPUT CARD1
        ASSEMBLER INPUT CARD2
           •
           •
|       ENDASMIN
|       LASTUPD((JCLIN|UCLIN|SYSMODID)
|       LASTUPDTYPE(ADD|UPD)
        •
```

UCL ASSEM Operands

ASSEM(name)
specifies an ASSEM entry to be deleted from the CDS, where "name" is the one to eight character ASSEM entry name.

| ASMIN
indicates that assembler input control cards follow. This operand must start in col 1. If specified then ENDASIN must also be specified. If DELETE is specified then no comparison is made between the assembler input entered and that already in the CDS. No other options may be specified on the same line as ASMIN.

| ENDASMIN
must start in column 1. End of assembler input.

| LASTUPD(JCLIN|UCLIN|SYSMOD)
identifies the cause of the last change made to this entry.

| LASTUPDTYPE(ADD|UPD)
identifies the last type of update made to this entry.

UCL DLIB Syntax

```
{ ADD | DEL | REP } DLIB (name)
   [SYSLIB(ddname[,ddname])]
LASTUPD(JCLIN|UCLIN|SYSMODID)
LASTUPDTYPE(ADD|UPD)
.
```

UCL DLIB Operands

DLIB(name)
specifies a DLIB entry to be created or deleted, or subentries within the DLIB entry to be added, deleted, or replaced on the CDS, where "name" is the one to eight character DLIB entry name, which is the ddname of the distribution library.

SYSLIB(ddname[,ddname])
specifies one or two SYSLIB subentries, where "ddname" is a target system library ddname that the distribution library members were copied to.

Note: When creating a DLIB entry, this operand must be specified. When deleting a SYSLIB subentry, at least one SYSLIB subentry must remain.

| LASTUPD(JCLIN|UCLIN|SYSMODID)
Identifies the cause of the last change made to this entry.

| LASTUPDTYPE(ADD|UPD)
Identifies the last type of update made to this entry.

UCL FMID Syntax

```
{ ADD | DEL | REP } FMID(name)
   [SYSMOD(sysmodid[,sysmodid])]
   .
```

UCL FMID Operands

FMID(name)
specifies a FMID entry to be created or deleted, or subentries within the FMID entry to be added, deleted, or replaced on the ACRQ or CRQ, where "name" is the one to eight character FMID entry name, which is the SYSMOD-ID of a function SYSMOD.

SYSMOD(sysmodid[,sysmodid])
specifies one or more SYSMOD subentries, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is a SYSMOD entry on the ACRQ or CRQ.

Note: When creating a FMID entry, this operand must be specified. When deleting a SYSMOD subentry, at least one SYSMOD subentry must remain.

UCL LMOD Syntax

```
{ ADD | DEL | REP } LMOD(name)
   [AC=1]
   [ALIGN2]
   [COPY]
   [DC]
   [NE]
   [OVLY]
   [REFR]
   [RENT]
   [REUS]
   [SCTR]
   [STD]
   [SYSLIB(ddname[,ddname])]
|  [LMODIN]
```

```
| [ENDLMODIN]
| [LASTUPD[JCLIN|UCLIN|SYSMODID])]
| [LASTUPDTYPE[ADD|UPD])] •
```

UCL LMOD Operands

LMOD(name)
specifies a LMOD entry to be deleted, or subentries within the LMOD entry to be added, deleted, or replaced on the CDS, where "name" is the one to eight character LMOD entry name.

Note: An LMOD entry cannot be created by UCLIN processing.

AC=1
specifies the AC=1 indicator, which is the authorization code. When this indicator is set, the AC=1 parameter is passed to the linkage editor program when the load module is link edited.

ALIGN2
specifies the ALIGN2 indicator, which is alignment on a 2K boundary. This operand can also be specified as "ALN2". When this indicator is set, the ALIGN2 parameter is passed to the linkage editor program when the load module is link edited.

COPY
specifies the COPY indicator, which means the load module was copied at system generation time.

DC
specifies the DC indicator, which is the downward-compatible load module attribute. When this indicator is set, the DC parameter is passed to the linkage editor program when the load module is link edited.

NE
specifies the NE indicator, which is the non-editable load module attribute. When this indicator is set, the NE parameter is passed to the linkage editor program when the load module is link edited.

OVLY
specifies the OVLY indicator, which is the overlay attribute. When this indicator is set, the OVLY parameter is passed to the linkage editor program when the load module is link edited.

REFR
specifies the REFR indicator, which is the refreshable attribute. When this indicator is set, the REFR parameter is passed to the linkage editor program when the load module is link edited.

RENT
specifies the RENT indicator, which is the reenterable attribute. When this indicator is set, the RENT parameter is passed to the linkage editor program when the load module is link edited.

REUS
specifies the REUS indicator, which is the reusable attribute. When this indicator is set, the REUS parameter is passed to the linkage editor program when the load module is link edited.

SCTR
specifies the SCTR indicator, which is the scatter load attribute. When this indicator is set, the SCTR parameter is passed to the linkage editor program when the load module is link edited.

STD
specifies the STD indicator for standard linkage editor attributes. The standard attributes are NCAL, LET, LIST, and XREF, and is the minimum default attribute if the load module is link edited. When this indicator is set, the standard parameters are passed to the linkage editor program when the load module is link edited. The remaining attributes, as defined above, augment the standard attributes when their associated indicators are set.

SYSLIB(ddname[,ddname])
specifies one or two SYSLIB subentries, where "ddname" is a target system library ddname that contains the load module.

Note: When creating a LMOD entry, this operand must be specified. When deleting a SYSLIB subentry, at least one SYSLIB subentry must remain.

| LMODIN
Indicates that linkage editor input cards follow. This operand must start in column 1. If specified then ENDLMODIN must also be specified. If DELETE is specified then no comparison is made between the linkage editor input entered and that already in the CDS. The existing linkage editor control cards are deleted. If REP is specified all existing control cards (including CHANGE/REPLACE control cards) are replaced by those entered. This is a difference from JCLIN processing of linkage editor steps where all cards are replaced except CHANGE/REPLACE which are merged with the existing CHANGE/REPLACE cards. Changing the LMOD linkage editor control cards does not change or create any other entries in the CDS. If a MOD is added to the LMOD and LMODIN is specified for the LMOD, then the user must also add of modify the CDS module entry. No other options may be specified on the same line as LMODIN.

| ENDLMODIN
Indicates end of Linkage Editor input.

| LASTUPD(JCLIN|UCLIN|SYSMOD)
Identifies the cause of the last change made to this entry.

| LASTUPDTYPE(ADD|UPD)
Identifies the last type of update made to this entry.

UCL MAC Syntax

```
{ ADD | DEL | REP } MAC(name)
   [DISTLIB(ddname)]
   [FMID(sysmodid)]
   [GENASM(name[,name]...)]
   [MALIAS(alias[,alias]...)]
   [RMID(sysmodid)]
   [SYSLIB(ddname)]
   [UMID(sysmodid[,sysmodid]...)]
|  [LASTUPD[JCLIN|UCLIN|SYSMODID])]
|  [LASTUPDTYPE[ADD|UPD])]
   .
```

UCL MAC Operands

MAC(name)
specifies a MAC entry or subentries within an entry to be added, deleted, or replaced on the ACDS, CDS, or MTS, where "name" is the one to eight character macro name. For the MTS, only DEL with no other operands can be specified.

DISTLIB(ddname)
specifies the DISTLIB subentry, where "ddname" is the one to eight character distribution library ddname. This operand can also be specified as "DLIB".

Note: When creating a new entry, DISTLIB must be specified and the DISTLIB subentry cannot be deleted from an entry.

FMID(sysmodid)
specifies the FMID subentry, where "sysmodid" is the SYSMOD-ID of the function SYSMOD which owns the macro.

GENASM(name[,name]...)
specifies one or more GENASM subentries, where "name" is a one to eight character ASSEM or SRC entry name.

Note: This operand can be used to add ASSEM and SRC entry names whose source text includes the macro. This causes the assembly of the source text during APPLY processing for CDS MAC entries and during ACCEPT processing for ACDS MAC entries when the macro is modified.

MALIAS(alias[,alias]...)
specifies one or more MALIAS subentries, where "alias" is a one to eight character alias name of the macro in the distribution library and, if present, in the target system library.

RMID(sysmodid)
specifies the RMID subentry, where "sysmodid" is the SYSMOD-ID of the SYSMOD that last replaced the macro text.

SYSLIB(ddname)
specifies the SYSLIB subentry, where "ddname" is the target system library ddname.

Note: If the SYSLIB subentry is not present in or is deleted from a CDS MAC entry, modifications to the macro results in the macro text being placed in the MTS during APPLY processing.

UMID(sysmodid[,sysmodid]...)
specifies one or more UMID subentries, where "sysmodid" is the SYSMOD-ID of a SYSMOD that updated the macro since it was last replaced.

| LASTUPD(JCLIN|UCLIN|SYSMOD)
Identifies the cause of the last change made to this entry.

| LASTUPDTYPE(ADD|UPD)
Identifies the last type of update made to this entry.

UCL MOD Syntax

```
      { ADD | DEL | REP } MOD(name)
         [DALIAS(alias[,alias]...)]
         [DISTLIB(ddname)]
         [FMID(sysmodid)]
         [LMOD(name[,name]...)]
         [RMID(sysmodid)]
         [TALIAS(alias[,alias]...)]
         [UMID(sysmodid[,sysmodid]...)]
|        [LASTUPD[JCLIN|UCLIN|SYSMODID])]
|        [LASTUPDTYPE[ADD|UPD])]
         .
```

UCL MOD Operands

MOD(name)
specifies a MOD entry or subentries within an entry to be added, deleted, or replaced on the ACDS or CDS, where "name" is the one to eight character MOD entry name.

DALIAS(alias[,alias]...)
specifies one or more DALIAS subentries, where "alias" is a one to eight character alias name of the module in the distribution library and, for a copied module, in the target system library.

Note: DALIAS subentries are equivalent to TALIAS subentries, therefore, either operand can be used to add, delete, or replace.

DISTLIB(ddname)
specifies the DISTLIB subentry, where "ddname" is the one to eight character distribution library ddname. This operand can also be specified as "DLIB".

Note: When creating a new MOD entry, the DISTLIB operand must be specified and the DISTLIB subentry cannot be deleted.

FMID(sysmodid)
specifies the FMID subentry, where "sysmodid" is the SYSMOD-ID of the function SYSMOD which owns the module.

LMOD(name[,name]...)
specifies one or more LMOD subentries, where "name" is an LMOD entry name.

Note: When creating a MOD entry with the UCL MOD statement, if no LMOD operand is specified, an LMOD subentry with the same name as the MOD entry is placed in the MOD entry.

RMID(sysmodid)
specifies the RMID subentry, where "sysmodid" is the SYSMOD-ID of the SYSMOD that last replaced the module.

TALIAS(alias[,alias]...)
specifies one or more TALIAS subentries, where "alias" is a one to eight character alias name of the module in the distribution library and, for a copied module, in

the target system library.

Note: TALIAS subentries are equivalent to DALIAS subentries, therefore, either operand can be used to add, delete, or replace.

UMID(sysmodid[,sysmodid]...)
specifies one or more UMID subentries, where "sysmodid" is the SYSMOD-ID of a SYSMOD that updated, via IMASPZAP control statements, the module since it was last replaced.

| LASTUPD(JCLIN|UCLIN|SYSMOD)
Identifies the cause of the last change made to this entry.

| LASTUPDTYPE(ADD|UPD)
Identifies the last type of update made to this entry.

UCL SRC Syntax

```
  { ADD | DEL | REP } SRC(name)
     [DISTLIB(ddname)]
     [FMID(sysmodid)]
     [RMID(sysmodid)]
     [SYSLIB(ddname)
     [UMID(sysmodid[,sysmodid]...)]
|    [LASTUPD[JCLIN|UCLIN|SYSMODID])]
|    [LASTUPDTYPE[ADD|UPD])]
     .
```

UCL SRC Operands

SRC(name)
specifies a SRC entry or subentries within an entry to be added, deleted, or replaced on the ACDS, CDS, or STS, where "name" is the one to eight character source module name. For the STS, only DEL with no other operands can be specified.

DISTLIB(ddname)
specifies the DISTLIB subentry, where "ddname" is the one to eight character distribution library ddname. This operand can also be specified as "DLIB".

Note: When creating a new entry, DISTLIB must be specified and the DISTLIB subentry cannot be deleted from an entry.

FMID(sysmodid)
specifies the FMID subentry, where "sysmodid" is the SYSMOD-ID of the function SYSMOD which owns the source module.

RMID(sysmodid)
specifies the RMID subentry, where "sysmodid" is the SYSMOD-ID of the SYSMOD that last replaced the source text.

SYSLIB(ddname)
specifies the SYSLIB subentry, where "ddname" is the target system library ddname.

Note: If the SYSLIB subentry is not present in or is deleted from a CDS SRC entry, modifications to the source module results in the source text being placed in the STS during APPLY processing.

UMID(sysmodid[,sysmodid]...)
specifies one or more UMID subentries, where "sysmodid" is the SYSMOD-ID of a SYSMOD that updated the source module since it was last replaced.

| LASTUPD(JCLIN|UCLIN|SYSMOD)
Identifies the cause of the last change made to this entry.

| LASTUPDTYPE(ADD|UPD)
Identifies the last type of update made to this entry.

UCL SYS Syntax

```
{ ADD | DEL | REP } SYS
   [ASMNAME(name)]
   [ASMPARM(parm)]
   [ASMPRINT(ddname)]
   [ASMRC(value)]
   [CDSID(name)]
   [COMPNAME(name)]
   [COMPPARM(parm)]
   [COMPPRINT(ddname)]
   [COMPRC(value)]
   [COPYNAME(name)]
   [COPYPARM(parm)]
   [COPYPRINT(ddname)]
   [COPYRC(value)]
   [DSPREFIX(prefix)]
   [DSSPACE(prim,sec,dirblks)]
   [FMID(sysmodid[,sysmodid]...)]
   [IOSUPNAME(name)]
   [IOSUPPARM(parm)]
   [IOSUPPRINT(ddname)]
   [IOSUPRC(value)]
   [LKEDNAME(name)]
   [LKEDPARM(parm)]
   [LKEDPRINT(ddname)]
   [LKEDRC(value)]
   [NUCID(n)]
   [PAGELEN(nnnn)]
   [PEMAX(nnnn)]
   [PURGE]
   [REJECT]
|  [RETRYNAME(name)]
|  [RETRYPARM(parm)]
|  [RETRYPRINT(ddname)]
|  [RETRYRC(value)]
|  [RETRYDDN(all | ddname,ddname,. . .)
   [SAVEMTS]
   [SAVESTS]
   [SREL(cnnn[,cnnn]...)]
   [UPDATNAME(name)]
   [UPDATPARM(parm)]
   [UPDATPRINT(ddname)]
   [UPDATRC(value)]
   [ZAPNAME(name)]
   [ZAPPARM(parm)]
   [ZAPPRINT(ddname)]
   [ZAPRC(value)]
   .
```

SYS

specifies a SYSTEM entry or subentries and indicators within an entry to be added, deleted, or replaced on the ACDS, CDS, or PTS.

Note: Changes to a SYSTEM entry are effective immediately after processing of the UCL SYS statement.

ASMNAME(name)

specifies the ASMNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the assembler function.

Note: If the ASMNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program ASMBLR to perform the assembler function. If you chose to use a different assembler program, ensure that it uses the SYSPUNCH DD statement, which is used as the output data set for the object text.

ASMPARM(parm)

specifies the ASMPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the assembler function. A maximum of 100 characters may be specified.

Note: If the ASMPARM subentry is not present in the PTS SYSTEM entry, SMP passes the character string "XREF,NOLOAD,DECK" to the invoked program. If you specify an ASMPARM subentry, ensure that DECK is included or that your substitute assembler program produces an object text deck.

ASMPRINT(ddname)

specifies the ASMPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the assembler program.

Note: If the ASMPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the ASMPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the assembler program.

ASMRC(value)

specifies the ASMRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the assembler program. When the value returned is higher than the ASMRC subentry value, then the result of the assembler function is considered unsuccessful and the SYSMOD for which the assembler program was invoked is terminated. The value may be any number from 0 to 16.

See OS/VS and DOS/VS Assembler Language for a description of the assembler return codes for program ASMBLR.

Note: If the ASMRC subentry is not present in the PTS SYSTEM entry, then the default value of 4 is compared with the assembler program return code.

CDSID(name)

specifies the CDSID subentry of the ACDS or CDS SYSTEM entry, where "name" is a one to eight character identifier for the control data set. The CDSID subentry value from the CDS SYSTEM entry is placed in the SYSMOD entry on the PTS as an APPID subentry when the SYSMOD is applied. The CDSID subentry value from the ACDS SYSTEM entry is placed in the SYSMOD entry on the PTS as an ACCID subentry when the SYSMOD is accepted.

Note: This operand is required when creating the SYSTEM entry on the ACDS and CDS.

COMPNAME(name)

specifies the COMPNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the PDS compress function.

Note: If the COMPNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IEBCOPY to perform the PDS compress function.

COMPPARM(parm)

specifies the COMPPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the PDS compress function. A maximum of 100 characters may be specified.

Note: If the COMPPARM subentry is not present in the PTS SYSTEM entry, SMP does not pass any parameters to the PDS compress program. If you specify a COMPPARM subentry, ensure that the parameters are valid for your substitute PDS compress program or IEBCOPY.

COMPPRINT(ddname)
specifies the COMPPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the PDS compress program.

Note: If the COMPPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the COMPPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the PDS compress program.

COMPRC(value)
specifies the COMPRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the PDS compress program. When the value returned is higher than the COMPRC subentry value, then the result of the PDS compress function is considered unsuccessful and the SMP function which invoked the PDS compress program is terminated. The value may be any number from 0 to 16.

See OS/VS Utilities for a description of the IEBCOPY return codes.

Note: If the COMPRC subentry is not present in the PTS SYSTEM entry, then the default value of 0 is compared with the PDS compress program return code.

COPYNAME(name)
specifies the COPYNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the PDS copy and load functions.

Note: If the COPYNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IEBCOPY to perform the PDS copy and load functions.

COPYPARM(parm)
specifies the COPYPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the PDS copy and load functions. A maximum of 100 characters may be specified.

Note: If the COPYPARM subentry is not present in the PTS SYSTEM entry, SMP does not pass any parameters to the PDS copy and load program. If you specify a COPYPARM subentry, ensure that the parameters are valid for your substitute PDS copy and load program or IEBCOPY.

COPYPRINT(ddname)

specifies the COPYPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the PDS copy and load program.

Note: If the COPYPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the COPYPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the PDS copy and load program.

COPYRC(value)

specifies the COPYRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the PDS copy and load program. When the value returned is higher than the COPYRC subentry value, then the result of the PDS copy or load function is considered unsuccessful and the SYSMOD for which the PDS copy and load program was invoked is terminated. The value may be any number from 0 to 16.

See OS/VS Utilities for a description of the IEBCOPY return codes.

Note: If the COPYRC subentry is not present in the PTS SYSTEM entry, then the default value of 0 is compared with the PDS copy and load program return code.

Note: IEBCOPY returns a code of 4 when it encounters I/O errors during the copying of members.

DSPREFIX(prefix)

specifies the DSPREFIX subentry of the PTS SYSTEM entry, where "prefix" is the high level qualifier data set name of data sets which are allocated during RECEIVE processing for library loading. "prefix" may have a maximum length of 26 characters. The value must conform to Operating System data set naming conventions. For example, "MYPREFIX.SET1.SYS1" is a valid prefix; "MYPREFIXSET1SYS1" is not. If the DSPREFIX subentry is not present, then no high order qualifier is used during allocation and subsequent accessing.

DSSPACE(prim,sec,dirblks)
specifies the DSSPACE subentry of the PTS SYSTEM entry, which contains space parameters for data sets that are allocated during RECEIVE processing for library loading. "prim" and "sec" are the primary and secondary allocation in tracks, and "dirblks" specifies the number of directory blocks to be allocated.

Note: This operand must be specified when the PTS SYSTEM entry is created.

FMID(sysmodid[,sysmodid]...)
specifies the FMID subentries of the PTS SYSTEM entry, where "sysmodid" is the SYSMOD-ID of a function SYSMOD. During RECEIVE processing, the SYSMODs in the PTFIN data set have their FMID operand values in the ++VER modification control statements compared with the FMID subentries to determine if the SYSMODs should be received.

IOSUPNAME(name)
specifies the IOSUPNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the IEHIOSUP function.

Note: If the IOSUPNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IEHIOSUP to perform the IEHIOSUP function.

IOSUPPARM(parm)
specifies the IOSUPPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the IEHIOSUP function. A maximum of 100 characters may be specified.

Note: If the IOSUPPARM subentry is not present in the PTS SYSTEM entry, SMP does not pass any parameters to the IEHIOSUP program. If you specify a IOSUPPARM subentry, ensure that the parameters are valid for your substitute IEHIOSUP program or IEHIOSUP.

IOSUPPRINT(ddname)
specifies the IOSUPPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the IEHIOSUP program.

Note: If the IOSUPPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the IOSUPPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the IEHIOSUP program.

IOSUPRC(value)
specifies the IOSUPRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the IEHIOSUP program. When the value returned is higher than the IOSUPRC subentry value, then the result of the IEHIOSUP function is considered unsuccessful and the SYSMOD for which the IEHIOSUP program was invoked is terminated. The value may be any number from 0 to 16.

See OS/VS Utilities for a description of the IEHIOSUP return codes.

Note: If the IOSUPRC subentry is not present in the PTS SYSTEM entry, then the default value of 0 is compared with the IEHIOSUP program return code.

LKEDNAME(name)
specifies the LKEDNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the linkage editor function.

Note: If the LKEDNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IEWL to perform the linkage editor function.

LKEDPARM(parm)
specifies the LKEDPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the linkage editor functions. A maximum of 100 characters may be specified.

Note: If the LKEDPARM subentry is not present in the PTS SYSTEM entry, SMP passes as parameters to the linkage editor program only those attributes specified as indicators in the LMOD entries. If the LKEDPARM subentry is present in the PTS SYSTEM entry, SMP passes as parameters to the linkage editor program the LKEDPARM subentry plus the attributes specified as indicators in the LMOD entries. The SIZE parameter may be specified in the LKEDPARM subentry. If you specify a LKEDPARM subentry, ensure that the parameters are valid for your substitute linkage editor program or IEWL.

LKEDPRINT(ddname)

specifies the LKEDPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the linkage editor program.

Note: If the LKEDPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the LKEDPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the linkage editor program.

LKEDRC(value)

specifies the LKEDRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the linkage editor program. When the value returned is higher than the LKEDRC subentry value, then the result of the linkage editor function is considered unsuccessful and the SYSMOD for which the linkage editor program was invoked is terminated. The value may be any number from 0 to 16.

See OS/VS Linkage Editor and Loader for a description of the linkage editor return codes.

Note: If the LKEDRC subentry is not present in the PTS SYSTEM entry, then the default value of 8 is compared with the linkage editor program return code.

Note: This operand may be specified for UCL operations on the SMPPTS only.

NUCID(n)

specifies the NUCID subentry of the ACDS or CDS SYSTEM entry, where "n" is a 1-digit number appended to the nucleus program name IEANUC0 to form the name of the nucleus load module saved during APPLY processing.

Note: This operand must be specified when adding the system entry to the CDS and ACDS. It may not be deleted. If an alternate NUCID is to be used, either alter the default ID via UCLIN (REP SYS NUCID(n)) or specify the alternate NUCID as an operand on the APPLY statement.

PAGELEN(nnnn)

specifies the PAGELEN subentry of the PTS SYSTEM entry, where "nnnn" is a number from 1 to 9999 that is used as the number of lines per page for the output listing in the SMPOUT data set. If this subentry is not present, the default number of lines per page is 60.

PEMAX(nnnn)
specifies the PEMAX subentry of the ACDS, CDS, or PTS SYSTEM entry, where "nnnn" is a number from 1 to 9999 that defines the maximum number of subentries that can be present in an entry on the respective data sets. If this subentry is not present in a SYSTEM entry, a default value of 500 is used for that SYSTEM entry. The value is used to calculate the buffer size needed in order to process the entries.

PURGE
specifies the PURGE indicator of the PTS SYSTEM entry. When this indicator is set, any SYSMOD that is successfully processed by ACCEPT is deleted from the PTS provided that the APPLY indicator is set in the SYSMOD entry on the PTS and NOAPPLY was not specified on the ACCEPT control statement.

Note: When the PTS SYSTEM entry is created, the PURGE indicator is set. To reset the indicator requires a second UCL statement specified as "DEL SYS PURGE.".

Note: When the PTS SYSTEM entry is listed, the PURGE option is shown as "YES" if the PURGE indicator is set and as "NO" if the PURGE indicator is reset.

REJECT
specifies the REJECT indicator of the PTS SYSTEM entry. When this indicator is set, any SYSMOD that is successfully processed by RESTORE is deleted from the PTS.

Note: When the PTS SYSTEM entry is created, the REJECT indicator is set. To reset the indicator requires a second UCL statement specified as "DEL SYS REJECT.".

Note: When the PTS SYSTEM entry is listed, the REJECT option is shown as "YES" if the REJECT indicator is set and as "NO" if the REJECT indicator is reset.

| RETRYNAME(name)
specifies the RETRYNAME subentry of the PTS system entry, where 'name' is the name of the program to be invoked by SMP4 to perform the recovery COMPRESS function before attemping a RETRY following a UTILITY failure.

<u>NOTE</u>: If the RETRYNAME subentry is not present in the PTS system entry SMP4 invokes the program IEBCOPY to perform the recovery COMPRESS function.

| RETRYPARM(parm)
specifies the RETRYPARM subentry of the PTS system entry, where 'parm' specifies values to be passed as parameters to the program invoked by SMP4 to perform the recovery COMPRESS function before attempting a RETRY following a utility failure. A maximum of 100 characters may be specified.

<u>NOTE</u>: If the RETRYNAME subentry is not present in the PTS system entry, SMP4 does not pass any parameters to the recovery COMPRESS program. If a RETRYPARM subentry is specified, ensure that the parameters are valid for the substitute recovery COMPRESS program or IEBCOPY.

| RETRYPRINT(ddname)
specifies the RETRYPRINT subentry of the PTS system entry, where 'ddname' is the DDNAME for the output listing data set produced by the recovery COMPRESS program.

<u>NOTE</u>: If the RETRYPRINT subentry is not present in the PTS system entry, then the ddname SYSPRINT is used as the default. A dd-statement specifying either SYSPRINT or the DDNAME in the RETRYPRINT subentry, when present, must be supplied when SMP4 is invoked to perform functions that may use the recovery COMPRESS program.

| RETRYRC(value)
specifies the RETRYRC subentry of the PTS system entry, where 'value' is the return code value to be compared with the code returned from the recovery COMPRESS program. When the value returned is higher than the RETRYRC subentry value, then the result of the recovery COMPRESS function is considered unsuccessful and the SMP retry is considered to have failed. In this case SMP is terminated. The 'value' may be any number from 0 to 16.

See 'OS/VS UTILITIES' (GC35-0005) for a description of the IEBCOPY return codes.

<u>NOTE</u>: If the RETRYRC subentry is not present in the PTS system entry, then the default value of 0 is compared with the recovery COMPRESS program return code.

| RETRYDDN(ALL | ddname[,ddname]. . .)

specifies the RETRYDDN subentry or subentries of the CDS or ACDS system entry, where 'ALL' causes RETRY to be attempted for utilities failures on any PDS target data set and where 'ddname' causes RETRY to be attempted for utility failures on the named PDS target data set.

NOTE: If a RETRYDDN subentry is not present in the CDS or ACDS system entry, then no RETRY will be attempted. If a RETRYDDN subentry of 'ALL' and one or more 'ddname' values exists, RETRY will be processed as if only 'ALL' were specified.

SAVEMTS

specifies the SAVEMTS indicator of the CDS SYSTEM entry. When this indicator is set, the macros in the MTS data set are not deleted by ACCEPT processing.

Note: When the CDS SYSTEM entry is created, if the SAVEMTS operand is not specified, the indicator is reset.

Note: This operand may be specified for the ACDS SYSTEM entry, but does not have any meaning and is only for compatibility with the CDS SYSTEM entry.

Note: When the CDS SYSTEM entry is listed, the SAVEMTS option is shown as "YES" if the SAVEMTS indicator is set and as "NO" if the SAVEMTS indicator is reset.

SAVESTS
specifies the SAVESTS indicator of the CDS SYSTEM entry. When this indicator is set, the modules in the STS data set are not deleted by ACCEPT processing.

Note: When the CDS SYSTEM entry is created, if the SAVESTS operand is not specified, the indicator is reset.

Note: This operand may be specified for the ACDS SYSTEM entry, but does not have any meaning and is only for compatibility with the CDS SYSTEM entry.

Note: When the CDS SYSTEM entry is listed, the SAVESTS option is shown as "YES" if the SAVESTS indicator is set and as "NO" if the SAVESTS indicator is reset.

SREL(cnnn[,cnnn]...)
specifies the SREL subentry of the ACDS, CDS, or PTS SYSTEM entry, where "cnnn" is a system release identifier. Only one "cnnn" value may be specified for UCL operations to the CDS or ACDS SYSTEM entry, whereas multiple "cnnn" values may be specified for operations to the PTS SYSTEM entry.

Note: When creating a SYSTEM entry, this operand must be specified. The SREL subentry cannot be deleted from the ACDS and CDS SYSTEM entries. At least one SREL subentry must be present in the PTS SYSTEM entry.

UPDATNAME(name)
specifies the UPDATNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the text update function.

Note: If the UPDATNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IEBUPDTE to perform the text update function.

UPDATPARM(parm)

specifies the UPDATPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the text update function. A maximum of 100 characters may be specified.

Note: If the UPDATPARM subentry is not present in the PTS SYSTEM entry, SMP passes the parameter "MOD" if the member in the output PDS exists and is being updated, or "REP" if the member does not exist or is being replaced. If the UPDATPARM subentry is present, then its value is appended to the "MOD" or "REP" parameter and passed to the text update program. If you specify a UPDATPARM subentry, ensure that the parameters are valid for your substitute text update program or IEBUPDTE.

UPDATPRINT(ddname)

specifies the UPDATPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the text update program.

Note: If the UPDATPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the UPDATPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the text update program.

UPDATRC(value)

specifies the UPDATRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the text update program. When the value returned is higher than the UPDATRC subentry value, then the result of the text update function is considered unsuccessful and the SYSMOD for which the text update program was invoked is terminated. The value may be any number from 0 to 16.

See OS/VS Utilities for a description of the IEBUPDTE return codes.

Note: If the UPDATRC subentry is not present in the PTS SYSTEM entry, then the default value of 0 is compared with the text update program return code.

ZAPNAME(name)

specifies the ZAPNAME subentry of the PTS SYSTEM entry, where "name" is the name of the program to be invoked by SMP to perform the IMASPZAP service aid function.

Note: If the ZAPNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IMASPZAP to perform the IMASPZAP function.

ZAPPARM(parm)
specifies the ZAPPARM subentry of the PTS SYSTEM entry, where "parm" specifies values to be passed as parameters to the program invoked by SMP to perform the IMASPZAP function. A maximum of 100 characters may be specified.

Note: If the ZAPPARM subentry is not present in the PTS SYSTEM entry, SMP does not pass any parameters to the IMASPZAP program. If you specify a ZAPPARM subentry, ensure that the parameters are valid for your substitute IMASPZAP program or IMASPZAP.

ZAPPRINT(ddname)
specifies the ZAPPRINT subentry of the PTS SYSTEM entry, where "ddname" is the ddname for the output listing data set produced by the IMASPZAP program.

Note: If the ZAPPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used as the default. A DD statement specifying either SYSPRINT or the ddname in the ZAPPRINT subentry, when present, must be supplied when SMP is invoked to perform functions that use the IMASPZAP program.

ZAPRC(value)
specifies the ZAPRC subentry of the PTS SYSTEM entry, where "value" is the return code value to be compared with the code returned from the IMASPZAP program. When the value returned is higher than the ZAPRC subentry value, then the result of the IMASPZAP function is considered unsuccessful and the SYSMOD for which the IMASPZAP program was invoked is terminated. The value may be any number from 0 to 16.

See OS/VS1 Service Aids or OS/VS2 System Programming Library: Service Aids for a description of the IMASPZAP return codes.

Note: If the ZAPRC subentry is not present in the PTS SYSTEM entry, then the default value of 4 is compared with the IMASPZAP program return code.

UCL SYSMOD Syntax

```
{ ADD | DEL | REP } SYSMOD(name)
   [APAR | FUNCTION | PTF | USERMOD]
   [ACCDATE(yyddd)]
   [ACCEPT]
   [ACCID(cdsid[,cdsid]...)]
   [ACCTIME(hh.mm.ss)
   [APPDATE(yyddd)]
   [APPID(cdsid[,cdsid]...)]
   [APPLY]
   [APPTIME(hh.mm.ss)
   [ASSEM(name[,name]...)]
   [BYPASS]
   [DELBY(sysmodid)]
   [DELETE(sysmodid[,sysmodid]...)]
   [ERROR]
   [FMID(sysmodid)]
   [JCLIN]
   [LASTSUP(sysmodid)]
   [LASTUPD(JCLIN|UCLIN)]
   [LASTUPDTYPE(ADD|UPD)]
   [MAC(name[,name]...)]
   [MACUPD(name[,name]...)]
   [MOD(name[,name]...)]
   [NPRE(sysmodid[,sysmodid]...)]
   [PRE(sysmodid[,sysmodid]...)]
   [RECDATE(yyddd)]
   [RECTIME(hh.mm.ss)]
   [REGEN]
   [REQ(sysmodid[,sysmodid]...)]
   [RESDATE(yyddd)]
   [RESTIME(hh.mm.ss)]
   [RESTORE]
   [RMAC(name[,name]...)]
   [RMACUPD(name[,name]...)]
   [RMOD(name[,name]...)]
   [RSRC(name[,name]...)]
   [RSRCUPD(name[,name]...)]
   [RSZAP(name[,name]...)]
   [RXZAP(name[,name]...)]
   [SRC(name[,name]...)]
   [SRCUPD(name[,name]...)]
   [SUPBY(sysmodid[,sysmodid]...)]
   [SUPING(sysmodid[,sysmodid]...)]
   [SZAP(name[,name]...)]
   [UPDTE(name[,name]...)]
   [UCLDATE(yyddd)]
   [UCLTIME(hh.mm.ss)]
   [VERNUM(value)]
   [VERSION(sysmodid[,sysmodid]...)]
   [XZAP(name[,name]...)]
   .
```

SYSMOD(name)
specifies a SYSMOD entry or subentries and indicators within an entry are to be added, deleted, or replaced, where "name" is the SYSMOD entry name corresponding to the SYSMOD-ID of a SYSMOD. SYSMOD entries exist on the ACDS, ACRQ, CDS, CRQ, PTS, and SCDS. For the SCDS, the only valid operation is DEL with no other operands. For the PTS, the valid operations are ADD, DEL, or REP of the ACCID and APPID indicators in a SYSMOD entry, and DEL of the SYSMOD entry. In the latter case, deleting the SYSMOD entry also causes SMP to delete the associated MCS entries.

APAR
specifies the APAR indicator of an ACDS or CDS SYSMOD entry. If this indicator is set in the SYSMOD entry, the SYSMOD is considered to be an APAR SYSMOD. If this operand is specified with the ADD operand and either the APAR, FUNCTION, PTF, or USERMOD indicator is set, then an error message is issued and the indicator is not set. If this operand is specified with the REP operand and either the FUNCTION, PTF, or USERMOD indicator is set, then that indicator is reset and the APAR indicator is set. If this operand is specified with the DEL operand and the APAR indicator is set, it is reset leaving the SYSMOD entry with no type characteristic.

FUNCTION
specifies the FUNCTION indicator of an ACDS or CDS SYSMOD entry. If this indicator is set in the SYSMOD entry, the SYSMOD is considered to be a FUNCTION SYSMOD. If this operand is specified with the ADD operand and either the APAR, FUNCTION, PTF, or USERMOD indicator is set, then an error message is issued and the indicator is not set. If this operand is specified with the REP operand and either the APAR, PTF, or USERMOD indicator is set, then that indicator is reset and the FUNCTION indicator is set. If this operand is specified with the DEL operand and the FUNCTION indicator is set, it is reset leaving the SYSMOD entry with no type characteristic.

PTF
specifies the PTF indicator of an ACDS or CDS SYSMOD entry. If this indicator is set in the SYSMOD entry, the SYSMOD is considered to be a PTF SYSMOD. If this operand is specified with the ADD operand and either the APAR, FUNCTION, PTF, or USERMOD indicator is set, then

an error message is issued and the indicator is not set. If this operand is specified with the REP operand and either the APAR, FUNCTION, or USERMOD indicator is set, then that indicator is reset and the PTF indicator is set. If this operand is specified with the DEL operand and the PTF indicator is set, it is reset leaving the SYSMOD entry with no type characteristic.

Note: This is the default if neither APAR, FUNCTION, PTF, nor USERMOD is specified when a SYSMOD entry is created as the result of an ADD operation.

USERMOD

specifies the USERMOD indicator of an ACDS or CDS SYSMOD entry. If this indicator is set in the SYSMOD entry, the SYSMOD is considered to be a USERMOD SYSMOD. If this operand is specified with the ADD operand and either the APAR, FUNCTION, PTF, or USERMOD indicator is set, then an error message is issued and the indicator is not set. If this operand is specified with the REP operand and either the APAR, FUNCTION, or PTF indicator is set, then that indicator is reset and the USERMOD indicator is set. If this operand is specified with the DEL operand and the USERMOD indicator is set, it is reset leaving the SYSMOD entry with no type characteristic.

ACCDATE(yyddd)

specifies the ACCDATE subentry of an ACDS or CDS SYSMOD entry, where "yyddd" is the Julian date that the SYSMOD was accepted. If the ACCDATE subentry is present in a SYSMOD entry, then the ACCEPT indicator is set. If the ACCDATE subentry is deleted, the ACCEPT indicator is reset.

Note: When creating a new entry on the ACDS, this operand must be specified if the SYSMOD entry is an ordinary type, that is, not superseded only. The ACCDATE subentry cannot be deleted from an ordinary SYSMOD entry on the ACDS.

ACCEPT

specifies the ACCEPT indicator of an ACDS or CDS SYSMOD entry. When this indicator is set, the SYSMOD is considered accepted. This operand can also be specified as "ACC" or "ACPT".

Note: The ACCEPT indicator reflects the presence or absence of the ACCDATE subentry. The ACCEPT operand need not be specified when the ACCDATE operand is specified since they are automatically synchronized.

The reason for inclusion of this operand is for compatibility with UCL statements processable by previous versions of SMP.

ACCID(cdsid[,cdsid]...)
specifies one or more ACCID subentries of a PTS SYSMOD entry, where "cdsid" is the CDS identifier from the CDSID subentry of an ACDS SYSTEM entry. Each ACCID subentry present in a SYSMOD entry indicates that the SYSMOD is considered accepted in the corresponding ACDS.

| ACCTIME(hh:mm:ss)
specifies the ACCTIME subentry of an ACDS or CDS SYSMOD entry, where HH:MM:SS are the hour, minute, and second that the SYSMOD was accepted. A semicolon must be specified between digits. If the ACCDATE is changed without a corresponding change to ACCTIME in the same UCL statement the ACCTIME is reset to 00:00:00. If ACCDATE is deleted then ACCTIME is deleted. If the ACCDATE is added to a SYSMOD but ACCDATE is not specified then ACCTIME is set to 00:00:00.

APPDATE(yyddd)
specifies the APPDATE subentry of an ACDS or CDS SYSMOD entry, where "yyddd" is the Julian date that the SYSMOD was applied. If the APPDATE subentry is present in a SYSMOD entry, then the APPLY indicator is set. If the APPDATE subentry is deleted, the APPLY indicator is reset.

Note: When creating a new entry on the CDS, this operand must be specified if the SYSMOD entry is an ordinary type, that is, not superseded only. The APPDATE subentry cannot be deleted from an ordinary SYSMOD entry on the CDS.

APPID(cdsid[,cdsid]...)
specifies one or more APPID subentries of a PTS SYSMOD entry, where "cdsid" is the CDS identifier from the CDSID subentry of a CDS SYSTEM entry. Each APPID subentry present in a SYSMOD entry indicates that the SYSMOD is considered applied in the corresponding CDS.

APPLY
specifies the APPLY indicator of an ACDS or CDS SYSMOD entry. When this indicator is set, the SYSMOD is considered accepted. This operand can also be specified as "APP" or "APPL".

Note: The APPLY indicator reflects the presence or absence of the APPDATE subentry. The APPLY operand need not be specified when the APPDATE operand is specified since they are automatically synchronized. The reason for inclusion of this operand is for downward compatibility with UCL statements processable by previous versions of SMP.

| APPTIME(hh:mm:ss)

specifies the APPTIME subentry of an ACDS or CDS SYSMOD entry, where HH:MM:SS are the hour, minute, and second that the SYSMOD was accepted. A semicolon must be specified between digits. If the APPDATE is changed without a corresponding change to APPTIME in the same UCL statement the APPTIME is reset to 00:00:00. If APPDATE is deleted then APPTIME is deleted. If the APPDATE is added to a SYSMOD but APPDATE is not specified then APPTIME is set to 00:00:00.

ASSEM(name[,name]...)
specifies one or more ASSEM subentries of an ACDS or CDS SYSMOD entry, "name" is the name of an ASSEM or SRC entry that was specified in the ASSEM operand list of a ++MAC, ++MACUPD, or ++UPDTE modification control statement of the SYSMOD.

BYPASS
specifies the BYPASS indicator of an ACDS or CDS SYSMOD entry. When this indicator is set, the SYSMOD is considered to have been processed only because one or more conditions that would have resulted in termination of processing for the SYSMOD were bypassed.

DELBY(sysmodid)
specifies the DELBY subentry of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that deleted this SYSMOD.

Note: This subentry is only valid for SYSMOD entries with the FUNCTION indicator.

DELETE(sysmodid[,sysmodid]...)
specifies one or more DELETE subentries of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is deleted by this SYSMOD. Each DELETE subentry present is considered to have been in the operand list of the DELETE operand of the processed ++VER modification control statement for the SYSMOD. The only other UCL operand that you can specify with DELETE is FUNCTION.

Note: DELETE subentries are considered invalid if the SYSMOD entry does not have the FUNCTION indicator set.

ERROR
specifies the ERROR indicator of an ACDS or CDS SYSMOD entry. This operand can also be specified as "ERR". When this indicator is set, the SYSMOD is considered to have been unsuccessfully processed.

FMID(sysmodid)
specifies the FMID subentry of an ACDS, ACRQ, CDS, or CRQ SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a function SYSMOD.

For ACDS and CDS SYSMOD entries, the FMID subentry is considered to be the FMID operand from the processed ++VER modification control statement for the SYSMOD or, for base level function SYSMODs, the SYSMOD-ID from the ++FUNCTION modification control statement.

Note: This operand is required when creating an ACDS or CDS SYSMOD entry that is not a superseded-only type.

For ACRQ and CRQ SYSMOD entries, the FMID subentry is considered to be the FMID operand from a ++IF modification control statement included with the SYSMOD. If the ADD or REP operand is specified, then the REQ operand must also be specified and must physically follow the FMID operand on UCL statement. If the DEL option is specified, then if the REQ operand is also specified, it is ignored. If the REP operand is specified and there is a matching FMID subentry in the SYSMOD entry being processed, then the SYSMOD-IDs specified in REQ operand replace the existing REQ subentries in the SYSMOD entry.

Note: The associated FMID entry on the ACRQ or CRQ should be updated to reflect changes made to a SYSMOD entry.

| JCLIN
indicates that the SYSMOD contain inline JCLIN.

LASTSUP(sysmodid)
specifies the LASTSUP subentry of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of the last SYSMOD which superseded this SYSMOD.

| LASTUPD(UCLIN|SYSMODID)
identifies the cause of the last change to this entry.

| LASTUPDTYPE(ADD|UPD)
identifies the last type of update made to this entry

MAC(name[,name]...)
specifies one or more MAC subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a macro replaced by this SYSMOD. Each MAC subentry is considered to be present because of the inclusion of a ++MAC modification control statement in the SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no RMAC, MACUPD, or RMACUPD subentries are present in the SYSMOD entry with the same names.

MACUPD(name[,name]...)
specifies one or more MACUPD subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a macro updated by this SYSMOD. Each MACUPD subentry is considered to be present because of the inclusion of a ++MACUPD or ++UPDTE modification control statement in the SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MAC, RMAC, or RMACUPD subentries are present in the SYSMOD entry with the same

names.

MOD(name[,name]...)
specifies one or more MOD subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a module replaced by this SYSMOD. Each MOD subentry is considered to be present because of the inclusion of a ++MOD modification control statement in the SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no RMOD, SZAP, RSZAP, XZAP, or RXZAP subentries are present in the SYSMOD entry with the same names.

NPRE(sysmodid[,sysmodid]...)
specifies one or more NPRE subentries of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is a negative prerequisite of this SYSMOD. Each NPRE subentry present is considered to have been in the operand list of the NPRE operand of the processed ++VER modification control statement for the SYSMOD.

Note: NPRE subentries are considered invalid if the SYSMOD entry does not have the FUNCTION indicator set.

PRE(sysmodid[,sysmodid]...)
specifies one or more PRE subentries of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is a prerequisite of this SYSMOD. Each PRE subentry present is considered to have been in the operand list of the PRE operand of the processed ++VER modification control statement for the SYSMOD.

RECDATE(yyddd)
specifies the RECDATE subentry of an ACDS or CDS SYSMOD entry, where "yyddd" is the Julian date that the SYSMOD was received.

Note: When creating a new entry, this operand must be specified if the SYSMOD entry is an ordinary type, that is, not superseded only. The RECDATE subentry cannot be deleted from an ordinary SYSMOD entry.

| RECTIME(hh:mm:ss)
specifies the RECTIME subentry of an ACDS or CDS SYSMOD entry, where HH:MM:SS are the hour, minute, and second that the SYSMOD was accepted. A semicolon must be specified between digets. If the RECDATE is changed without a corresponding change to RECTIME in the same

UCL statement the RECTIME is reset to 00:00:00. If RECDATE is deleted then RECTIME is deleted. If the RECDATE is added to a SYSMOD but RECDATE is not specified then RECTIME is set to 00:00:00.

REGEN
specifies the REGEN indicator of an ACDS or CDS SYSMOD entry. If this indicator is set, the SYSMOD is considered to have been in the ACDS prior to system generation and its associated elements updated in the distribution libraries. SMP does not use this indicator

to imply ACCEPT status. This operand can be specified as "RGN".

REQ(sysmodid[,sysmodid]...)
specifies one or more REQ subentries of an ACDS, ACRQ, CDS, or CRQ SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is a requisite of this SYSMOD.

For ACDS and CDS SYSMOD entries, each REQ subentry present is considered to have been in the operand list of the REQ operand of the processed ++VER modification control statement for the SYSMOD. entry being processed.

For ACRQ and CRQ SYSMOD entries, each REQ subentry present is considered to have been in the operand list of the REQ operand of a ++IF modification control statement included in the SYSMOD. When this operand is specified, the FMID operand must also be specified. For ADD operations, this operand is required. For DEL operations, this operand is ignored, if it is specified.

RESDATE(yyddd)
specifies the RESDATE subentry of a CDS SYSMOD entry, where "yyddd" is the Julian date that the SYSMOD was attempted to be restored. If the RESDATE subentry is present in a SYSMOD entry, then the RESTORE indicator is set. If the RESDATE subentry is deleted, the RESTORE indicator is reset. If the RESDATE subentry is added to a SYSMOD entry, the ERROR indicator is set.

| RESTIME(hh:mm:ss)
specifies the RESTIME subentry of an ACDS or CDS SYSMOD entry, where HH:MM:SS are the hour, minute, and second that the SYSMOD was accepted. A semicolon must be specified between digets. If the RESDATE is changed without a corresponding change to RESTIME in the same UCL statement the RESTIME is reset to 00:00:00. If RESDATE is deleted then RESTIME is deleted. If the RESDATE is added to a SYSMOD but RESDATE is not specified then RESTIME is set to 00:00:00.

RESTORE
specifies the RESTORE indicator of a CDS SYSMOD entry. If this indicator is set, the SYSMOD is considered to have had a RESTORE operation attempted. This operand can also be specified as "RES" or "REST".

Note: The RESTORE indicator reflects the presence or absence of the RESDATE subentry. The RESTORE operand need not be specified when the RESDATE operand is specified since they are automatically synchronized. The reason for inclusion of this operand is for compatibility with UCL statements processable by previous versions of SMP.

RMAC(name[,name]...)
specifies one or more RMAC subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a macro

replaced by this SYSMOD. Each RMAC subentry is considered to be present because of the inclusion of a ++MAC modification control statement in the SYSMOD that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated MAC entry may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MAC, MACUPD, or RMACUPD subentries are present in the SYSMOD entry with the same names.

RMACUPD(name[,name]...)
specifies one or more RMACUPD subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a macro updated by this SYSMOD. Each MACUPD subentry is considered to be present because of the inclusion of a ++MACUPD or ++UPDTE modification control statement in the SYSMOD that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated MAC entry may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MAC, MACUPD, or RMAC subentries are present in the SYSMOD entry with the same names.

RMOD(name[,name]...)
specifies one or more RMOD subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a module replaced by this SYSMOD. Each RMOD subentry is considered to be present because of the inclusion of a ++MOD modification control statement in the SYSMOD that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated MOD entry may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MOD, SZAP, RSZAP, XZAP, or RXZAP subentries are present in the SYSMOD entry with the same names.

RSRC(name[,name]...)
specifies one or more RSRC subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a source module replaced by this SYSMOD. Each RSRC subentry is considered to be present because of the inclusion of a ++SRC modification control statement in the SYSMOD that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated SRC entry

may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no SRC, SRCUPD, or RSRCUPD subentries are present in the SYSMOD entry with the same names.

RSRCUPD(name[,name]...)
specifies one or more RSRCUPD subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a source module updated by this SYSMOD. Each RSRCUPD subentry is considered to be present because of the inclusion of a ++SRCUPD modification control statement in the SYSMOD that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated SRC entry may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no SRC, SRCUPD, or RSRC subentries are present in the SYSMOD entry with the same names.

| UCLDATE(YYDDD)
specifies the UCLDATE subentry of an ACDS or CDS SYSMOD entry, where YYDDD is the JULIAN date that the SYSMOD was updated by UCLIN. If no UCLDATE is specified then the SMP data will be used.

| UCLTIME(hh:mm:ss)
specifies the UCLTIME subentry of an ACDS or CDS SYSMOD entry, where HH:MM:SS are the hour, minute, and second that the SYSMOD was accepted. A semicolon must be specified between digets. If the UCLDATE is changed without a corresponding change to UCLTIME in the same UCL statement the UCLTIME is reset to 00:00:00. If UCLDATE is deleted then UCLTIME is deleted. If the UCLDATE is added to a SYSMOD but UCLDATE is not specified then UCLTIME is set to 00:00:00.

| VERNUM(value)
specifies a 1 to 3 digit number of the ++VER statement which SMP used when processing the SYSMOD. This number is associated with those subentries that come from the ++VER statements, such as SUP and PRE. If VERNUM is not specified then any entries that are added or replaced by the UCL statement that require the VERNUM will assume a VERNUM of 0. No changes can be made to a SYSMOD that result in subentries with different VERNUM values. If subentries are added that require the VERNUM value and

VERNUM is specified then VERNUM must be specified before the other subentries.

RSZAP(name[,name]...)
specifies one or more RSZAP subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a module updated by this SYSMOD. Each RSZAP subentry is considered to be present because of the inclusion of a ++ZAP modification control statement in the SYSMOD without an EXPAND statement that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated MOD entry may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MOD, RMOD, SZAP, XZAP, or RXZAP subentries are present in the SYSMOD entry with the same names.

RXZAP(name[,name]...)
specifies one or more RXZAP subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a module updated by this SYSMOD. Each RXZAP subentry is considered to be present because of the inclusion of a ++ZAP modification control statement in the SYSMOD with an EXPAND statement that was regressed by the subsequent processing of another SYSMOD. The RMID subentry of the associated MOD entry may contain the SYSMOD-ID of the regressing SYSMOD.

Note: If this operand is specified with the ADD or REP

operand, you must ensure that no MOD, RMOD, XZAP, SZAP, or RSZAP subentries are present in the SYSMOD entry with the same names.

SRC(name[,name]...)
specifies one or more SRC subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a source module replaced by this SYSMOD. Each SRC subentry is considered to be present because of the inclusion of a ++SRC modification control statement in the SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no RSRC, SRCUPD, or RSRCUPD subentries are present in the SYSMOD entry with the same names.

SRCUPD(name[,name]...)
specifies one or more SRCUPD subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a source module updated by this SYSMOD. Each SRCUPD subentry is considered to be present because of the inclusion of a ++SRCUPD modification control statement in the SYSMOD.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no SRC, RSRC, or RSRCUPD subentries are present in the SYSMOD entry with the same names.

SUPBY(sysmodid[,sysmodid]...)
specifies one or more SUPBY subentries of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that supersedes this SYSMOD. This operand can also be specified as "SUP".

Note: The SUPBY subentry cannot be deleted from a superseded-only SYSMOD entry. A superseded-only SYSMOD entry is one created during APPLY or ACCEPT processing for a superseded SYSMOD that was never applied or accepted. A superseded-only SYSMOD entry can be created with a UCL SYSMOD statement that contains only the SYSMOD and SUPBY operands.

SUPING(sysmodid[,sysmodid]...)
specifies one or more SUPING subentries of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is superseded by this SYSMOD. Each SUPING subentry present is considered to have been in the operand list of the SUP operand of the processed ++VER modification control statement for the SYSMOD.

SZAP(name[,name]...)

specifies one or more SZAP subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a module updated by this SYSMOD. Each SZAP subentry is considered to be present because of the inclusion of a ++ZAP modification control statement in the SYSMOD without an EXPAND statement.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MOD, RMOD, XZAP, RXZAP, or RSZAP subentries are present in the SYSMOD entry with the same names.

UPDTE(name[,name]...)

specifies one or more MACUPD subentries of an ACDS or CDS SYSMOD entry.

Note: This operand is equivalent to the MACUPD operand and is included for compatibility with UCL statements processable by previous versions of SMP.

VERSION(sysmodid[,sysmodid]...)

specifies one or more VERSION subentries of an ACDS or CDS SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a function SYSMOD that is considered to have inferior versions of identically named elements with those present in the SYSMOD. Each VERSION subentry present is considered to have been in the operand list of the VERSION operand of the processed ++VER modification control statement for the SYSMOD.

XZAP(name[,name]...)

specifies one or more XZAP subentries of an ACDS or CDS SYSMOD entry, where "name" is the name of a module updated by this SYSMOD. Each XZAP subentry is considered to be present because of the inclusion of a ++ZAP modification control statement in the SYSMOD with an EXPAND statement.

Note: If this operand is specified with the ADD or REP operand, you must ensure that no MOD, RMOD, SZAP, RSZAP, or RXZAP subentries are present in the SYSMOD entry with the same names.

UCL Return Codes

See 'UCLIN Return Codes' under 'The UCLIN Control Statement'

The SMPADDIN Control Statement

The control statements provide the user with the ability to change the UCLIN statements produced by UNLOAD for certain fields in selected entries. Any data provided in the SMPADDIN control statements overrides the corresponding data in the CDS or ACDS entry. The UCLIN will then be generated using the data from the SMPADDIN statement. The format of the SMPADDIN statements is similiar to that of the UCLIN statements, however, only a limited number of fields are supported. All keywords on the SMPADDIN statements are optional.

SMPADDIN Syntax

```
REP MAC(macname) FMID(sysmodid)

   macname - MACRO name
   sysmodid - FMID to be generated for macro by UNLOAD

REP MOD(modname) FMID(sysmodid)

   modname - MODULE name
   sysmodid - FMID to be generated for macro by UNLOAD

REP SRC(srcname) FMID(sysmodid)

   srcname - SRC name
   sysmodid - FMID to be generated for macro by UNLOAD

REP SYS[CDSID(name] [SREL(cnnn)]

   name - CDSID to be generated by UNLOAD
   cnnn - SREL to be generated by UNLOAD

REP SYSMOD(sysmodid) [FMID(fmid)]

   [FUNCTION | PTF | APAR | USERMOD]
   [NEWNAME(newsmid)]

   sysmodid - SYSMOD entry to be changed
   fmid - new FMID for SYSMOD entry

   FUNCTION | PTF | APAR | USERMOD the type of SYSMOD entry
   to be generated.
```

NEWSMID - New SYSMOD entry name for sysmodid. The sysmod entry name should be changed only by IBM SUPPLIED SMPADDIN to cause a Product's SU or PTF like names to be changed to SMP4 format SU names.

SMPADDIN DDnames

SMPADDIN
contains the SMPADDIN control statements.

The UNLOAD Control Statement

The UNLOAD control statement will dump either the CDS or ACDS to the SMPPUNCH dataset. This function enables the user to unload all or selected parts of a CDS or ACDS to UCLIN format control statements. SMP can then be used to recreate the unloaded datasets.

UNLOAD Program Considerations

Since the volume of output produced by the UNLOAD function will be large the SMPPUNCH DD statement should be directed to either a direct access dataset or to tape. In addition the SMPPUNCH DD statement should specify the DCB parameter with a BLKSIZE that is a multiple of 80. The larger the BLKSIZE the less I/O operations SMP will perform.

UNLAOD Syntax

[UNLOAD CDS|ACDS](options)

Specify either ACDS or CDS but not both. No default is assumed. Options may be specified to limit the number of elements UNLOADED. Options are not required and if specified are not enclosed in parenthesis. The parenthesis are here to indicated optionality. The UNLOAD statement is terminated by a period.

UNLOAD Operands

ACDS
: specifies that all or selected information from the ACDS is to be unloaded.

CDS
: specifies that all or selected information from the CDS is to be unloaded.

OPTIONS
: specifies the options that you need for the ACDS, and CDS operand. For the syntax and explanations of the options, see the descriptions that follow for each data set type.

UNLOAD ACDS Syntax

```
UNLOAD ACDS

[ADDINPUT]

[MAC[(macname[,macname]...)]]

[MOD[(modname[,modname]...)]]

[SRC[(srcname[,srcname]...)]]

[SYSMOD[(sysmodid[,sysmodid]...)]

   [APAR] [DELETE]  [ERROR] [FUNCTION]  [NOAPPLY] [NOSUP]
   [PTF] [SUP] [USERMOD]

[SYS]

[UCLINDIS (READ|WRITE|NO)]

.
```

UNLOAD ACDS Operands

ADDINPUT
: directs SMP to read the set of control statements present in the dataset specified by the SMPADDIN DD statement. These control statements contain data that SMP will merge with that present in the dataset being UNLOADED and produce appropriate UCLIN control statements. The data specified in the SMPADDIN control statements overide the data present in the dataset being UNLOADED. See SMPADDIN control statement description for further information.

MAC[(macname[,macname]...)]
: specifies that information for all MAC entries or the specified MAC entries is to be unloaded. This information includes:

FMID
: the SYSMOD-ID of the owning function SYSMOD.

RMID
: the SYSMOD-ID of the last SYSMOD that replaced the macro.

UMID
: a list of the SYSMOD-IDs for SYSMODs that updated the macro.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID or 'UCLIN' and the last type of update made to the entry.

GENASM
a list of the ASSEM and SRC entries that are reassembled when this macro is changed.

MOD[(modname[,modname]...)]
specifies that information for all MOD entries or the specified MOD entries is to be unloaded. This information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the module.

UMID
a list of SYSMOD-IDs for the SYSMODs that updated the module.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID or 'UCLIN' and the last type of update made to the entry.

LMODS
a list of the load modules that include the module.

SRC[(srcname[,srcname]...)]
specifies that information for all SRC entries or the specified SRC entries is to be unloaded. This information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the source module.

UMID
a list of the SYSMOD-IDs for SYSMODs that updated the source module.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID or 'UCLIN' and the last type of update made to the entry.

MACROS
a list of the MAC entries with a GENASM subentry for the source module. This information is produced only when you specify the XREF keyword.

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that information for all SYSMOD entries or the specified SYSMOD entries is to be unloaded. This information includes:

TYPE
the type of SYSMOD ('APAR', 'FUNCTION', 'PTF', 'USERMOD', or 'SUPERSEDED').

FMID
the SYSMOD-ID from the ++VER or ++FUNCTION modification control statement.

JCLIN
an indicator that there is inline JCLIN within the SYSMOD.

STATUS

'BYP' if the SYSMOD was accepted using the BYPASS keyword
'ERR' if the SYSMOD was not successfully accepted
'REC' if the SYSMOD was received
'APP' if the SYSMOD was applied
'ACC' if the SYSMOD was accepted
'RGN' if the SYSMOD was accepted.

DATE/TIME
the date and time stamps for RECEIVE, ACCEPT, and UCLIN processing for the SYSMOD.

LASTSUP
the last SYSMOD processed that superseded this SYSMOD.

SREL, DELETE, PRE, NPRE, REQ, SUP, and VERSION
the contents of the keyword lists from the ++VER modification control statement used by ACCEPT processing.

MAC, MACUPD, MOD, SRC, SRCUPD, SZAP, and XZAP
the names from element modification control statements included in the SYSMOD.

RMAC, RMACUPD, RMOD, RSRC, RSRCUPD, RSZAP, and RXZAP
the names from element modification control statements included in the SYSMOD that represent regressed modifications. A regression occurs when a subsequent SYSMOD did not specify this SYSMOD in the PRE or SUP operand of its ++VER modification control statement.

ASSEM
the names of modules to be assembled as a result of macro or source changes contained in a SYSMOD.

SUPBY
a list of SYSMODs that supersede this SYSMOD; that is, SUP is specified in their ++VER modification control statements.

DELBY
a SYSMOD that deletes this SYSMOD; that is, DELETE is specified in its ++VER modification control statement.

APAR
specifies that APAR SYSMODs are to be unloaded.

DELETE
specifies that function SYSMODs that have been deleted from the CDS by other function SYSMODs are to be unloaded. This operand can be abbreviated as 'DEL'.

ERROR
specifies that SYSMODs that have the ERROR indicator set are to be unloaded. This operand can be abbreviated as 'ERR'.

FUNCTION
specifies that all function SYSMODs are to be unloaded. This operand can be abbreviated as 'FUNC'.

NOAPPLY
specifies that SYSMODs that have been received and accepted, but not applied are to be unloaded. Both the CDS and the ACDS data sets must be available when NOAPPLY is coded. A SYSMOD is considered applied when the SYSMOD entry exists on the CDS with the ERROR status indicator set off. This operand can be abbreviated as 'NOAPP'.

NOSUP
specifies that only SYSMODs that have not been superseded are to be unloaded.

This operand is mutually exclusive with the SUP operand. Specification of both causes a syntax error.

PTF
specifies that all PTF SYSMODs are to be unloaded.

SUP
specifies that only superseded SYSMODs are to be unloaded.

This operand is mutually exclusive with the NOSUP operand. Specification of both causes a syntax error.

USERMOD
specifies that all USERMOD type SYSMODs are to be unloaded. This operand can be abbreviated as 'USER'.

SYS
specifies that system information, such as the default NUCID, system type and release, and the identifier of the ACDS, is to be unloaded.

UCLINDIS(READ|WRITE|NO)
indicates to SMP the DIS option to generate on the UCLIN statement produced.

UNLOAD CDS Syntax

```
UNLOAD CDS

[ADDINPUT]

[ASSEM[(asmname[,asmname]...)]]

[DLIB[(dlibname[,dlibname]...)]]

[LMOD[(modname[,modname]...)]]

[MAC[(macname[,macname]...)]]

[MOD[(modname[,modname]...)]]

[SRC[(srcname[,srcname]...)]]

[SYSMOD[(sysmodid[,sysmodid]...)]
```

[APAR] [DELETE] [ERROR] [FUNCTION] [NOACCEPT] [NOSUP]
[PTF] [SUP] [RESTORE] [USERMOD]]

[SYS]

[UCLINDIS(READ|WRITE|NO)] •

UNLOAD CDS Operands

CDS
specifies that all or selected information from the CDS is to be unloaded. The list of ASSEM, DLIB, MOD, SRC, and SYSMOD is optional. However if specified for one type then a list must be specified for all types.

ADDINPUT
directs SMP to read the set of control statements present in the data set specified by the SMPADDIN DD statement. These control statements contain data that SMP will merge with the data present with the data set being UNLOADED and produce appropriate UCLIN control statements. The data specified in the SMPADDIN control statements overide the data present in the data set being UNLOADED. See SMPADDIN control statement description for further information.

ASSEM[(asmname[,asmname]...)]
specifies that information for all ASSEM entries or the specified ASSEM entries is to be unloaded. This includes:

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

ASSEMBLER INPUT
the contents of each text card in the entry.

DLIB[(dlibname[,dlibname]...)]
specifies that information for all DLIB entries or the specified DLIB entries are to be unloaded. The information includes:

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

SYSTEM LIBRARY
the names of the target system libraries.

LMOD[(modname[,modname]...)]
specifies that information for all LMOD entries or the specified LMOD entries are to be unloaded. The information includes:

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

SYSTEM LIBRARY
the names of the target system libraries.

LKED ATTRIBUTES
the parameters used to link edit the load module.

LKED CONTROL
the linkage editor control cards for the load module.

MAC[(macname[,macname]...)]
specifies that information for all MAC entries or the specified MAC entries are to be unloaded. The information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the macro.

UMID
a list of the SYSMOD-IDs for the SYSMODs that updated the macro.

DISTLIB
the distribution library name.

SYSLIB
the target system library name.

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

GENASM
a list of the ASSEM and SRC entries that are reassembled when this macro is changed.

MOD[(modname[,modname]...)]
specifies that information for all MOD entries or the specified MOD entries are to be unloaded. The information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the module.

UMID
a list of the SYSMOD-IDs for the SYSMODs that updated the module.

DISTLIB
the distribution library name.

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

LMODS
a list of the load modules that include the module.

SRC[(srcname[,srcname]...)]
specifies that information for all SRC entries or the specified SRC entries are to be unloaded. The information includes:

FMID
the SYSMOD-ID of the owning function SYSMOD.

RMID
the SYSMOD-ID of the last SYSMOD that replaced the source module.

UMID
a list of the SYSMOD-IDs for the SYSMOD that updated the source module.

DISTLIB
the distribution library name.

SYSLIB
the target system library name.

LAST UPDATE
the SYSMOD-ID, 'JCLIN' or 'UCLIN' and the last type of update made to the entry.

MACROS
a list of the MAC entries with a GENASM subentry for the source module. This information is produced only when you specify the XREF option.

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that information for all SYSMOD entries or the specified SYSMOD entries is to be unloaded. The options specified under SYSMOD are valid only if SYSMOD is specified. THE information includes:

TYPE
the type of SYSMOD ('APAR', 'FUNCTION', 'PTF', 'USERMOD', or 'SUPERSEDED').

FMID
the SYSMOD-ID from the ++VER or ++FUNCTION modification control statement.

JCLIN
an indicator that there is inline JCLIN within the SYSMOD.

STATUS
a status indicator that contains one of the following:

'BYP' if the SYSMOD was applied using the BYPASS keyword
'ERR' if the SYSMOD was not successfully applied or restored
'RGN' if the SYSMOD entry was copied from the ACDS
'RES' if an attempt was made to restore the SYSMOD
'REC' if the SYSMOD is received
'APP' if the SYSMOD is applied
'ACC' if the SYSMOD is accepted.

DATE/TIME
the date and time stamps for RECEIVE, APPLY, ACCEPT, UCLIN, and RESTORE processing for this SYSMOD.

LASTSUP
the last SYSMOD processed that superseded this SYSMOD.

SREL, DELETE, NPRE, PRE, REQ, SUP, and VERSION
the contents of the keyword lists from the ++VER modification control statement used during APPLY processing.

MAC, MACUPD, MOD, SRC, SRCUPD, SZAP, and XZAP
the names from element modification control statements included in the SYSMOD.

RMAC, RMACUPD, RMOD, RSRC, RSRCUPD, RSZAP, and RXZAP
the names from element modification control statements included in SYSMODs that represent regressed modifications. A regression occurs when a subsequent SYSMOD did not specify this SYSMOD in the PRE or SUP operand list of its ++VER modification control statement.

ASSEM
the names of modules to be assembled as a result of macro or source changes contained in the SYSMOD.

SUPBY
a list of the SYSMODs that supersede this SYSMOD; that is, SUP was specified in their ++VER modification control statements.

DELBY
a SYSMOD that deletes this SYSMOD; that is, DELETE was specified in its ++VER modification control statement.

APAR
specifies that APAR SYSMODs are to be unloaded.

DELETE
specifies that function SYSMOD entries that have been deleted from the CDS by other function SYSMODs are to be unloaded. This operand can be abbreviated as 'DEL'.

ERROR
specifies that SYSMODs that have the ERROR indicator set are to be unloaded. This operand can be abbreviated as 'ERR'.

FUNCTION
specifies that all function SYSMODs are to be unloaded. This operand can be abbreviated as 'FUNC'.

NOACCEPT
specifies that SYSMODs that have been received and applied, but not accepted in the ACDS are to be unloaded. Both the CDS and the ACDS data sets must be available when NOACCEPT is coded. A SYSMOD is considered accepted if the SYSMOD entry exists on the ACDS with the ERROR status indicator set off. This operand can be abbreviated as 'NOACC'.

NOSUP
specifies that only SYSMODs that have not been superseded are to be unloaded.

Note: This operand is mutually exclusive with the SUP operand. Specification of both causes a syntax error.

PTF
specifies that all PTF SYSMODs are to be unloaded.

RESTORE
specifies that SYSMODs that have the RESTORE indicator set are to be unloaded. The ERROR indicator must also be on for this condition to be valid. This operand can be abbreviated as 'RES'.

SUP
specifies that only superseded SYSMODs are to be unloaded.

Note: This operand is mutually exclusive with the NOSUP operand. Specification of both is causes a syntax error.

USERMOD
specifies that all USERMOD SYSMODs are to be unloaded. This operand can be abbreviated as 'USER'.

SYS
specifies that system information, such as the default NUCID, system type and release, the SAVESTS and SAVEMTS indicators, and the identifier of the CDS, is to be unloaded.

UCLINDIS(READ|WRITE|NO)
indicates to SMP the DIS option to generate on the UCLIN statement produced.

Chapter 8: SMP Modification Control Statements

Modification Control Statements are the input definitions of elements to be added to, modified in, and deleted from the target system and distribution libraries, and the information necessary to ensure that the environment of the target system and distribution libraries meets the required functional and service levels. The SMPPTFIN data set is used to contain the modification control statements.

This chapter describes the format and use of these statements. The SMP modification control statements are described in the following alphabetical order:

- ++APAR (temporary corrective fix)
- ++FUNCTION (new or replacement function)
- ++IF (conditional action)
- ++JCLIN (JCL input data)
- ++MAC (macro replacement)
- ++MACUPD/++UPDTE (macro update)
- ++MOD (module replacement)
- ++PTF (permanent corrective fix)
- ++SRC (source module replacement)
- ++SRCUPD (source update)
- ++USERMOD (user modification of IBM software)
- ++VER (for verification of environment)
- ++ZAP (module update)

Each modification control statement is described in the following format:

Introduction and Description: The name of the modification control statement is stated followed by a brief description of the function performed by the statement.

Syntax: The syntax of the modification control statement is given. See 'Appendix A: Rules for Coding SMP Statements' and 'Appendix B: Syntax Notation Conventions'

Note: The syntax in this chapter uses a comma to separate values in an operand list. One or more blanks can be used instead of a comma.

Operands: The function of each operand that can be coded with the modification control statement is described.

Programming Considerations: Any special considerations and notes applicable to the modification control statement are stated.

Examples: At least one coding example of the modification control statement is illustrated.

SYSMOD Construction

The modification control statements are used to construct a SYSMOD. The following is a summary of the rules pertaining to each modification control statement.

- ++APAR, ++FUNCTION, ++PTF, ++USERMOD

  These are referred to as header modification control statements; one of them is required for each SYSMOD. It must be the first modification control statement in the SYSMOD. All other modification control statements for the SYSMOD follow.

- ++VER

  At least one ++VER modification control statement must be present for a SYSMOD, and a maximum of 255 ++VER modification control statements are permitted.

- ++IF

  ++IF modification control statements, if specified, are associated with the ++VER modification control statement preceding them in the SYSMOD. Multiple ++IF modification control statements can be specified following each ++VER modification control statement.

- ++JCLIN

  There can be only one ++JCLIN modification control statement for each SYSMOD. It appears anywhere after the ++VER and ++IF modification control statements.

- ++MAC, ++MOD, ++SRC, ++MACUPD, ++SRCUPD, ++UPDTE, ++ZAP

  These modification control statement describe the elements being modified within a SYSMOD. They are referred to as element modification control statement, and at least one of them must be present in a SYSMOD.

  Some combinations of modifications to an element are invalid in the same SYSMOD.

  Figure 33 illustrates the combinations of modifications to an element that are either valid or invalid in the same SYSMOD.

	MOD	ZAP	SRC	SRCUPD	MAC	MACUPD/ UPDTE
MOD	INV	INV	VALID	VALID	VALID	VALID
ZAP	INV	INV	INV	INV	VALID	VALID
SRC	VALID	INV	INV	INV	VALID	VALID
SRCUPD	VALID	INV	INV	INV	VALID	VALID
MAC	VALID	VALID	VALID	VALID	INV	INV
MACUPD/ UPDTE	VALID	VALID	VALID	VALID	INV	INV

Figure 33. Valid Modifications to the Same Element

The APAR (++APAR) Modification Control Statement

The ++APAR modification control statement identifies a service SYSMOD. This type of modification is considered a temporary corrective fix to the elements of target system and distribution libraries. All other modification control statements for this SYSMOD follow this header modification control statement.

APAR Syntax

```
++APAR(sysmodid)
      [FILES(number)]
      .
```

APAR Operands

++ must be in columns 1 and 2

sysmodid
: specifies a unique seven-character system modification identifier which names the APAR system modification.

FILES(number)
: specifies the number of files belonging to the APAR SYSMOD that are unloaded partitioned data sets on a tape or set of tapes. The maximum number is 9999. The files must be on standard labelled tapes. Members of these files can be elements, JCL input data, or non-SMP data. When this operand is specified, the RELFILE keyword is required on those ++JCLIN, ++MAC, ++MOD, and ++SRC modification control statements that have their associated member in an unloaded PDS. At least one element or ++JCLIN modification control statement must have the RELFILE operand specified.

APAR Programming Considerations

1) During APPLY and ACCEPT processing, the sysmodid is placed in the MAC, MOD, and SRC entries in the CDS and ACDS, respectively, as RMID or UMID subentries. The Programming Considerations for the element modification control statement describe the updates of the CDS and ACDS entries.

2) An APAR SYSMOD is accepted into the distribution libraries only when the APARS keyword is specified on the ACCEPT control statement.

3) When you specify the FILES operand, the SMPTLIB DD statement is required during RECEIVE, REJECT, APPLY, RESTORE, and ACCEPT processing.

APAR Example

A temporary fix to module IEFJSSOB is needed to answer APAR OX12345 on an OS/VS1 system. The module must be at the service level provided by PTF UX11223 for function UX65300.

```
++APAR(AX12345).
++VER(X067) FMID(UX65300) PRE(UX11223).
++ZAP(IEFJSSOB) DISTLIB(AOS33).
  IMASPZAP Control Statements
```

The FUNCTION (++FUNCTION) Modification Control Statement

The ++FUNCTION modification control statement identifies a function SYSMOD. This type of modification introduces new or replacement function into target system and distribution libraries. All other modification control statements follow this header modification control statement.

FUNCTION Syntax

```
++FUNCTION(sysmodid)
     [FILES(number)]
     .
```

FUNCTION Operands

++ must be in columns 1 and 2

sysmodid
: specifies a unique seven character system modification identifier that names the function system modification.

FILES(number)
: specifies that the number of files belonging to this function are unloaded partitioned data sets on a tape or set of tapes. The maximum number is 9999. The files must be on standard labelled tapes. Members of these files can be elements, JCL input data, or non-SMP data. When this operand is specified, the RELFILE keyword is required on those ++JCLIN, ++MAC, ++MOD, and ++SRC modification control statements that have their associated member in an unloaded PDS. At least one element or ++JCLIN modification control statement must have the RELFILE operand specified.

FUNCTION Programming Considerations

1) During APPLY and ACCEPT processing, the SYSMOD-ID is placed in the MAC, MOD, and/or SRC entries in the CDS and ACDS, respectively, as FMID and RMID subentries. The Programming Considerations for each element modification control statement describe the updates to the CDS and ACDS entries.

2) ++MACUPD, ++UPDTE, ++SRCUPD, and ++ZAP modification control statements are not allowed in function SYSMOD packages.

3) When you specify the FILES operand, the SMPTLIB DD statement is required during RECEIVE, REJECT, APPLY, RESTORE, and ACCEPT processing.

FUNCTION Example

A function SYSMOD is to be created with a SYSMOD-ID of FIM1501 that is dependent on function GIM1500. The elements and JCL input data are members of three unloaded partitioned data sets on a tape created using the relative file technique.

```
++FUNCTION(FIM1501) FILES(3).
++VER(Z038) FMID(GIM1500).
++JCLIN RELFILE(1).
++MOD(IMQFGHI1) RELFILE(2) DISTLIB(AOS55).
++MOD(IMQPQRS3) RELFILE(2) DISTLIB(AOS55).
++MOD(IMQCVT) RELFILE(3) DISTLIB(AIMQMACS).
```

The Conditional Action (++IF) Modification Control Statement

The ++IF modification control statement describes actions to be taken when the condition described is satisfied during APPLY and ACCEPT processing of the SYSMOD that includes the ++IF modification control statement. The condition might also be satisfied during subsequent APPLY and ACCEPT processing, in which case the action is taken at that time. The purpose of conditional action specifications is to ensure that, when the functional environment of target system and distribution libraries changes, the correct function and/or service is also changed for elements of the system indirectly affected by the environment change.

++IF modification control statements are interpreted, reformatted and placed in the CRQ data set during APPLY processing and the ACRQ data set during ACCEPT processing. They are deleted from the CRQ and ACRQ during APPLY and ACCEPT processing when the associated SYSMOD is deleted or during RESTORE processing when the associated SYSMOD is successfully processed.

++IF modification control statements are associated with the ++VER modification control statement preceding it in the SYSMOD. Multiple ++IF modification control statements can be specified following each ++VER modification control statement.

IF Syntax

```
++IF FMID(sysmodid)
      [THEN]
      REQ(sysmodid[,sysmodid]...)
      .
```

IF Operands

++ must be in columns 1 and 2

FMID(sysmodid)
: specifies, as a condition, the SYSMOD-ID of a function SYSMOD that must be either installed or in the process of being installed on the target system by APPLY processing or on the distribution libraries by ACCEPT processing in order for the action portion of the ++IF modification control statement to be processed.

THEN
specifies that the action operand of the ++IF modification control statement follows.

REQ(sysmodid[,sysmodid]...)
specifies, as an action, one or more SYSMODs that are requisites of the SYSMOD containing the ++IF modification control statement. If the function SYSMOD specified in the FMID operand is applied to the target system or accepted into the distribution libraries, then the requisite SYSMODs must be applied or accepted with this SYSMOD or when the function SYSMOD is processed.

IF Programming Considerations

1) The operands must be specified in the order shown in the syntax.

2) Neither the SYSMOD-ID in the FMID operand of the associated ++VER modification control statement nor the SYSMOD-ID in the header modification control statement can be specified as the value for the FMID operand.

IF Example

PTF UZ00004 contains service to elements that belong to function FIM1501. If function FIM1509 has been applied or is in the process of being applied, the requisite PTF UZ00005 must be applied at the same time as PTF UZ00004 or have already been applied. If function FIM1509 is not presently applied, then PTF UZ00005 is not required, but the ++IF modification control statement is saved by SMP to be used if function FIM1509 is processed at a future time. When function FIM1509 is applied, PTF UZ00005 is considered to be an unsatisified conditional requisite that must be applied concurrently, if it has not already been applied.

```
++PTF(UZ00004).
++VER(Z038)  FMID(FIM1501).
++IF FMID(FIM1509) THEN REQ(UZ00005).
++MOD(IMQGHFI1) DISTLIB(AOS55).
++MACUPD(IMQCVT) DISTLIB(AIMQMACS).
```

The Job Control Language (++JCLIN) Modification Control Statement

The ++JCLIN modification control statement describes the job control language input data for a SYSMOD. Only one ++JCLIN modification control statement is allowed for a SYSMOD and it must be placed after all ++VER and ++IF modification control statements.

JCLIN Syntax

```
++JCLIN
      [ASM({PGM=name | procname})]
      [COPY({PGM=name | procname})]
      [LKED({PGM=name | procname})]
      [RELFILE(number) | TXLIB(ddname)]
      [UPDATE({PGM=name | procname})]
      .
```

JCLIN Operands

++ must be in columns 1 and 2

ASM({PGM=name | procname})
specifies the name of the assembler program or procedure that is used in the JCL data. This operand must be specified if the name is different from those recognized by SMP, which are the program names ASMBLR, IEUASM, and IFOX00, and procedure name ASMS.

COPY({PGM=name | procname})
specifies the name of the copy program or procedure that is used in the JCL data. This operand must be specified if the name is different from that recognized by SMP, which is the program name IEBCOPY.

LKED({PGM=name | procname})
specifies the name of the link edit program or procedure that is used in the JCL data. This operand must be specified if the name is different from those recognized by SMP, which are the program names HEWL and IEWL, and procedure name LINKS.

RELFILE(number)
specifies the relative position of the file containing the JCL data within the files associated with this SYSMOD. The file that contains the JCL data as one of its members must be an unloaded partitioned data set

that is physically located on the same tape or set of tapes as the file containing the SYSMOD to which this modification control statement belongs.

When RELFILE is specified, the FILES keyword must be specified on the header modification control statement.

The data set name is formed from the RELFILE operand as 'id#.Fnumber', where 'id#' is the SYSMOD-ID from the SYSMOD header modification control statement. The operand 'number' is a decimal number greater than or equal to one (1) with no leading zeroes; the maximum number allowed is 9999. The member of the data set that contains the JCL input data is identified by the SYSMOD-ID, such as UZ01234.

Note: This keyword is optional and mutually exclusive with TXLIB.

TXLIB(ddname)
specifies the ddname of a library that contains the JCL input data for the SYSMOD. The member of the library that contains the JCL input data is identified by the SYSMOD-ID, such as UZ01234.

Note: This keyword is optional and mutually exclusive with RELFILE.

UPDATE({PGM=name | procname})
specifies the name of the update program or procedure that is used in the JCL data. This operand must be specified if the name is different from that recognized by SMP, which is the program name IEBUPDTE.

JCLIN Programming Considerations

1) If the JCL input data is in the SMPPTFIN data set input stream, it must immediately follow the ++JCLIN modification control statement and must not contain any records that have the characters "++" in positions 1 and 2.

2) Processing the JCL data can be avoided by specifying the NOJCLIN operand on the APPLY control statement.

3) See the Programming Considerations of the JCLIN Control Statement in Chapter 7 for examples of JCL input data.

JCLIN Example

For function FIM1501, the JCL input data, an object module, ICGPRINT, and a macro, IEFJSSOB, are located in a separate text library named LIB1501. The JCL data contains an assembler program named ALTASM.

```
++FUNCTION (FIM1501).
++VER(Z038) FMID(HIM1500).
++JCLIN ASM(PGM=ALTASM) TXLIB(LIB1501).
++MOD(ICGPRINT) DISTLIB(AOS21) TXLIB(LIB1501).
++MAC(IEFJSSOB) DISTLIB(AMACLIB) TXLIB(LIB1501).
```

The following statement is needed at APPLY/ACCEPT time:

```
//LIB1501  DD  DSN=SYS1.LIB1501......
```

The Macro (++MAC) Modification Control Statement

The ++MAC modification control statement describes a single macro replacement within a SYSMOD. It must immediately precede the macro definition statements when they are in the SMPPTFIN data set input stream.

MAC Syntax

```
++MAC(name)
      [ASSEM(name[,name]...)]
      [BASE(FIXED | UPDATE)]
      [DELETE]
      [DISTLIB(ddname)]
      [DISTMOD(ddname) | DISTOBJ(ddname)]
      [DISTSRC(ddname) | ASMLIB(ddname)]
      [MALIAS(alias[,alias]...)]
      [RELFILE(number) | TXLIB(ddname)]
      [RMID(sysmodid)]
      [SSI(code)]
      [SYSLIB(ddname)]
      [VERSION(sysmodid[,sysmodid]...)]
      .
```

MAC Operands

++ must be in columns 1 and 2

(name)
: specifies the name of the macro member in the distribution library and, optionally, in the target system library. The name can contain any alphanumeric characters and '?', '$', '#', and '@'.

ASSEM(name[,name]...)
: specifies the names of the additional assembly or source modules to be assembled with this SYSMOD. The modules specified must reside in the library specified in the DISTSRC or ASMLIB keyword, or in the CDS.

  Note: APPLY and ACCEPT processing place the specified names into the SYSMOD entry created on the CDS and ACDS.

BASE(FIXED | UPDATE)
: not supported but included for compatibility with SYSMODs that can be processed by previous versions of

SMP.

DELETE
specifies that this macro is to be removed from target libraries, distribution libraries, and SMP control data sets.

Note: This keyword is mutually exclusive with all other keywords except DISTLIB, MALIAS and VERSION. If any other keywords are specified, a syntax error results.

DISTLIB(ddname)
specifies the ddname of the distribution library.

Note: This keyword must be specified if the macro has not been previously recorded on the CDS or ACDS data sets. If the entry does exist in the data sets, the value specified is compared with the DISTLIB subentry and, if they are not the same, the SYSMOD is not processed by APPLY and/or ACCEPT.

DISTMOD(ddname) | DISTOBJ(ddname)
specifies the ddname of the link edit distribution library for those modules specified in the ASSEM keyword. The object code from the assembler is link edited, during ACCEPT processing, to the library specified.

Note: Either DISTMOD or DISTOBJ can be specified, but not both. DISTMOD is preferred because it is more descriptive.

DISTSRC(ddname) | ASMLIB(ddname)
specifies the ddname of the library that contains the additional assembly or source modules to be assembled. The additional assembly or source modules must be specified in the ASSEM keyword.

Note: Either DISTSRC or ASMLIB can be specified, but not both. DISTSRC is preferred because it is more descriptive.

MALIAS(alias[alias]...)
specifies the alias names for the macro in both the target system and distribution libraries.

RELFILE(number)
specifies the relative position of the file containing the macro within the files associated with this SYSMOD. The file that contains the macros as one of it members must be an unloaded partitioned data set that is physically located on the same tape or set of tapes as the file containing the SYSMOD to which this modification control statement belongs.

When RELFILE is specified, the FILES keyword must be specified on the header modification control statement.

The data set name is formed from the RELFILE operand as 'id#.Fnumber', where 'id#' is the SYSMOD-ID from the SYSMOD header modification control statement, and 'number' is a decimal number greater than or equal to one with no leading zeroes; the maximum number allowed is 9999.

<u>Note</u>: This keyword is optional and mutually exclusive with TXLIB.

RMID(sysmodid)
: specifies the service SYSMOD that supplied this version of the macro as a replacement for the previous version of the macro.

This keyword is required on those elements changed in a service updated function SYSMOD and is only valid with function SYSMODs.

When specified, the RMID value in the MAC in the CDS or ACDS is set to the SYSMOD-ID specified in the RMID operand if the macro is selected for APPLY or ACCEPT processing.

SSI(code)
: specifies eight hexadecimal digits of system status information. This information is placed in the directory of the target system library or the MTS during APPLY processing and the distribution library during ACCEPT processing as four packed hexadecimal bytes of user data. See the IEBUPDTE program description in the OS/VS Utilities manual.

<u>Note</u>: This keyword is ignored if text is located in a library, which is the case when either the RELFILE or TXLIB keyword is specified.

SYSLIB(ddname)
: specifies the ddname of the target system library, if the macro exists in one. APPLY and RESTORE processing update this library.

TXLIB(ddname)
: specifies that the macro is not included in the SMPPTFIN input file but resides in the library pointed to by the specified ddname.

<u>Note</u>: This keyword is mutually exclusive with the RELFILE keyword.

VERSION(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs whose function is supported by this version of the macro. This version of the macro is superior to the version(s) of the macro found in each of the SYSMODs listed in the operand of the VERSION keyword.

When this parameter is specified it overrides any VERSION operand values that might be specified on the ++VER modification control statement.

MAC Programming Considerations

1) When inner macros, that is macros that are referred to by another macro instruction that resides in the macro library, are replaced, the modules that require reassembly must be specified in the ASSEM operand list.

2) If the macro replacement resides in a TXLIB partitioned data set instead of the SMPPTFIN data set, the TXLIB data set is required during SMP APPLY or ACCEPT processing for this macro.

3) If the SYSLIB keyword is specified or the distribution library containing the macro was totally copied at SYSGEN, the macro has not been stored in the SMPMTS. Therefore, you must specify the target library in the SYSLIB concatenation if assemblies are required. See 'SYSLIB Data Set' in Chapter 9 for a discussion of the SYSLIB concatenation.

4) Unless the distribution library specified in the DISTLIB operand was totally copied at SYSGEN time and the Stage I output was processed by the JCLIN function or the SYSLIB operand is specified, no target system library is updated. In this case, the SMPMTS data set will be used to hold the macro during APPLY processing.

MAC Example

The macro replacement for SGIECIOS does not follow in the input stream. The replacement resides in the text library SYS1.REPLACE.

```
++MAC(SGIECIOS) TXLIB(REPLACE).
```

In this example, the following statement is needed at APPLY and ACCEPT time:

```
//REPLACE   DD  DSN=SYS1.REPLACE....
```

Since the DISTLIB keyword was not specified, the MAC entry must exist on the CDS in order for APPLY processing to occur and on the ACDS in order for ACCEPT NOAPPLY processing to occur.

The Macro Update (++MACUPD/++UPDTE) Modification Control Statement

The ++MACUPD modification control statement describes a single macro update within a SYSMOD. It must immediately precede the macro update statements in the SMPPTFIN data set input stream. This statement may not appear in a function SYSMOD. For compatibility, ++MACUPD can be specified as ++UPDTE, but ++MACUPD is preferred because it is more descriptive.

MACUPD Syntax

```
++MACUPD(name) | ++UPDTE(name)
      [ASSEM(name[,name]...)]
      [BASE(FIXED | UPDATE)]
      [DISTLIB(ddname)]
      [DISTMOD(ddname) | DISTOBJ(ddname)]
      [DISTSRC(ddname) | ASMLIB(ddname)]
      [MALIAS(alias[,alias]...)]
      [SYSLIB(ddname)]
      [VERSION(sysmodid[,sysmodid]...)]
      .
```

MACUPD Operands

++MACUPD(name) | ++UPDTE(name)

Either ++MACUPD or ++UPDTE can be specified as the name of this modification control statement.

++ must be in columns 1 and 2

(name)
specifies the name of the macro member in the distribution library and, optionally, in the target system library. The name can contain any alphanumeric characters and '?', '$', '#', and '@'.

ASSEM(name[,name]...)
specifies the names of the additional assembly or source modules to be assembled with this SYSMOD. The modules specified must reside in the library specified in the DISTSRC or ASMLIB keyword, or in the CDS.

Note: APPLY and ACCEPT processing place the names specified into the SYSMOD entry created on the CDS and

ACDS.

BASE(FIXED | UPDATE)
not supported but included for compatibility with SYSMODs that can be processed by previous versions of SMP.

DISTLIB(ddname)
specifies the ddname of the distribution library.

Note: This keyword must be specified if the macro has not been previously recorded on the CDS or ACDS data sets. If the entry does exist in the data sets, the value specified is compared with the DISTLIB subentry and if it is not the same, the SYSMOD is not processed by APPLY and/or ACCEPT.

DISTMOD(ddname) | DISTOBJ(ddname)
specifies the ddname of the link edit distribution library for those modules specified in the ASSEM keyword. The object code from the assembler is link edited during ACCEPT processing to the library specified.

Note: Either DISTMOD or DISTOBJ can be specified, but not both. DISTMOD is preferred because it is more descriptive.

DISTSRC(ddname) | ASMLIB(ddname)
specifies the ddname of the library that contains the additional assembly or source modules to be assembled. The additional assembly or source modules must be specified in the ASSEM keyword.

Note: Either DISTSRC or ASMLIB can be specified, but not both. DISTSRC is preferred because it is more descriptive.

MALIAS(alias[,alias]...)
specifies the alias names for the macro for both the target system and distribution libraries.

SYSLIB(ddname)
specifies the ddname of the target system library if the macro exists in one. APPLY and RESTORE processing update this library.

VERSION(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs whose function is supported by this version of the macro. The version of the macro in this SYSMOD is superior to the version(s) of the macro to be found in each of the SYSMODs specified as values of the VERSION operand.

When this parameter is specified it overrides any VERSION operand values that might be specified on the ++VER modification control statement.

MACUPD Programming Considerations

1) When inner macros, that is macros that are referred to by another macro instruction that resides in the macro library, are replaced, the modules that require reassembly must be specified in the ASSEM operand list.

2) If the SYSLIB keyword is specified or the distribution library containing the macro was totally copied at SYSGEN, the macro has not been stored in the MTS. Therefore, the user must specify the target library in the SYSLIB concatenation if assemblies are required. See 'SYSLIB Data Set' in Chapter 9 for a discussion of the SYSLIB concatenation.

3) Unless either the distribution library specified in the DISTLIB operand was totally copied at SYSGEN time and the Stage I output was processed by the JCLIN function or the SYSLIB operand is specified, no target system library is updated. In this case, the MTS is used to hold the macro during APPLY processing.

MACUPD Example

The macro SEGMAC is in the AOSAA distribution library and is to be updated. The module IFKMYMOD must be reassembled when SEGMAC is modified. IFKMYMOD is a source module in the distribution library SYS1.AOS64 and a module in the distribution library SYS1.AOS23.

```
++MACUPD(SEGMAC) DISTLIB(AOSAA) ASSEM(IFKMYMOD)
           DISTSRC(AOS64) DISTMOD(AOS23).
```

In this example, the following statements are needed at ACCEPT time:

```
//AOSAA   DD   DSN=SYS1.AOSAA....
//AOS64   DD   DSN=SYS1.AOS64....
//AOS23   DD   DSN=SYS1.AOS23....
```

Furthermore, additional DD cards are needed for APPLY processing if the macro, source module, or module were copied at SYSGEN and DLIB entries exist for the libraries. For example, if the modules in SYS1.AOS23 were copied to SYS1.LINKLIB and the source modules in SYS1.AOS64 were copied to SYS1.CHGLIB, then the following DD cards are needed:

```
//LINKLIB  DD  DSN=SYS1.LINKLIB....
//CHGLIB  DD  DSN=SYS1.CHGLIB....
```

The Module (++MOD) Modification Control Statement

The ++MOD modification control statement describes a single module replacement within a SYSMOD. If the object code is in the SMPPTFIN data set input stream, the ++MOD modification control statement must immediately precede it.

MOD Syntax

```
++MOD(name)
      [DALIAS(alias)|TALIAS(alias[,alias]...)]
      [DELETE]
      [DISTLIB(ddname)]
      [LEPARM(leparm[,leparm]...)]
      [LKLIB(ddname)|TXLIB(ddname)|RELFILE(number)]
      [LMOD(name[,name]...)]
      [RMID(sysmodid)]
      [VERSION(sysmodid[,sysmodid]...)]
      •
```

MOD Operands

++ must be in columns 1 and 2

(name)
specifies the distribution library module name. The name can contain any alphanumeric characters and '?', '$', '#', and '@'.

DALIAS
specifies that the module has an alias only on a distribution library. The module might have been included under its alias name during system generation (SYSGEN).

DELETE
specifies that this module is to be removed from target libraries, distribution library, and SMP control data sets. If this is the only module in a load module, the LMOD entry is also removed from the CDS.

Note: This keyword is mutually exclusive with all other keywords except DALIAS, DISTLIB, TALIAS, and VERSION. If any other keywords are specified, a syntax error will result.

DISTLIB(ddname)
specifies the ddname of the distribution library.

Note: This keyword must be specified if the module has not been previously recorded on the CDS or ACDS. If an entry does exist in the CDS or ACDS, the value specified is compared with the DISTLIB subentry in the CDS or ACDS and, if it is not equal, the SYSMOD is not processed by APPLY or ACCEPT.

LEPARM(leparm[,leparm]..)
specifies linkage editor attributes for the module. Any of the following linkage editor parameters can be specified:

AC=1
ALIGN2
DC
NE
OVLY
REFR
RENT
REUS
SCTR
STD

Notes:

1. Refer to OS/VS Linkage Editor and Loader, GC26-3813, for a description of all parameters except STD.

2. STD can be used to indicate the SMP or user default set of linkage editor attributes. The SMP default set is 'LET,LIST,NCAL,XREF'. The user default set is defined via the LKEDPARM subentry of the PTS SYSTEM entry.

3. If the module was copied at SYSGEN, and no link edit attributes exist in the CDS LMOD entry, the LEPARM parameters are set in the CDS load module entry.

4. If LEPARM is not specified, and the module does not exist in the distribution library, ACCEPT processing uses the linkage editor defaults from the PTS SYSTEM entry, plus DC, RENT, REUS, and REFR.

5. The LEPARM values do not override existing linkage editor attributes in the CDS.

LKLIB(ddname)
specifies that the module is not being included in the SMPPTFIN input file but is contained in link edited format in the library pointed to by the DD card indicated by "ddname".

Note: This keyword is mutually exclusive with the RELFILE and TXLIB keywords.

LMOD(name[,name]...)
specifies one or more load module names that contain the module. If any of the names specified are not already LMOD subentries of the MOD entry on the CDS, they are added as such during APPLY processing.

Note: If an LMOD entry does not exist for an LMOD subentry, it will not be created and when the MOD is to be link edited during APPLY processing, a warning message is issued and no link edit is performed for that load module.

RELFILE(number)
specifies the relative position of the file containing the module within the files associated with this SYSMOD. The file that contains the module as one of its members must be an unloaded partitioned data set that is physically located on the same tape or set of tapes as the file containing the SYSMOD to which this modification control statement belongs.

When RELFILE is specified, the FILES keyword must be specified on the header modification control statement.

The data set name is formed from the RELFILE operand as 'id#.Fnumber', where 'id#' is the SYSMOD-ID from the SYSMOD header modification control statement, and 'number' is a decimal number greater than or equal to one with no leading zeroes; the maximum number allowed is 9999.

Note: This keyword is mutually exclusive with the LKLIB and TXLIB keywords.

RMID(sysmodid)
specifies the service SYSMOD that supplied this version of the module as a replacement for the previous version of the module.

This keyword is required on those modules changed in a service updated function SYSMOD package and is only valid with function SYSMODs.

Note: When specified, the RMID value in the MOD entry in the CDS or ACDS is set to the SYSMOD-ID specified in the

RMID operand if the module is selected for APPLY and/or ACCEPT processing.

TALIAS(alias[,alias]..)
specifies one or more alias names of the module on both the target system and distribution libraries for modules copied at SYSGEN.

TXLIB(ddname)
specifies that the module is not included in the SMPPTFIN input file but resides in object form in the library pointed to by the specified ddname.

Note: This keyword is mutually exclusive with the RELFILE and LKLIB keywords.

VERSION(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs whose function is supported by this version of the module. This version of the module is superior to the version(s) of the module to be found in each of the SYSMODs in the operand list of the VERSION keyword.

Note: When this parameter is specified it overrides any VERSION operand values that might be specified on the ++VER modification control statement.

MOD Programming Considerations

1) If the module replacement resides in a TXLIB or LKLIB partitioned data set, the TXLIB or LKLIB data set is required during SMP APPLY or ACCEPT processing for this module.

2) If the RELFILE keyword is specified, then the SMPTLIB DD statement is required during RECEIVE, REJECT, APPLY, RESTORE, or ACCEPT processing processing of the SYSMOD.

3) If SMP is unable to associate a module with a load module, no target system libraries are updated at APPLY time and message HMA286 is issued to warn of this condition. This will not occur if the distribution library specified in the DISTLIB operand was totally copied at system generation to a target system library, the module was recognized by JCLIN processing to be part of one or more load modules, or the LMOD operand is specified on the

++MOD modification control statement.

MOD Example

The module IEEFRQOD is a new module that is to be placed in the distribution library SYS1.AOSAA and is to be link edited with the existing load module IEEFRQ in the target system library SYS1.LINKLIB.

```
++MOD(IEEFRQOD) DISTLIB(AOSAA) LMOD(IEEFRQ).
```

The following DD statement is needed at APPLY time:

```
//LINKLIB  DD  DSN=SYS1.LINKLIB....
```

The following DD statement is needed at ACCEPT time:

```
//AOSAA  DD  DSN=SYS1.AOSAA....
```

The Program Temporary Fix (++PTF) Modification Control Statement

The ++PTF modification control statement identifies a service SYSMOD. This type of modification replaces and/or updates elements of target system and distribution libraries. All other modification control statements for this SYSMOD follow this header modification control statement.

PTF Syntax

```
++PTF(sysmodid)
     [FILES(number)]
     .
```

PTF Operands

++ must be in columns 1 and 2

sysmodid
: specifies a unique seven character system modification identifier that names the service system modification.

FILES(number)
: specifies the number of files belonging to this ++PTF modification control statement. These files are unloaded partitioned data sets on a tape or set of tapes. The maximum number allowed is 9999. The files must be on standard labelled tapes. Members of these files can be elements, JCL input data, or non-SMP data. When this operand is specified, the RELFILE keyword is required on those ++JCLIN, ++MAC, ++MOD, and ++SRC modification control statements that have their associated member in an unloaded PDS. At least one element or ++JCLIN modification control statement must have the RELFILE operand specified.

PTF Programming Considerations

1) During APPLY and ACCEPT processing, the SYSMOD-ID is placed in the MAC, MOD, and/or SRC entries in the CDS and ACDS, respectively, as RMID or UMID subentries. The element modification control statements Programming Considerations describe the updates to their respective CDS and ACDS entries.

2) You should use the ++USERMOD modification control statement to create modifications to IBM components rather than the ++PTF modification control statement.

3) If the FILES operand is specified, the SMPTLIB DD statement is required during RECEIVE, REJECT, APPLY, RESTORE, and ACCEPT processing.

PTF Example

A PTF is required to update macro IEQCVT and module IEQJJP for function FQQ4100. The prerequisite service SYSMODs for the macro and module are PTFs UZ13424 and UZ13457, respectively. The APAR incident fixed by this PTF is OZ34892.

```
++PTF(UZ13528).
++VER(Z038) FMID(FQQ4100) PRE(UZ13424,UZ13457)
     SUP(AZ34892).
++MACUPD(IEQCVT) DISTLIB(AQQMACLB).
++MOD(IEQJJP) DISTLIB(AOS59).
```

The Source (++SRC) Modification Control Statement

The ++SRC modification control statement describes a single source module replacement within a SYSMOD. If the source code is in the SMPPTFIN data set input stream, the statement must immediately precede the source code.

SRC Syntax

```
++SRC(name)
     [BASE(FIXED | UPDATE)]
     [DELETE]
     [DISTLIB(ddname)]
     [DISTMOD(ddname) | DISTOBJ(ddname)]
     [RELFILE(number)]
     [RMID(sysmodid)]
     [SSI(code)]
     [SYSLIB(ddname)]
     [TXLIB(ddname)]
     [VERSION(sysmodid[,sysmodid]...)]
     .
```

SRC Operands

++ must be in columns 1 and 2

(name)
specifies the name of the source module replacement in the distribution library. The name can contain any alphanumeric characters and '?', '$', '#', and '@'.

BASE(FIXED | UPDATE)
not supported but included for compatibility with SYSMODs that can be processed by previous versions of SMP.

DELETE
specifies that this source module is to be removed from target libraries, distribution libraries, and SMP control data sets.

Note: This keyword is mutually exclusive with all other keywords except DISTLIB and VERSION. If any other keywords are specified, a syntax error results.

DISTLIB(ddname)
specifies the ddname of the distribution library for the source module.

Note: This keyword must be specified if the SRC entry has not been previously recorded on the CDS or ACDS. If the entry does exist in the CDS or ACDS, the ddname value specified is compared with the DISTLIB subentry of the SRC entry, and, if it is not equal, the SYSMOD is not processed by APPLY or ACCEPT.

DISTMOD(ddname) | DISTOBJ(ddname)
specifies the ddname of the link edit distribution library for object code produced from the assembly of the source code. During ACCEPT processing, the object code from the assembler is link edited to the library specified.

Note: Either DISTMOD or DISTOBJ can be specified, but not both. DISTMOD is preferred because it is more descriptive.

RELFILE(number)
specifies the relative position of the file containing the source module within the files associated with this SYSMOD. The file that contains the source module as one of its members is an unloaded partitioned data set that is physically located on the same tape or set of tapes as the file containing the SYSMOD to which this modification control statement belongs.

When RELFILE is specified, the FILES keyword must be specified on the header modification control statement.

The data set name is formed from the RELFILE operand as 'id#.Fnumber', where 'id#' is the SYSMOD-ID from the SYSMOD header modification control statement, and 'number' is a decimal number greater than or equal to one with no leading zeroes; the maximum number allowed is 9999.

Note: This keyword is optional and mutually exclusive with TXLIB.

RMID(sysmodid)
specifies the service SYSMOD that supplied this version of the source module as a replacement for the previous version of the source module.

This keyword is required on those source modules changed in a service updated function SYSMOD package and is only valid with function SYSMODs.

When specified,the RMID value in the source element entry in the CDS and/or ACDS is set to the SYSMOD-ID specified in the RMID operand, if the source element is selected for APPLY/ACCEPT processing.

SSI(code)
specifies eight hexadecimal digits of system status information. This information is placed in the directory of the target system library or the STS during APPLY processing and the distribution library during ACCEPT processing as four packed hexadecimal bytes of user data. See the IEBUPDTE program description in the OS/VS Utilities manual.

Note: This keyword is ignored if the text is located in a library; that is, the RELFILE or TXLIB keyword was specified.

SYSLIB(ddname)
specifies the ddname of the target system library if the source module exists in one. APPLY and RESTORE processing update this library.

TXLIB(ddname)
specifies that the source is not included in the SMPPTFIN input stream, but resides in the library pointed to by the specified ddname.

Note: This keyword is mutually exclusive with the RELFILE keyword.

VERSION(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs whose function is supported by this version of the source module. This version of the source module is superior to the version(s) of the source module found in each of the SYSMODs specified in the operand list of the VERSION keyword.

Note: When this parameter is specified, it overrides any VERSION operand values that might be specified on the ++VER modification control statement.

SRC Programming Considerations

Unless either the distribution library specified in the DISTLIB operand was totally copied at SYSGEN time and the Stage I output was processed by the JCLIN function, or the SYSLIB operand is specified, no target system library is updated. In this case, the STS is used to hold the source module during APPLY processing.

SRC Example

A replacement for the source module IEAIOSP is in an unloaded library referenced by the ddname REPLACE. The distribution library for the source module is SYS1.ASRCLIB and SYS1.AOS68 for the module.

```
++SRC(IEAIOSP) TXLIB(REPLACE) DISTLIB(ASRCLIB)
     DISTMOD(AOS68).
```

The following DD statement is needed at APPLY and ACCEPT time:

```
//REPLACE  DD  DSN=SYS1.REPLACE....
```

The Source Update (++SRCUPD) Modification Control Statement

The ++SRCUPD modification control statement describes a single set of source update statements within a SYSMOD. It must immediately precede the source update statements in the SMPPTFIN data set input stream. If it appears in a function SYSMOD, the SYSMOD is not received.

SRCUPD Syntax

```
++SRCUPD(name)
      [BASE(FIXED | UPDATE)]
      [DISTLIB(ddname)]
      [DISTMOD(ddname) | DISTOBJ(ddname)]
      [SYSLIB(ddname)]
      [VERSION(sysmodid[,sysmodid]...)]
      .
```

SRCUPD Operands

++ must be in columns 1 and 2

(name)
: specifies the name of the source member in the distribution library and, optionally, in the target system library. The name can contain any alphanumeric characters and ?, $, #, and @.

BASE(FIXED | UPDATE)
: not supported but included for compatibility with SYSMODs that can be processed by previous versions of SMP.

DISTLIB(ddname)
: specifies the ddname of the distribution library for the source module.

  Note: This keyword must be specified if the SRC entry has not been previously recorded on the CDS or ACDS. If the SRC entry does exist, the value specified is compared with the DISTLIB subentry and, if it is not equal, the SYSMOD is not processed by APPLY or ACCEPT.

DISTMOD(ddname) | DISTOBJ(ddname)
: specifies the ddname of the link edit distribution library for object code produced from the assembly of the source code. During ACCEPT processing, the object

code from the assembler is link edited to the library specified.

Note: Either DISTMOD or DISTOBJ can be specified, but not both. DISTMOD is preferred because it is more descriptive.

SYSLIB(ddname)
specifies the ddname of the target system library if the source module exists in one. APPLY and RESTORE processing update this library.

VERSION(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs whose function is supported by this version of the source module. The version of the source module with this update is superior to the version(s) of this source module found in each of the SYSMODs in the operand list of the VERSION keyword.

Note: When this parameter is specified it overrides any VERSION operand values that might be specified on the ++VER modification control statement.

SRCUPD Programming Considerations

Unless either the distribution library specified in the DISTLIB operand was totally copied at SYSGEN time and the Stage I output was processed by the JCLIN function or the SYSLIB operand is specified, no target system library is updated. In this case, the STS is used to contain the source module during APPLY processing.

SRCUPD Example

The source module IKJLKTD to be updated resides on the target system library SYS1.OPLIB1 and distribution library SYS1.AOS33.

```
++SRCUPD(IKJLKTD)  SYSLIB(OPLIB1) DISTLIB(AOS33).
```

The following DD statement is needed at APPLY time:

```
//OPLIB1  DD DSN=SYS1.OPLIB1....
```

The following DD statement is needed at ACCEPT time:

```
//AOS33  DD DSN=SYS1.AOS33....
```

The User Modification (++USERMOD) Modification Control Statement

The ++USERMOD modification control statement identifies a service SYSMOD. This type of modification is created by you to update your private libraries and to replace or update IBM elements in the target system and distribution libraries. All other modification control statements for this SYSMOD follow this header modification control statement.

USERMOD Syntax

```
++USERMOD(sysmodid)
     [FILES(number)]
     .
```

USERMOD Operands

++ must be in columns 1 and 2

sysmodid
: specifies a unique seven character system modification identifier that names the user supplied system modification.

FILES(number)
: specifies the number of files belonging to this USERMOD SYSMOD that are unloaded partitioned data sets on a tape or set of tapes. The maximum number allowed is 9999. The files must be on standard labelled tapes. Members of these files can be elements, JCL input data, or non-SMP data. When this operand is specified, the RELFILE keyword is required on those ++JCLIN, ++MAC, ++MOD, and ++SRC modification control statements that have their associated member in an unloaded PDS. At least one element or ++JCLIN modification control statement must have the RELFILE operand specified.

USERMOD Programming Considerations

1) You must define the 7-character SYSMOD-ID when you create your own modifications. By convention, IBM development and service organizations use the letters 'A' through 'K' and 'U' through 'Z' for their SYSMOD-IDs. Therefore you should not use these sets of letters. SMP is insensitive to the content of the system modification name, but an alphabetic first character might be required by some system utilities invoked by you.

2) During APPLY and ACCEPT processing, the SYSMOD-ID is placed in the MAC, MOD, and/or SRC entries in the CDS and ACDS, respectively, as RMID and/or UMID subentries. The element modification control statements Programming Considerations describe the updates to their respective CDS and ACDS entries.

3) Subsequent replacements to elements modified by your modification cannot occur unless you explicitly allow them, with one exception: a function SYSMOD can replace an element you have modified.

4) A user modification is only accepted into the distribution libraries if the USERMODS keyword is specified on the ACCEPT control statement.

5) When the FILES operand is specified, the SMPTLIB DD statement is required during RECEIVE, REJECT, APPLY, RESTORE, and ACCEPT processing.

USERMOD Example

A source module, IQQABC, which is owned by function SYSMOD FQQ5200, is modified by you. Your modification requires a service level provided PTF UZ15639 and you are only updating, rather than replacing, the source module. You have chosen a SYSMOD-ID of MY00005 for your modification.

```
++USERMOD(MY00005).
++VER(Z038) FMID(FQQ5200) PRE(UZ15639).
++SRCUPD(IQQABC) DISTLIB(AQQSRCLB).
```

The Verify (++VER) Modification Control Statement

The ++VER modification control statement describes the environment required to apply and accept the SYSMOD and the SYSMODs and APARs that are to be superseded if the SYSMOD is applied to the target system or accepted into the distribution libraries. SYSMODs applicable to more than one system or environment may have multiple ++VER modification control statements, one for each system and environment to which the modifications apply. At least one ++VER modification control statement must be present in a SYSMOD, and a maximum of 255 ++VER modification control statements are allowed for each SYSMOD.

VER Syntax

```
++VER(srel-id[,srel-id]...)
      [DELETE(sysmodid[,sysmodid]...)]
      [FMID(sysmodid)]
      [NPRE(sysmodid[,sysmodid]...)]
      [PRE(sysmodid[,sysmodid]...)]
      [REQ(sysmodid[,sysmodid]...)]
      [SUP(sysmodid[,sysmodid]...)]
      [VERSION(sysmodid[,sysmodid]...)]
      •
```

VER Operands

++ must be in columns 1 and 2

(srel-id[,srel-id]...)

specifies one or more system code and release levels in character strings of four bytes. These values are compared with the SREL subentries in the PTS, CDS, and ACDS during RECEIVE, APPLY, and ACCEPT processing, respectively. When no match is found for any of the values specified, the ++VER modification control statement is not applicable and, if it is the only ++VER modification control statement, the SYSMOD is not applicable.

For ++VER modification control statements that can be processed only by this version of SMP, the same srel-id cannot be specified in more than one ++VER modification control statement unless the FMID operand is present and their contents are different in each ++VER modification control statement.

DELETE(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs that are to be removed when this SYSMOD is processed by APPLY or ACCEPT. This operand is only valid when included with function system modification packages. Specifying this operand causes both the removal of the function system modification referenced and the removal of all function and service modifications that are related in hierarchies lower than the referenced SYSMOD-ID(s). During APPLY processing, these SYSMODs are removed from the CDS and their elements are removed from the target system libraries. During ACCEPT processing, these SYSMODs are removed from the ACDS and their elements are removed from the distribution libraries. This operand has no effect on RECEIVE eligibility.

SYSMODs specified in the DELETE operand do not have to be respecified in VERSION operands of ++VER, ++MAC, ++SRC, or ++MOD modification control statements.

FMID(sysmodid)
specifies, for a service SYSMOD, the function SYSMOD to which all of the elements in the service SYSMOD belong, or, for a function SYSMOD, a prerequisite function SYSMOD. The elements contained in a function SYSMOD belong to that function. This operand must be present for a service SYSMOD. Any service SYSMOD containing this operand is not received unless the PTS SYSTEM entry contains an FMID subentry corresponding to this operand. The SYSMOD is applied if the FMID refers to a SYSMOD that is applied or is being applied in the same APPLY pass. The SYSMOD is accepted if the FMID refers to a SYSMOD that is accepted or is being accepted in the same ACCEPT pass.

For ++VER modification control statements processable by this version of SMP, the same FMID value cannot be specified in more than one ++VER modification control statement unless the srel-ids are different for the entire set of ++VER modification control statements. If one ++VER modification control statement contains an FMID operand, then all others processable by this version of SMP must also contain an FMID operand.

NPRE(sysmodid[,sysmodid]...)
specifies one or more SYSMODs that cannot exist on the CDS during APPLY processing or the ACDS during ACCEPT processing for the SYSMOD to be applicable. If any of the SYSMODs in the list are present, the SYSMOD cannot be applied or accepted. This operand has no effect on RECEIVE eligibility.

For ++VER modification control statements processable by this version of SMP, this operand can only be specified if it is in a function SYSMOD

PRE(sysmodid[,sysmodid]...)
specifies one or more prerequisite SYSMOD-IDs. The indicated SYSMODs must have been applied without error or be applied in the same APPLY pass to allow the application of this SYSMOD. The indicated SYSMODs must have been accepted with out error or be accepted in the same ACCEPT pass to allow acceptance of this SYSMOD. This operand has no effect on RECEIVE eligibility.

REQ(sysmodid[,sysmodid]...)
specifies one or more SYSMODs that must be applied and accepted along with this SYSMOD. If any of the requisite SYSMODs are not present or eligible for processing at APPLY or ACCEPT time, or have not been previously applied or accepted, the SYSMOD is not processed. This operand has no effect on RECEIVE eligibility.

SUP(sysmodid[,sysmodid]...)
specifies one or more SYSMODs that are superseded by this SYSMOD and/or one or more APARs fixed in the element modifications supplied with this SYSMOD.

VERSION(sysmodid[,sysmodid]...)
specifies one or more function SYSMODs whose function is supported by the versions of the elements contained within this SYSMOD.

The same SYSMOD-ID cannot be specified more than once in the same operand or be present in more than one operand list in a single ++VER modification control statement except that a SYSMOD-ID that is specified in the VERSION operand list can also be specified in any one of the other operand lists with the exception of FMID.

VER Programming Considerations

1) The ++VER modification control statement must immediately follow the header modification control statement (that is, the ++APAR, ++FUNCTION, ++PTF, or ++USERMOD modification control statement). Additional ++VER modification control statements, if specified, must immediately follow the first ++VER and its ++IF modification control statements, if

any.

2) SYSMODs can be constructed that can be processed by previous versions as well as this version of SMP. For service SYSMODs, this construction requires at least two ++VER modification control statements, one processable by previous versions of SMP and the other processable by this version of SMP. The srel-ids in these ++VER modification control statements must be different to enable the SYSMOD to be processed correctly by the applicable version of SMP.

VER Examples

A PTF is needed to modify module ISSDEF in function UZ88700, which is applicable to OS/VS2 Releases 3.7 and 3.8. PTF UZ00364 is a prerequisite in both releases.

```
++PTF(UZ12345).
++VER(Z037) PRE(UZ00364,UZ88700).
++VER(Z038) PRE(UZ00364) FMID(UZ88700).
++MOD(ISSDEF) DISTLIB(AOS88).
```

The IMASPZAP (++ZAP) Modification Control Statement

The ++ZAP modification control statement describes a module update within a SYSMOD. It must precede the IMASPZAP statements in the SMPPTFIN data set input stream. This modification control statement may not appear in a function SYSMOD.

ZAP Syntax

```
++ZAP(name)
      [DALIAS(alias) | TALIAS(alias[,alias]...)]
      [DISTLIB(ddname)]
      .
```

ZAP Operands

++ must be in columns 1 and 2

(name)
specifies the distribution library module name. The name can contain any alphanumeric characters and '?', '$', '#', and '@'.

DALIAS
specifies that the module has an alias only on a distribution library. The module might have been included under its alias name during system generation.

DISTLIB(ddname)
specifies the ddname of the distribution library.

Note: This keyword must be specified if the MOD entry has not been previously recorded on the CDS or ACDS. If the MOD entry does exist, the value specified is compared with the DISTLIB subentry in the MOD entry and, if it is not equal, the SYSMOD is not processed by APPLY or ACCEPT.

TALIAS(alias[,alias]..)
specifies one or more alias names, both on the target system and distribution libraries, for modules copied during system generation.

ZAP Programming Considerations

1) An EXPAND control statement in linkage editor format can be placed within IMASPZAP input to allow lengthening of control sections. The EXPAND statement may be placed anywhere within the IMASPZAP input for the module to be expanded. Refer to the OS/VS Linkage Editor and Loader for the syntax and description of the EXPAND statement.

2) Any SETSSI statements placed in the input stream for expand type IMASPZAP processing must be in a form acceptable to both IMASPZAP and the linkage editor; that is, they must begin in column 2 or after. The SSI statements must follow the EXPAND statements.

3) Expand-type IMASPZAP processing cannot be performed against a non-editable (NE) module.

4) The 'name' operand of the ++ZAP modification control statement must be the same as the distribution library module name. The CSECT name operand of the IMASPZAP control statement must be the same as the load module's CSECT name. The module's CSECT name is usually the same as the distribution library name.

   "LIST CDS LMOD." produces a CDS listing of linkage editor control statements that might have changed the CSECT name of the member. A LINKEDIT MAP may be helpful in other cases where the names differ.

5) The NAME statement for ZAP may optionally be coded as follows:

```
NAME csect-name

  or

NAME lmod-name csect-name
```

   The coding of one operand assumes that operand to be a CSECT name for the module referenced in the ZAP statement. In this case, all load modules containing the module named in the ZAP statement are processed by IMASPZAP.

Two operands can be specified, in which case the second operand is assumed to be a CSECT name, as specified above. The first operand is assumed to be a valid load module containing the module named in the ZAP statement. In this case, only the indicated load module is processed by IMASPZAP.

6) Care must be taken when using IMASPZAP on an assembled module because the modification identifier is updated, but not the modification of any associated macros.

   A subsequent update of the associated macros results in a re-assembly of the module and loss of the IMASPZAP modification without detecting a mismatch between the SYSMOD-ID and the modification identifier.

   It is not recommended that you use IMASPZAP to modify assembled modules. An assembled module modified by IMASPZAP cannot cause updating of the distribution library during ACCEPT processing, therefore, RESTORE processing will not replace the module in the target system with the IMASPZAP modification present. To allow error-free application and backing-off in these cases, ++MACUPD modification control statements to the macros that create the assembler input should, but need not, be performed.

7) SMP processing does not save a back-up copy of the nucleus during APPLY processing when the nucleus is modified by a SYSMOD containing a non-expand-type IMASPZAP modification.

8) Since only one ZAP can be applied to a module in one APPLY pass, multiple ZAPs to a module require re-execution of APPLY for each ZAP.

ZAP Example

The module IQRMYMOD is to be changed via IMASPZAP. The module is owned by function SYSMOD IQR4310 and the current RMID subentry value in the MOD entry is UX32564. The module is in load module IQRMAIN. You are creating the modification and assigning a SYSMOD-ID of MY00006.

```
++USERMOD(MY00006).
++VER(X067) FMID(IQR4310) PRE(UX32564).
++ZAP(IQRMYMOD) DISTLIB(MYDLIB).
  NAME IQRMAIN IQRMYMOD
  VER 13F6 47E0A138
  REP 13F6 4770A14C
```

Chapter 9: SMP Data Sets

SMP requires a variety of data sets. The total number of data sets is determined by the types of functions being executed.

The data sets are described in the following order:

- Link and text library data sets
- SMPACDS (Alternate Control Data Set)
- SMPACRQ (Alternate Conditional Requisite Queue Data Set)
- SMPCDS (Control Data Set)
- SMPCNTL (Control Statement Input Data Set)
- SMPCRQ (Conditional Requisite Queue Data Set)
- SMPJCLIN (JCL Input Data Set)
- SMPLIST (LIST Output Data Set)
- SMPLOG (History Log Data Set)
- SMPMTS (Macro Temporary Store Data Set)
- SMPOUT (Message Output Data Set)
- SMPPTFIN (SYSMOD Input Data Set)
- SMPPTS (SYSMOD Temporary Store Data Set)
- SMPRPT (Report Output Data Set)
- SMPSCDS (Saved Control Data Set)
- SMPSTS (Source Temporary Store Data Set)
- SMPTLIB (temporary library)
- SMPWRK1, SMPWRK2, SMPWRK3, SMPWRK4, SMPWRK5 (work data sets)

- SYSLIB (macro library data set for assembler)
- SYSPRINT (output data set)
- SYSUT1, SYSUT2, SYSUT3 (temporary utility storage)
- Target and distribution library data sets

Each data set is described in the following format:

Ddname: The name required in the DD statement that is written for the data set.

Acronym: The character string commonly associated with the data set.

Data Set: The common name of the data set.

Device: The types of devices that can be used for the data set.

Information: Information about the data set, such as the contents, special information, and the type of structure used.

Data Sets Required

Figure 34 provides a summary of the data sets required by each SMP control statement. The following list explains the meaning of each number used in Figure 34:

1 - Required

2 - One for each different LKLIB operand value on ++MOD modification control statements, if any.

3 - One for each different TXLIB operand value element modification control statements, if any.

4 - Optional, and if not supplied, data is written to SMPOUT.

5 - Required unless the NOAPPLY keyword is specified on the ACCEPT control statement or the SAVEMTS or SAVESTS indicators in the CDS are set on.

6 - Required unless the NOAPPLY keyword is specified on the ACCEPT control statement.

7 - One required for each distribution library being updated.

8 - Required when any SYSMODs contain unloaded partitioned data sets that were loaded to temporary libraries during RECEIVE processing.

9 - Required if COMPRESS is specified.

10 - Required when modifications were loaded to temporary libraries during RECEIVE processing and the REJECT indicator in the PTS SYSTEM entry is on.

11 - One required for each target system library being updated.

12 - corresponding macro or source module target system library, the modification being applied is an update, and no copy of the macro or source module exists in the MTS or STS.

13 - One required for each library containing copies of the elements being restored.

14 - Required when this data set is requested on the SMP control statement.

15 - Required when this data set is requested on the SMP control statement or when it is required for the specified LIST options.

	ACCEPT	APPLY	JCLIN	LIST	LOG	RECEIVE	REJECT	RESETRC	RESTORE	UCLIN
lklib	2	2								
txlib	3	3								
SMPACDS	1			15					1	14
SMPACRQ	1			15						14
SMPCDS	6	1	1	15					1	14
SMPCNTL	1	1	1	1	1	1	1	1	1	1
SMPCRQ		1		15					1	14
SMPJCLIN			1							
SMPLIST				4						
SMPLOG	1	1	1	1	1	1	1	1	1	1
SMPMTS	5	1							1	14
SMPOUT	1	1	1	1	1	1	1	1	1	1
SMPPTFIN						1				
SMPPTS	1	1		15		1	1		1	14
SMPRPT	4	4				4			4	
SMPSCDS	6	1		15					1	14
SMPSTS	5	1							1	14
SMPTLIB	8	8				8	8		10	
SMPWRK1	1	1							1	
SMPWRK2	1	1							1	
SMPWRK3	1	1							1	
SMPWRK4	1	1							1	
SMPWRK5		1								
SYSLIB	1	1							1	
SYSPRINT	1	1				8	9		1	
SYSUT1	1	1				1	1		1	
SYSUT2	1	1				1	1		1	
SYSUT3	1	1				1	1		1	
distlib	7	12							13	
tgtlib		11							11	

The ENDUCL and UCL statements use the same data sets as the UCLIN control statement.
You can supply a substitute ddname for the SYSPRINT data set, which is the default. See 'The UCL SYS Operands' in Chapter 7 for information. A DD statement specifying either SYSPRINT or the substitute ddname must be supplied as described for the SYSPRINT data set above.

Figure 34. Data Set Requirements

Data Set Definitions

Link and Text Library Data Sets

Ddname: The ddname required is indicated by the TXLIB or LKLIB keyword on the element modification control statement. For example, the statement ++MOD(MODA) TXLIB(LIBX) would require a ddname of LIBX.

Acronym: None

Description: Link and text libraries

Attributes: PO

Device: Direct access only

Information: These libraries contain replacement modules, macros, or source modules for use with the ++MOD, ++MAC or ++SRC modification control statements, and JCL input data associated with the ++JCLIN modification control statement.

If the LKLIB or TXLIB keyword is specified on the ++MOD, ++MAC, or ++SRC modification control statement statement, it means that the data does not immediately follow the modification control statement in the input stream. The data must therefore be a member of the library specified by the LKLIB keyword, if the replacement is in link edited format, or the TXLIB keyword, if the replacement is in object or source format or is JCL input.

SMPACDS Data Set

Ddname: SMPACDS

Acronym: ACDS

Description: Alternate Control Data Set

| Attributes: PO; LRECL=80, BLKSIZE=multiple of 80, DISP=OLD

Device: Direct access only

Information: This data set contains information about the macros, modules, source modules and SYSMODs in the distribution libraries. The data in the ACDS is used by SMP to control the checking, inserting, or removing of modules, source modules and macro definitions in the distribution libraries.

A SYSTEM entry is required for any processing involving this data set. The SYSTEM entry is created by UCLIN processing and contains system information, such as the default NUCID, system type and release, and the identifier of the ACDS.

Unless the DIS(NO) option is specified on the ACCEPT or UCLIN control statements, SMP brings the ACDS directories into storage during ACCEPT or UCLIN processing.

The ACDS should reside on one of the DLIB volumes to ensure it would correspond to the DLIBs if the system were to be restored.

You should update the ACDS only through the use of the SMP UCLIN control statement. Its contents can be listed by using the LIST control statement. The ACDS directories may be brought into storage by SMP during LIST processing if enough storage is available.

SMPACRQ Data Set

Ddname: SMPACRQ

Acronym: ACRQ

Description: Alternate Conditional Requisite Queue Data Set

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct Access only

Information: This data set is used to hold parsed ++IF modification control statements for use by the ACCEPT function. The entries in the first part of the ACRQ are stored according to the SYSMOD-ID of the SYSMOD that contained the ++IF modification control statements, and are referred to as SYSMOD entries. They include the SYSMOD-IDs specified in the FMID and REQ operand values in the ++IF modification control statements.

The entries in the second part of the ACRQ are stored according to the SYSMOD-IDs specified in the FMID operand values in the ++IF modification control statements. They are referred to as FMID entries because they name the functional environment that must exist in order for the requisite SYSMODs to be accepted. The entries reference the SYSMOD entries that contained the ++IF modification control statements in which they were specified as FMID operand values.

ACRQ entries are created when a SYSMOD is successfully accepted. Deletion of ACRQ entries occurs when the associated SYSMOD is deleted as a result of a DELETE specification on the ++VER modification control statement of a function SYSMOD which is successfully processed by ACCEPT.

The ACRQ can be updated using the UCLIN control statement. Its contents can be listed using the LIST control statement.

SMPCDS Data Set

Ddname: SMPCDS

Acronym: CDS

Data Set: Control Data Set

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct access only

Information: This data set contains information about the macros, modules, assemblies, load modules, libraries copied at system generation time, source modules, and SYSMODs in the target system. The data in the CDS is used by SMP to control the checking, inserting, or removing of modules, source modules and macro definitions in the target system libraries.

A SYSTEM entry is required for any processing involving this data set. The SYSTEM entry is created by UCLIN processing and contains system information such as the default NUCID, system type and release, the SAVESTS and SAVEMTS indicators, and the identifier of the CDS.

The CDS is created from information collected from the Stage I output of system generation or similar output, and the ACDS, which, if there are entries on the ACDS, must be copied to the CDS when the CDS is created.

Once the CDS has been created, the SMP JCLIN function should be run to update the CDS entries so that SYSMODs can be checked and correctly processed by SMP. The JCLIN function should be run after each partial system generation.

The CDS is updated by SMP during APPLY or RESTORE, or by the user through the use of the JCLIN or UCLIN control statements or the ++JCLIN modification control statement. Updating of the CDS should be done only through the use of SMP.

Unless the DIS(NO) option is specified on the APPLY, RESTORE or UCLIN control statements, SMP brings the CDS directories into storage during APPLY, RESTORE, JCLIN, or UCLIN processing.

The contents of the CDS can be listed by using the LIST control statement. The CDS directories may be brought into storage by SMP during LIST processing if enough storage is available.

SMPCNTL Data Set

Ddname: SMPCNTL

Acronym: CNTL

Description: Control statement input

Attributes: Sequential; LRECL=80, BLKSIZE=multiple of 80

Device: Card, tape, direct access, or terminal device

Information: This data set contains the SMP control statements used to direct the execution of SMP functions.

SMPCRQ Data Set

Ddname: SMPCRQ

Acronym: CRQ

Description: Conditional Requisite Queue

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct Access only

Information: This data set is used to hold parsed ++IF modification control statements for use by APPLY processing. The entries in the first part of the CRQ are stored according to the SYSMOD-ID of the SYSMOD that contained the ++IF modification control statements. These entries are referred to as SYSMOD entries. They include the SYSMOD-IDs specified in the FMID and REQ operand values in the ++IF modification control statements.

The entries in the second part of the CRQ are stored according to the SYSMOD-IDs specified in the FMID operand values in the ++IF modification control statements. They are referred to as FMID entries because they name the functional environment that must exist in order for the requisite SYSMODs to be applied. The entries reference the SYSMOD entries that contained the ++IF modification control statements in which they were specified as FMID operand values.

CRQ entries are created when a SYSMOD is successfully applied. Deletion of CRQ entries occurs when the associated SYSMOD is processed by RESTORE and when the associated SYSMOD is deleted as a result of a DELETE specification on the ++VER modification control statement of a function SYSMOD which is successfully processed by APPLY.

The CRQ can be updated using the UCLIN control statement. Its contents can be listed using the LIST control statement.

SMPJCLIN Data Set

Ddname: SMPJCLIN

Acronym: JCLIN

Description: JCL Input Data Set

Attributes: Sequential; LRECL=80, BLKSIZE=multiple of 80

Device: Card, tape, direct access, or terminal device

Information: This data set contains the Stage I output from the most recent full or partial system generation (or other data in a similar format).

Information from this data set is used to update or create the CDS, or update or create entries on the CDS.

SMPLIST Data Set

Ddname: SMPLIST

Acronym: None

Description: LIST Output Data Set

Attributes: Sequential; BLKSIZE=multiple of 121, LRECL=121, RECFM=FBA

Device: SYSOUT, printer, direct access, tape, or terminal

Information: This data set contains all SMP LIST output when the SMPLIST DD card is present.

SMPLOG Data Set

Ddname: SMPLOG

Acronym: LOG

Description: History Log Data Set

Attributes: Sequential; RECFM=U, BLKSIZE=260, DISP=MOD

Device: Tape or direct access

Information: This data set contains a time-stamped record of events that occur during SMP processing. SMP automatically writes records to this data set. The user can write records to SMPLOG using the LOG control statement. The LIST control statement can be used to obtain a listing of all or selected portions of the information on the data set.

The SMPLOG also contains SMP messages that result from BLDL and STOW operations and any messages that would help in diagnosing and understanding the processing that SMP performs.

The SMPLOG should be updated only through the use of SMP.

DISP=MOD must be specified to maintain a cumulative history.

SMPMTS Data Set

Ddname: SMPMTS

Acronym: MTS

Description: Macro Temporary Store Data Set

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct access only

Information: This data set contains macros that do not reside in a target system library (that is, the SYSLIB keyword was not specified on the SMP modification control statements, and there is no SYSLIB information in the SMPCDS for that macro). The updated version of the macro is stored on the SMPMTS during APPLY processing. The data is used in APPLY, ACCEPT, and RESTORE processing.

Note: To ensure that the updated version of the macro is used for any assemblies done by SMP, SMPMTS must be the first data set in the SYSLIB concatenation. The block size must be equal to or greater than the block size of all the other system macro libraries used by the assembler.

SMPOUT Data Set

Ddname: SMPOUT

Acronym: None

Description: Message Output Data Set

Attributes: Sequential; RECFM=FBA, LRECL=121, BLKSIZE=multiple of 121

Device: SYSOUT, printer, direct access, tape, or terminal device

Information: This data set contains all SMP messages. If the SMPRPT DD card is not present, then SMPOUT also contains report output. If the SMPLIST DD card is not present, SMPOUT also contains LIST output.

SMPPTFIN Data Set

Ddname: SMPPTFIN

Acronym: PTFIN

Description: System Modification Input Data Set

Attributes: Sequential; LRECL=80, BLKSIZE=multiple of 80

Device: Card, tape, direct access, or terminal device

Information: This data set contains the system modifications to be processed.

SMPPTS Data Set

Ddname: SMPPTS

Acronym: PTS

Description: PTF Temporary Store Data Set

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct Access only

Information: This data set is used as temporary storage for SYSMODs. The name PTF Temporary Store is a carry-over from previous SMP releases when the name 'PTF' described all types of modifications.

Two entries are present for each SYSMOD received. The first is an exact copy of the SYSMOD as it was received and is called a Modification Control Statement (MCS) entry. The second entry is similar to a SYSMOD entry on the CDS and ACDS and is also called a SYSMOD entry. Each ++VER modification control statement is represented with its operand values as subentries (that is, PRE values become PRE subentries). Each element modification control statement has its type and element name represented as a subentry (that is, ZAP HMASMREC).

The SYSMOD data is deleted by REJECT processing or by ACCEPT processing when a SYSMOD that has been accepted during the process has also been applied and the PURGE indicator is set in the PTS SYSTEM entry.

The MCS entries can be printed or punched from the PTS using the IEBPTPCH utility program. The SYSMOD entries and the MCS entries can be listed using the LIST control statement.

A SYSTEM entry is required for any processing involving this data set. The SYSTEM entry is created by UCLIN processing and contains at least one system release (SREL) subentry and any number of function modification identifier (FMID) subentries. The subentry fields can be modified by UCLIN processing.

SMPRPT Data Set

Ddname: SMPRPT

Acronym: RPT

Description: Report Output Data Set

Attributes: Sequential; BLKSIZE=multiple of 121, LRECL=121, RECFM=FBA

Device: SYSOUT, printer, direct access, tape or terminal device

Information: This data set contains all SMP reports when the SMPRPT DD card is present.

SMPSCDS Data Set

Ddname: SMPSCDS

Acronym: SCDS

Description: Save Control Data Set

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct access only

Information: This data set contains back-up copies of CDS entries that are modified during APPLY processing when ++JCLIN modification control statements are present in SYSMODs. The back-up copies are used during RESTORE processing to return the required CDS entries to the state that they were in before APPLY processing.

The SCDS entries can be deleted using the UCLIN control statement. Its contents can be listed using the LIST control statement.

SMPSTS Data Set

Ddname: SMPSTS

Acronym: STS

Description: Source Temporary Store

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct access only

Information: This data set contains source modules that do not reside in a target system library (that is, no SYSLIB keyword was specified on the SMP modification control statements, and there is no SYSLIB information in the CDS for that source module). The updated version of the source module is stored on the SMPSTS during APPLY processing. The data set is used in APPLY, ACCEPT, and RESTORE processing, and is passed to the assembler as input.

SMPTLIB Data Set

Ddname: SMPTLIB

Acronym: TLIB

Description: Temporary Libraries

Attributes: Partitioned

Device: Direct access only

Information: The SMPTLIB ddname is used by SMP to access partitioned data sets used as temporary storage for unloaded partitioned data sets, contained on the SMPPTFIN tape, that are dynamically loaded during RECEIVE processing. The DD statement should specify at least one direct access storage device with sufficient space to enable RECEIVE processing to dynamically allocate storage for the libraries. Up to five volumes can be specified.

Temporary libraries are deleted in their entirety when their associated SYSMOD is deleted by REJECT, RESTORE, or ACCEPT processing.

SMPWRK1 Data Set

Ddname: SMPWRK1

Acronym: WRK1

Description: Work Data Set One

Attributes: Partitioned; LRECL=80; BLKSIZE=multiple of 80, DISP=(NEW,DELETE)

Device: Direct access only

Information: This data set is used as a temporary storage for input to the IEBUPDTE and IEBCOPY programs. Data is placed in this data set by SMP during APPLY and ACCEPT processing before invoking the utility. The source of the data is text following ++MAC, ++MACUPD, or ++UPDTE modification control statements on the SMPPTS. The data set is only needed for the duration of the SMP job step. The disposition of this data set should be specified as DISP=(NEW,DELETE) to minimize space loss problems. If you require that the data set be kept beyond the duration of the SMP job step, it is your responsibility to reclaim any space that might be required by subsequent invocations of SMP.

SMPWRK2 Data Set

Ddname: SMPWRK2

Acronym: WRK2

Description: Work Data Set Two

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80, DISP=(NEW,DELETE)

Device: Direct access only

Information: This data set is used as temporary storage for input to the IEBUPDTE or IEBCOPY program. Data is placed in this data set by SMP during APPLY and ACCEPT processing before invoking the utility. The source of the data is text following ++SRC and ++SRCUPD modification control statements on the PTS. The data set is only needed for the duration of the SMP job step. The disposition of this data set should be specified as DISP=(NEW,DELETE) to minimize space loss problems. If you require that the data set be kept beyond the duration of the SMP job step, it is your responsibility to reclaim any space which that be required by subsequent invocations of SMP.

SMPWRK3 Data Set

Ddname: SMPWRK3

Acronym: WRK3

Description: Work Data Set Three

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80 and maximum of 3200, DISP=(NEW,DELETE)

Device: Direct Access only

Information: This data set is used as temporary storage for input to the Linkage Editor and output from the assembler. The data consists of object modules. The data set is only needed for the duration of the SMP job step. The disposition of this data set should be specified as DISP=(NEW,DELETE) to minimize space loss problems. If you require that the data set be kept beyond the duration of the SMP job step, it is your responsibility to reclaim any space that might be required by subsequent invocations of SMP.

SMPWRK4 Data Set

Ddname: SMPWRK4

Acronym: WRK4

Description: Work Data Set Four

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80, and maximum of 3200, DISP=(NEW,DELETE)

Device: Direct access only

Information: This data set is used as temporary storage for input to the IMASPZAP utility program. The data consists of control cards that are text following ++ZAP modification control statements on the SMPPTS. The data set is only needed for the duration of the SMP job step. The disposition of this data set should be specified as DISP=(NEW,DELETE) to minimize space loss problems. If you require that the data set be kept beyond the duration of the SMP job step, it is your responsibility to reclaim any space that might be required by subsequent invocations of SMP.

SMPWRK5 Data Set

Ddname: SMPWRK5

Acronym: WRK5

Description: Work Data Set Five

Attributes: Partitioned; RECFM=U

Device: Direct access only

Information: This data set is used when modules that would be link edited to form new or replacement modules exist in more than one temporary library on the SMPTLIB volumes. All applicable modules are copied to the SMPWRK5 data set before the link edit, except for those in one of the SMPTLIB data sets chosen by SMP.

This data set is used during APPLY processing and is needed only for the duration of the SMP job step. The disposition of this data set should be specified as DISP=(NEW,DELETE) to minimize space loss problems. If you require that the data set be kept beyond the duration of the SMP job step, it is your responsibility to reclaim any space that might be required by subsequent invocations of SMP.

The blocksize of the data set must be compatible with the blocksize of the SMPTLIB data sets.

SYSLIB Data Set

Ddname: SYSLIB

Acronym: None

Description: Macro library (for assembler)

Attributes: Partitioned; LRECL=80, BLKSIZE=multiple of 80

Device: Direct access only

Information: The macro libraries are used as input to the assembler. These libraries consist of data sets concatenated in the following sequence (for APPLY and RESTORE):

- SMPMTS
- Target system macro libraries (for example, those libraries specified on the SYSLIB operand of the ++MAC modification control statement.)
- Distribution macro libraries (for example, those libraries specified on the DISTLIB operand of the ++MAC modification control statement.)

For ACCEPT, only the distribution macro libraries make up the input to the assembler.

The blocksize of the first data set in the concatenation must be equal to or larger than any of the subsequent data sets in the concatenation.

SYSPRINT Data Set

Ddname: SYSPRINT

Acronym: None

Description: Output Data Set

Attributes: Sequential

Device: SYSOUT, printer, direct access, or tape. SYSOUT or a printer is recommended because SYSPRINT might be opened with different DCB attributes by the utilities and service aids invoked by SMP.

Information: This data set contains all output generated by all invoked programs. The LRECL, BLKSIZE, or RECFM attributes should not be specified unless they are compatible with the attributes used by the utilities invoked.

You can specify an output listing data set to replace the SYSPRINT data set, which is the default. See 'The UCL SYS Operands' in Chapter 7 for information regarding substitute ddnames for SYSPRINT.

SYSUT1, SYSUT2 and SYSUT3 Data Sets

Ddname: SYSUT1, SYSUT2 and SYSUT3

Acronym: None

Description: Temporary Utility Storage Data Sets

Attributes: Sequential

Device: Direct access only

Information: These data sets are used as scratch data sets for SMP and any programs called by SMP that require work data sets.

SYSUT4 Data Set

Ddname: SYSUT4

Acronym: None

Description: Temporary Utility Storage Data Set

Attributes: Sequential; TRK=1, LRECL=80, BLKSIZE=multiple of 80

Device: Direct access only

Information: Required only if RETRY is requested or defaulted on an APPLY, ACCEPT, or RESTORE.

Target and Distribution Library Data Sets

Ddname: The ddnames used to define these libraries should be the lowest level qualifiers of the data set names. For example, SYS1.LINKLIB has the ddname LINKLIB.

Acronym: tgtlibs, DLIBs

Description: Target and distribution libraries

Attributes: Partitioned

Device: Direct access only

Information: These libraries contain updated versions of macros, source modules. and load modules stored during APPLY, ACCEPT, and RESTORE processing.

SMPADDIN Dataset

Ddname: SMPADDIN

Acronym: None

Description: Contains UNLOAD control statements

Attributes: Sequential

Device: Card, Tape, or Direct access

Information: The SMPADDIN dataset is used to contain control statements that are used during the UNLOAD function. If the ADDIN option is specified on the UNLOAD function control statement, then the SMPADDIN DD statement must be present.

SMPPUNCH Dataset

Ddname: SMPPUNCH

Acronym: None

Description: Output from the UNLOAD function

Attributes: Sequential, LRECL=80, BLKSIZE=Multiple of 80

Device: Card, Tape, or Direct access

Information: The SMPPUNCH dataset contains dumped CDS, or ACDS data in UCLIN format control statements.

Chapter 10: SMP Messages

This chapter contains the SMP diagnostic messages, arranged in alphanumeric order. Each message is described in the following format:

Message and Message Text: The number of the message followed by the text of the message in the format:

HMAnnns yy text

where:

- nnn - the message serial number
- s - the severity code, as follows:

  0 - Informational message (return code = 0)

  1 - Warning message (return code = 4)

  2 - Error message (return code = 8)

  3 - Severe error message (return code = 12)

  4 - Terminal error message (return code = 16)

  The severity code of a message is set when that message causes an SMP return code to be set. The severity code is not propagated to further messages. If a message does not cause an SMP return code to be set, the severity code of that message is 0.

  For example, if two SYSMODs (UZ00001 and UZ00002) are selected for APPLY processing and SYSMOD UZ00001 is not found on the CDS, the following messages are issued for SYSMOD UZ00001:

  HMA2462 ** SYSMOD UZ00001 NOT FOUND ON SMPCDS LIBRARY

  HMA2260 APPLY PROCESSING TERMINATED FOR SYSMOD UZ00001

  The following message is issued for SYSMOD UZ00002:

HMA2050 APPLY PROCESSING COMPLETED - HIGHEST RETURN CODE IS 08

Refer to 'Detecting Error Conditions' in Chapter 5 for a description of the SMP return codes.

- yy - the severity highlighting code, as follows:

  - blanks - severity 0 and 1 messages
  - ** - severity 2, 3, and 4 messages

- text - the message text. Optional text is indicated by brackets.

Explanation: Describes what caused this message to appear and explains the values of the operands set by SMP in the message text.

System Action: Describes what SMP does when the error condition is detected.

Programmer Response: Explains what you should do when you receive this message.

HMA201 xxxx FAILED FOR LIBRARY SPECIFIED BY lib

Explanation:

- xxxx - OPEN or CLOSE
- lib - the ddname of the library that could not be opened or closed.

The function in progress terminates.

System Action: The messages that follow indicate the action taken by SMP.

Programmer Response: If OPEN failed, check for a missing DD statement or an incorrect data set name, or perform any steps required to correct the problem, and resubmit the job. If CLOSE failed, resubmit the job. If close continues to fail, data set maintenance is required.

HMA202 UNABLE TO OBTAIN STORAGE FOR WORK AREAS

Explanation: Insufficient storage was available for SMP to allocate internal tables.

System Action: Subsequent messages in the output listing indicate the actions taken by SMP.

Programmer Response: Increase the REGION parameter on the EXEC statement (VS2) or increase the partition size (VS/1), or decrease the number of SYSMODs being processed in this run, and resubmit the job.

HMA203 SYNTAX ERROR IN {xxx | yyy INPUT | EXEC PARM} STATEMENT AT COL nn

- xxx - CONTROL or UCL
- yyy - LINKAGE EDITOR, ASSEMBLER, or IEBCOPY
- nn - column number

When xxx=CONTROL:

Explanation: A syntax error was found in the modification control statement or the control statement at the specified column. Note that the message indicates that the line immediately previous is the one with the syntax error.

System Action: If the error occurred in the modification control statement, processing terminates for this SYSMOD. If the error occurred in an SMP control statement, processing terminates for the control statement.

Programmer Response: Check the format of the keyword on the specified modification control statement or control statement. Correct the syntax error and resubmit the job.

When xxx=UCL:

Explanation: A syntax error was detected in the UCL statement at the specified column.

System Action: The UCL statement is ignored. Processing continues with the next UCL statement.

Programmer Response: Correct the UCL statement and resubmit the job.

When yyy=LINKAGE EDITOR, ASSEMBLER, or IEBCOPY:

Explanation: During JCLIN processing, a syntax error was found on an input statement for the job step being scanned.

System Action: The scan terminates.

Programmer Response: Correct the error and resubmit the job.

When EXEC PARM is produced:

Explanation: An invalid parameter was specified on the EXEC statement.

System Action: SMP processing terminates.

Programmer Response: Correct the problem and resubmit the job.

HMA204 lib AT HIGHER|LOWER FUNCTION LEVEL THAN CURRENT HMASMP

<u>Explanation</u>:

- lib - ddname of data set

If HIGHER is specified, the data set named is at a higher release level than the level of SMP being used. If LOWER is specified, the data set identified is not in a format acceptable to SMP Release 4, but applies to a previous release of SMP.

<u>System Action</u>: SMP terminates.

<u>Programmer Response</u>: Ensure that the correct data set and version of SMP are being used and rerun the job.

HMA205 {HMASMP|function|UNKNOWN} PROCESSING COMPLETED-HIGHEST RETURN CODE IS rc

<u>Explanation</u>:

- function - the function being processed
- rc - the return code for that function.

If HMASMP is specified, rc is the return code for the job step. If UNKNOWN is specified, SMP was not able to determine the function that was being processed.

<u>System Action</u>: The system action is determined by the return code.

<u>Programmer Response</u>: See the return codes for each function in Chapter 7 to determine the success or failure of the function that was executed.

HMA206 USER EXIT RETURN CODE INDICATES TERMINATION OF {SYSMOD|function|SMP}

Explanation:

- function - the current function

The return code from a user exit routine indicated termination of the SYSMOD in process, the current function, or all of SMP.

System Action: Processing is terminated as indicated in the message.

Programmer Response: Determine why the user exit routine terminated the request. Ensure that the exit routine issued the correct return code for this request.

HMA207 UNKNOWN USER EXIT RETURN CODE-{function|SMP} TERMINATED

Explanation:

- function - the current function

The user exit routine issued an undefined return code.

System Action: Based on the exit routine called, either the current function terminates or SMP terminates.

Programmer Response: Check the logic of the user exit routine to ensure that only defined codes are returned.

HMA214 STORE FAILED FOR type name ON lib LIBRARY

Explanation:

- type - the entry type

- name - the entry name
- lib - ddname of data set

The directory entry for this entry type and name cannot be stored. A previous message in the output listing indicates the reason.

System Action: Processing for this SYSMOD terminates.

Programmer Response: Determine the cause of the error from the previous messages. Correct the error and resubmit the job.

HMA216 UPDATE {FAILED|SUCCESSFUL} -MEMBER=name -LIBRARY=lib -SYSMOD=nnn -RETURN CODE=rc

Explanation:

- name - the entry name
- lib - ddname of data set
- nnn - SYSMOD-ID
- rc - return code from IEBUPDTE

An execution of IEBUPDTE completed for the named entry into library lib with a return code equal to rc. The element represented by the entry name was part of SYSMOD nnn.

System Action: Processing continues as indicated by the messages that follow in the output listing.

Programmer Response: If IEBUPDTE failed, examine the output to determine the cause of the error. If IEBUPDTE error message "MEMBER NAME NOT FOUND" was also issued, ensure that the member exists on DISTLIB and/or SYSLIB as reflected in the CDS or the modification control statement.

HMA218 SUCCESSFULLY STORED type name ON lib LIBRARY

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of the SMP data set

The named entry was successfully stored or restored.

System Action: SYSMOD processing continues.

Programmer Response: None.

HMA219 PEMAX EXCEEDED FOR type name ON lib LIBRARY

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of SMP data set

The entry cannot be created, updated, or listed because the SYSMOD, module, or macro requires a PEMAX value greater than the value specified in the SYSTEM entry.

System Action: Processing is terminated for the SYSMOD or function.

Programmer Response: Increase the value of PEMAX in the SYSTEM entry using the UCLIN control statement. The value of PEMAX should not be decreased after SYSMODs have been processed with a larger PEMAX or existing SYSMOD entries may be too large for SMP to process.

HMA224 SUCCESSFULLY DELETED type name ON lib LIBRARY

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of the data set

The named entry was successfully deleted from the named library.

System Action: Processing continues.

Programmer Response: None.

HMA226 xxx PROCESSING TERMINATED FOR SYSMOD nnn

Explanation:

- xxx - RECEIVE, REJECT, APPLY, RESTORE or ACCEPT
- nnn - SYSMOD-ID

The reason for the failure is described in a preceding message. The error was found for one SYSMOD only. Other SYSMODs will continue to be processed. Additional information may be found in the LOG data set.

System Action: Processing is terminated for the SYSMOD or function.

Programmer Response: Check previous messages to determine the cause of error. Correct the error and resubmit the job.

HMA227 xxx PROCESSING SUCCESSFULLY COMPLETED FOR SYSMOD nnn

Explanation:

- xxx - ACCEPT, APPLY, REJECT, or RESTORE
- nnn - SYSMOD-ID

Processing successfully completed for the specified function.

System Action: None.

Programmer Response: None.

HMA228 IEANUC01 NOT FOUND ON NUCLEUS LIBRARY

Explanation: The nucleus, IEANUC01, was not found on the nucleus library as a result of a BLDL operation.

System Action: Processing for all SYSMODs affecting IEANUC01 is terminated.

Programmer Response: Create IEANUC01 or specify a different NUCLEUS DD statement.

HMA229 CONTROL STATEMENT IGNORED DUE TO PREVIOUS ERROR

Explanation: An error, described in a previous message, caused this control statement to be ignored. The control statement is checked for syntax errors but is not processed.

System Action: Processing continues with the next statement.

Programmer Response: Correct the cause of the error on the previous control statement and rerun the job.

HMA230   IEHIOSUP EXECUTED FOR {APPLY|RESTORE} - RETURN CODE=rc

Explanation:

- rc - return code

The IEHIOSUP program was executed to update the TTR entries in the transfer control tables of the SVCLIB.

System Action: If the return code is non zero, the function and job step are terminated.

Programmer Response: See Chapter 4 to determine the success or failure of IEHIOSUP.

HMA231   IMASPZAP CONTROL STATEMENT ERROR IN MODULE mod FOR SYSMOD nnn

Explanation:

- mod - module name
- nnn - SYSMOD-ID

SMP detected a syntax error in the IMASPZAP statement for the specified module name in the named SYSMOD.

System Action: Processing of the named SYSMOD terminates. Processing continues with the next SYSMOD.

Programmer Response: Correct the syntax error and resubmit the HMASMP step.

HMA234   BLDL FAILED FOR PROGRAM pgm REQUIRED FOR HMASMP EXECUTION

Explanation:

- pgm - program name

The specified program is required in order for HMASMP to execute, but cannot be found.

System Action: The step terminates. For the exceptional system action when using IEHIOSUP, see Chapter 4. If the LINKLIB DD statement was present and IEHIOSUP was not found, SMP terminates.

Programmer Response: Add the indicated program to the JOBLIB, STEPLIB, or link library. This problem can occur when an invalid utility name is specified in the PTS SYSTEM entry. In this case, correct the name and resubmit the job. If IEHIOSUP is used, ensure that a LINKLIB DD statement is present, and that the library specified contains a version of IEHIOSUP.

HMA237 ZAP {VERIFY|REPLACE} PASS {FAILED|SUCCESSFUL} -MOD=xxx -LMOD=yyy -LIBRARY=zzz -SYSMOD=nnn -RETURN CODE=rc

Explanation:

- xxx - module name
- yyy - load module name
- zzz - library name
- nnn - SYSMOD-ID
- rc - return code

IMASPZAP completed for module xxx in load module yyy in library zzz with a return code equal to rc. Module xxx was part of SYSMOD nnn.

System Action: Processing for the SYSMOD is terminated if the return code is non zero and greater than the user-specified or default return code.

Programmer Response: Check the output from IMASPZAP to determine the cause of the error. Correct the error and resubmit the job.

HMA238 COPY {FAILED | SUCCESSFUL} -MOD=xxx -LMOD=yyy -LIBRARY=zzz -SYSMOD=nnn -RETURN CODE=rc

Explanation:

- xxx - module name
- yyy - load module name
- zzz - library name
- nnn - SYSMOD-ID
- rc - return code

IEBCOPY completed for module xxx into load module yyy in library zzz with a return code equal to rc. Module xxx was part of SYSMOD nnn.

Multiple SYSMODs might have LMODs copied in one invocation; therefore, some SYSMODs might have modules successfully copied even though an error code resulted. This message indicates that all the modules and/or load modules handled during this invocation of IEBCOPY failed although only one may have an error. Also, this message may appear for modules within a SYSMOD that were never copied if other modules in the SYSMOD were in error.

System Action: Processing of the SYSMOD is terminated if the return code is non zero and greater than the user-specified or default return code.

Programmer Response: If the copy failed, check the output from IEBCOPY to determine the error. Correct the error and resubmit the job.

HMA239 LINK {FAILED | SUCCESSFUL} -MOD=xxx -LMOD=yyy -LIBRARY=zzz -SYSMOD=nnn -RETURN CODE=rc

Explanation:

* xxx - module name
* yyy - load module name
* zzz - library name
* nnn - SYSMOD-ID
* rc - return code

An execution of the linkage editor completed for module xxx into load module yyy in library zzz with a return code equal to rc. Module xxx was part of SYSMOD nnn.

Multiple SYSMODs might cause modules to be link edited in one invocation; therefore, some SYSMODs might have modules that are link edited successfully even though an error code resulted. This message indicates that all of the modules and/or load modules handled during this invocation of the linkage editor failed although only one may have an error. Also, this message may appear for modules within a SYSMOD that were never link edited if other modules in the SYSMOD resulted in an error.

System Action: Processing of the SYSMOD is terminated if the return code is non-zero and greater than the user-specified or default return code.

Programmer Response: If the link edit failed, check the output from the linkage editor to determine the error. Correct the error and resubmit the job.

HMA240 ASSEMBLY {FAILED | SUCCESSFUL}
-MOD=xxx -LIBRARY=zzz
-SYSMOD=nnn -RETURN CODE=rc

Explanation:

* xxx - module name
* zzz - library name
* nnn - SYSMOD-ID
* rc - return code

An assembly completed for module xxx from library zzz with a return code equal to rc. Module xxx was part of SYSMOD nnn.

System Action: Processing of the SYSMOD terminates if the return code is greater than the user-specified or default return code.

Programmer Response: If the assembly failed, check the output from the assembler to determine the cause of the error.

HMA245 SYSMOD nnn SELECTED FOR RESTORE ERROR CONDITION

| Explanation

- nnn - SYSMOD ID
- error condition - (see below)

The SYSMOD named cannot be RESTORED due to one of the following error conditions:

- IS SUPERSEDED - SYSMOD nnn was found by SMP as a superseded only entry on the CDS. This means that the SYSMOD was never applied by SMP; rather, it was supreseded by one or more SYSMODs which were applied. In this situation, SMP cannot determine the set of SYSMODs which should be restored.
- DELETES SYSMODS - SYSMOD nnn deleted other SYSMODs when it was applied. SMP cannot restore the elements from the deleted SYSMODS, therefore, SYSMOD nnn cannot be restored.
- IS DELETED - SYSMOD nnn was deleted by another SYSMOD which was applied.
- HAS BEEN ACCEPTED - SYSMOD nnn is accepted into the system's distribution libraries. Therefore, the elements on the distribution libraries cannot be used to restore the target system libraries.
- IS NOT APPLIED - SYSMOD nnn is not applied and therefore cannot be restored.

| System Action SYSMOD nnn is terminated. If SYSMOD nnn is a function-type SYSMOD, this message will be followed by HMA370 indicating that the SMP RESTORE function is terminating.

| Programmer Response Correct the list of SYSMODs selected for

RESTORE by eliminating the named SYSMOD from the SELECT or GROUP list.

HMA246 type name NOT FOUND ON lib LIBRARY
[FOR SYSMOD nnn]

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of the data set
- nnn - SYSMOD-ID

An entry for the named element does not exist on the specified library and is required for this function.

System Action: The system action can be determined from

examination of subsequent messages in the output listing. However, if the entry type specified is assembly, subsequent messages do not result and assemblies are not done for the SYSMOD; processing continues for the SYSMOD.

Programmer Response: Use the SMPLOG to determine why the named entry was not found on the library. It is possible that the SYSMOD being processed is not applicable to your system.

HMA247 BLDL FAILED IN LIBRARY lib FOR LOAD MODULE mod IN SYSMOD nnn

Explanation:

- lib - ddname of data set
- mod - the load module name
- nnn - SYSMOD-ID

A BLDL was issued to obtain linkage editor parameters but failed for this load module.

System Action: For ACCEPT processing, a default set of linkage editor parameters is used: 'RENT, REUS, DC, and REFR'. For APPLY processing, the parameters used are those specified during system generation.

Programmer Response: If you are applying the SYSMOD, check for an incorrect library name.

HMA248 THE xxx FUNCTION WAS REQUESTED - NO SYSMODS MEET SPECIFICATIONS

Explanation:

where xxx is one of the following SMP functions (RECEIVE, APPLY, ACCEPT, RESTORE or REJECT)

The requested function terminated because there were no SYSMODs that met the specifications that you indicated on the control statement.

System Action: Processing of the named function terminates. Processing continues with the next control statement.

Programmer Response: Review the other messages that were issued during this function, and verify that the operands that you specified on the control statement are correct.

HMA249 SYSMOD nnn FAILED BECAUSE OF NAME CARD CONFLICT IN MODULE mod

Explanation:

- nnn - SYSMOD-ID
- mod - module name

Name cards of different types occurred within the same HMASPZAP function (NAME csect and NAME lmod csect).

System Action: Processing of the named SYSMOD terminates. Processing continues with the next SYSMOD.

Programmer Response: Correct the NAME cards and resubmit the job.

HMA252 INCOMPLETE HMASMP CONTROL STATEMENT

Explanation: SMP detected an incomplete control statement. An end-of-file occurred before the end of the statement. The SMP control statement in error is listed before this message.

System Action: The function is not performed. The action of SMP is indicated by the messages that follow in the output listing.

Programmer Response: Check for a missing comment terminator (*/), a missing statement terminator (.), or a previous LOG control statement that has missing parentheses.

HMA253 type ENTRY name TO BE DELETED DOES NOT EXIST

Explanation:

- type - the entry type
- name - the entry name

When updating the library specified on the UCLIN control statement, SMP could not find the entry to be deleted.

System Action: The UCL statement is ignored. Processing continues with the next UCL statement.

Programmer Response: Correct the UCL statement and resubmit the job.

HMA255 UPDATE COMPLETE FOR name

Explanation:

- name - the entry name

UCLIN processing for the entry completed.

System Action: Update processing continues with the next UCL statement.

Programmer Response: None.

HMA256 UPDATE PROCESSING TERMINATED - UPDATE NOT COMPLETE

Explanation: UCLIN processing for the entry did not complete because of an error identified in a previous message. The entry was not changed.

System Action: Update processing terminates.

Programmer Response: Correct the source of the error and resubmit the job.

HMA257 SPECIFIED UPDATE RESULTS IN INSUFFICIENT DATA - xxx REQUIRED

Explanation:

- xxx - the required UCL keyword

Insufficient data was supplied to update an entry. The entry was not changed.

System Action: The UCL statement is ignored. Processing continues with the next UCL statement.

Programmer Response: Provide the missing information and resubmit the job.

HMA258 END OF FILE ON UCL INPUT STREAM - PROCESSING TERMINATED

Explanation: End of file occurred on the SMPCNTL data set before the ENDUCL statement was found.

System Action: The current UCL statement, if any, is ignored and UCL processing is terminated.

Programmer Response: Add the ENDUCL statement to the input data stream and resubmit the job.

HMA259 type name ELEMENT PEMAX EXCEEDED ATTEMPTING TO ADD element ____.

| Explanation

- type - Entry Type
- name - Entry name
- element - Element name

The attempt to add a sub-entry to the specified entry causes the number of elements in the entry to exceed the maximum allowed number (PEMAX or fixed value.)

| System Action For UCL processing, the UCL statement is ignored, and processing continues with the next UCL

statement. For RECEIVE, APPLY, ACCEPT and RESTORE processing, the SYSMOD associated with the entry-type, entry-name is terminated.

During RECEIVE processing this situation can also occur for the SMPPTS SYSTEM ENTRY. In this case, the named FMID entry is not added to the SMPPTS SYSTEM ENTRY; however, the function-type SYSMOD is successfully RECEIVED.

Programmer Response For UCL processing, the number of subentries specified in the UCL statement may be reduced or the PEMAX value in the SYSTEM entry may be increased. For RECEIVE, APPLY, ACCEPT and RESTORE processing, the PEMAX value in the SYSTEM entry must be increased.

If this situation occurs for the SMPPTS SYSTEM ENTRY, the

PEMAX value in the SYSTEM entry must be increased and UCLIN must be run against the SMPPTS SYSTEM entry to add the FMIDs which were RECEIVED but not added to the SYSTEM entry.

HMA261 xxx ENTERED IS NOT EQUAL TO xxx yyy IN ENTRY

Explanation:

- xxx - the UCL keyword
- yyy - the value of the xxx keyword in the existing entry

Using the UCL DEL statement, you requested SMP to delete the xxx keyword; however, the value specified to be deleted did not match the value of the existing entry.

System Action: UCL processing terminates for the UCL statement.

Programmer Response: Resubmit the UCL statement, specifying the correct value.

HMA262 ERROR FORCES JCLIN SCAN TO TERMINATE

Explanation: An error, explained in a previous message, causes the JCLIN scan of the Stage I system generation file to terminate.

System Action: JCLIN processing terminates.

Programmer Response: Correct the cause of the previous error and resubmit the job.

HMA263 ERROR OCCURRED IN STEP xxx OF JOB jjj

Explanation:

- xxx - the step name

- jjj - the job name

JCLIN processing. This message indicates the job and step in which an error, indicated by a previous message, occurred. An error description follows this message. Message HMA263 is preceded by the control statement in error, and the description of the error.

System Action: None.

Programmer Response: Error descriptions appear as follows:

- LAST LINE PROCESSED

  An I/O error occurred and this was the last line processed by SMP. Resubmit the job after correcting the error, if necessary.

- TABLE OVERFLOW

  During linkage editor processing, this was the last line processed before the work area was used up. Allocate more main storage and resubmit the job.

- LAST LKED CNTL STMT

  A syntax error was found in a linkage editor statement during linkage editor processing. Consult the OS/VS Linkage Editor and Loader for the correct format, correct the error, and resubmit the job.

- NO MODNAME FOUND IN STMT

  The module name is not specified on the SYSLMOD DD statement or on a NAME link edit statement. Correct the error and resubmit the job.

- NO MOD KEYWORD FOUND

  Neither the NAME link edit statement nor the MOD= keyword was found on the EXEC statement during linkage editor processing. Correct the error and resubmit the job.

- INVALID KEYWORD FOR MOD

  Linkage editor processing found invalid characters in the MOD= keyword. Consult the OS/VS Linkage Editor and Loader for the correct format, correct the error, and resubmit the job.

- ERROR ON MOD NAME STOW

  An undefined error occurred while updating the module during linkage editor processing. Resubmit the job or make the applicable corrections.

- ERR LOCATING MOD KEYWORD

  The MOD keyword was not found on the EXEC statement during assembler processing. Correct the error and resubmit the job.

- MACRO TABLE EXCEEDED

  The space allocated for macro tables was exceeded during assembler processing. Allocate more storage and resubmit the job.

- INVAL. MACNAME SPECIFIED

  A macro name with an incorrect length was found during assembler processing. The length must be from one to eight characters. Correct the error and resubmit the job.

- INVALID IEBCOPY STATEMENT

  The statement printed is syntactically invalid. Consult the OS/VS Utilities manual for the correct IEBCOPY format, correct the error, and resubmit the job.

- NO DSNAME CODED

  DSNAME was not coded on the EXEC statement for the linkage editor procedure. Correct the error and resubmit the job.

- NO SYSLMOD CARD FOUND

  A SYSLMOD DD statement was not found and PGM= was specified on the EXEC statement. Correct the error and resubmit the job.

HMA266 ERROR OCCURRED IN {LINKAGE EDITOR|IEBCOPY| ASSEMBLER} INPUT

Explanation: This message indicates the type of system generation step that was being scanned by JCLIN processing when an error, indicated by a prior message, occurred.

System Action: JCLIN processing terminates.

Programmer Response: Correct the error and resubmit the job.

HMA267 DIRECTORY SPACE EXCEEDED ATTEMPTING TO STORE type name ON lib LIBRARY

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of the data set

The number of directory blocks allocated to the library was exceeded in attempting to store the specified member.

System Action: The member is not stored. SMP action is indicated by messages that follow.

Programmer Response: Increase the allocation for directory blocks for the indicated library and resubmit the job.

HMA268 I/O ERROR OCCURRED ATTEMPTING TO STORE type name ON lib LIBRARY

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of data set

An I/O error occurred while storing the indicated entry in the specified library.

System Action: The entry is not stored. SMP processing terminates as indicated by the following messages.

Programmer Response: Correct the cause of the I/O error and resubmit the job.

HMA269 I/O ERROR OCCURRED ATTEMPTING TO BLDL FOR type name ON lib LIBRARY

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of data set

A BLDL operation produced an I/O error on the library specified.

System Action: SMP action is indicated by the messages that follow in the output listing.

Programmer Response: Correct the cause of the I/O error and resubmit the job.

HMA273 INPUT TEXT NOT FOUND

Explanation: Either inline text or object records, expected following an SMP element modification control statement, were not found, or no input was present in the JCLIN input file.

System Action: The SYSMOD being processed is not received.

Programmer Response: An object deck or text deck must follow element modification control statements for elements not supplied on a TXLIB, LKLIB or RELFILE data set, or JCL must be present in the JCLIN input data set.

HMA274 I/O ERROR-jobname, stepname, unit address, device type, ddname, operation attempted, error description, followed by:
(1) access method, or
(2) rbn and access method, or
(3) track address, block number, access method.

Explanation: An I/O error occurred while processing the data set referenced by ddname. The information provided in the message corresponds to the SYNADAF information described in the OS/VS1 or OS/VS2 Data Management Services Guide.

System Action: SMP action is indicated by the messages that follow in the output listing.

Programmer/Operator Response: Correct the error and resubmit the job.

HMA276 ILLEGAL UPDATE REQUEST FOR lib

Explanation:

- lib - ddname of data set

An illegal combination of UCLIN operations was requested.

System Action: The member in the data set is not updated.

Programmer Response: See "The UCLIN Control Statement" in Chapter 7, for the syntax of the UCLIN statements, correct the statement in error, and resubmit the job.

HMA277 type name TO BE REPLACED DOES NOT EXIST - ADD ASSUMED

Explanation:

- type - the entry type
- name - the entry name

During UCL processing, a REP operation was requested but the entry was not found in the data set specified in the UCLIN control statement.

System Action: UCLIN processing continues and assumes that an ADD operation was requested.

Programmer Response: If 'ADD' was the correct assumption, no further processing is required. If the data set specified is incorrect, issue a DEL request for the data set just updated and a REP request for the correct data set.

HMA281 DUPLICATE ELEMENT name IN SYSMOD nnn

Explanation:

- name - the element name as specified in the SYSMOD
- nnn - SYSMOD-ID

During RECEIVE processing, two modification control statements specifying the same element were found in one SYSMOD. The modification control statement specifying the duplicate element is printed before this message.

System Action: Processing of this SYSMOD terminates.

Programmer Response: Correct the modification control statements so that duplicate elements do not exist and resubmit the job.

HMA282 lib DIRECTORY BLOCKS REQUIRED (xxx)
WILL EXCEED AVAILABLE DIRECTORY BLOCKS (yyy)

Explanation:

- lib - ddname of data set
- xxx - number of directory blocks required for the function to complete
- yyy - number of directory blocks currently allocated

When using the DIS(WRITE) option, SMP determined that the number of directory blocks currently allocated (yyy) is not sufficient to complete the current function. The number of directory blocks required is given.

System Action: The current function being processed terminates. The data set does not reflect any changes made by this function.

Programmer Response: Increase the number of directory blocks for the specified data set to a minimum value of xxx, and resubmit the job.

HMA283 dd DDCARD MISSING [FOR LOAD MODULE mod | FOR COMPRESS | FOR MODULE mod IN SYSMOD nnn]

Explanation:

- dd - ddname
- mod - module name
- nnn - SYSMOD-ID

The specified DD statement does not exist, but it is required for execution of the requested HMAfunction.

System Action: SMP action is indicated by the messages that follow in the output listing.

Programmer Response: Add the required DD statement or correct the ddname.

HMA284 SYSMOD nnn HAD A LOAD MODULE SPECIFICATION ERROR IN MODULE mod

Explanation:

- nnn - SYSMOD-ID
- mod - the module name

The load module name specified on an IMASPZAP NAME statement is not listed in the CDS as a valid load module for the specified module. If a valid load module name has been used in an ALIAS operand, you must use the name that appears in the CDS and not in the ALIAS operand.

System Action: Processing of the named SYSMOD terminates. Processing continues with the next SYSMOD.

Programmer Response: Correct or remove the load module name on the IMASPZAP NAME statement and resubmit the job.

HMA285 lib REFERENCES AN UNMOVABLE DATA SET

Explanation:

- lib - ddname of data set

The compress function has been requested for a library that cannot be compressed because it contains location-dependent data.

System Action: Compress processing for the named data set is bypassed. Processing continues with the next data set.

Programmer Response: None.

HMA287 CONTROL STATEMENT NOT PROCESSED-A USER SPECIFIED RETURN CODE HAS BEEN EXCEEDED

Explanation: You specified a return code for another SMP control statement using the RC operand that determines the processing of the current control statement. SMP determined that the return code for the specified control statement is greater than the limit you specified and did not process the current control statement.

System Action: Processing continues.

Programmer Response: Analyze the SMPOUT data to determine which control statement caused the current control statement to be bypassed. Correct any errors and reevaluate your course of action.

HMA288 NO HMASMP UPDATE FUNCTIONS HAVE BEEN PERFORMED

Explanation: An error, indicated by a previous message, caused JCLIN processing to terminate without updating the CDS.

System Action: Processing continues.

Programmer Response: Correct the cause of the error and re-execute JCLIN processing.

HMA292 INVALID MEMBER NAME ON IEBUPDTE CONTROL STATEMENT

Explanation: The member name on the IEBUPDTE control statement did not match the name on the ++UPDTE, ++MACUPD, or ++SRCUPD modification control statement.

System Action: RECEIVE processing terminates for the SYSMOD.

Programmer Response: Correct the IEBUPDTE control statement and issue the RECEIVE control statement again for the SYSMOD.

HMA302 xxx PROCESSING TERMINATED FOR SYSMOD nnn
- REASON=zzz

Explanation:

- xxx - APPLY, ACCEPT, or RESTORE
- nnn - SYSMOD-ID
- zzz - reason for termination

During xxx processing, SYSMOD nnn was terminated for one of the following reasons:

- SYSTEM ABEND
  - A system abend, as indicated by message HMA432, was intercepted. All SYSMODs for which processing was attempted were terminated by SMP.
- SYSTEM UTILITY FAILURE
  - SMP invoked one of the system utilities (assembler, update, copy, zap, or linkage editor) for one of the elements of the indicated SYSMOD. The utility returned a code that was defined by SMP or the user as an error code; thus, processing for the SYSMOD related to that element is terminated.
- RELATED SYSMOD FAILURE

- The indicated SYSMOD contained a version of a module, macro, or source module that was not selected for processing because another SYSMOD, also being processed, contained a higher level of the element. The SYSMOD with the higher level of the element was terminated. Use the SMP reports and messages to determine which element caused the problem.

- DELETE PROCESSING FAILURE
  - During the processing of a deleted SYSMOD, an error, detected by a previous message, was found. The SYSMOD containing the DELETE operand is terminated.
- INLINE JCLIN FAILURE
  - Inline JCLIN processing for a SYSMOD failed. See previous messages in the output listing for the reasons for failure.
- SMP UPDATE FAILURE
  - Processing completed for all elements of the indicated SYSMOD, but SMP was updating an SMP data set with the SUP and MODID data when an error occurred. This message is preceded by a message indicating the cause of the error.
- REQ SYSMOD FAILURE
  - Processing terminated for the indicated SYSMOD because one of its requisite SYSMODs terminated.
- IFREQ SYSMOD FAILURE
  - Processing terminated for the indicated SYSMOD because one of its IFREQ SYSMODs was terminated.
- PRE SYSMOD FAILURE
  - Processing terminated for the indicated SYSMOD because one of its prerequisite SYSMODs was terminated.
- FMID SYSMOD FAILURE

- Processing terminated for the indicated SYSMOD because the SYSMOD that it specified as the value of the FMID operand was being processed concurrently and terminated.

- ALL SUPERSEDING SYSMODS FAILED

  - Processing terminated for the indicated SYSMOD because all of the SYSMODs being processed that superseded the indicated SYSMOD failed.

- ALL DELETING SYSMODS FAILED

  - The indicated SYSMOD was being processed for deletion; however, the deleting SYSMODs failed.

- MISSING/NOGO REQUISITES

  - Processing terminated for the indicated SYSMOD because one or more requisites (PRE, REQ, IFREQ, or FMID) were missing or failed during APPLY or ACCEPT processing. Message HMA359 follows this message and names the missing or NOGO requisite SYSMODs.

System Action: Processing is terminated for the SYSMOD. No other processing is attempted for any other element of the SYSMOD.

Programmer Response: Examine SMPOUT for messages relating to elements of the SYSMOD, or use the APPLY/RESTORE/ACCEPT SUMMARY reports to determine which associated SYSMOD caused the failure. Correct the cause of the error and rerun the job.

HMA303 COMPRESS {FAILED|SUCCESSFUL}
-LIBRARY=lib -RETURN CODE=rc

Explanation:

- lib - ddname of the data set

- rc - return code

IEBCOPY was executed in order to compress the specified library; rc is the return code from IEBCOPY.

System Action: The current function is terminated if the return code is non-zero and greater than the user-specified or default return code.

Programmer Response: Check the output from IEBCOPY to determine the error. Correct the error and resubmit the job.

HMA304 COMPRESS OPTION INVALID - LIBRARY=lib

Explanation:

- lib - ddname of the data set

A compress of the CDS or ACDS was requested, but is not allowed: compression of these data sets by SMP could result in erroneous processing within the SMP job step.

System Action: COMPRESS for the CDS and ACDS is not performed. Processing continues for the other specified data sets.

Programmer Response: Do not specify the CDS or ACDS as values of the COMPRESS operand. These data sets should be compressed outside of SMP with IEBCOPY.

HMA305 INSUFFICIENT STORAGE FOR lib IN STORAGE STOW/BLDL OPERATIONS

Explanation:

- lib - ddname of data set

Storage was not available to perform STOW/BLDL operations for directories in-storage.

System Action: If the severity of the message was 3, processing for the requested function is not done. If the severity was less than 3, processing continues without the specified directory in storage. A severity of 3 results if the DIS(WRITE) option was specified or if DIS(WRITE) was the default for the requested function.

Programmer Response: For severity 3 messages, rerun the job with either a larger partition or region size or without the DIS(WRITE) option. If the severity was not 3, no further action is necessary.

HMA319 SYSMOD nnn DOES NOT PRE OR SUP ccc ELEMENT iii mmm

Explanation:

- nnn - SYSMOD-ID
- ccc - SMPCDS, SMPACDS, or SELECTED
- iii - RMID or UMID
- mmm - SYSMOD-ID that is not named as a prerequisite or is not superseded

This message is issued to describe the ID check reported by HMA382. It is issued for every element in a service SYSMOD (or a function SYSMOD being re-installed) that does not name, in the PRE or SUP operands, the RMID and all UMIDs of the previously processed version of the named element. When this situation occurs, SMP cannot determine the relationship between the element in SYSMOD nnn and the previously processed version of the element.

If ccc is:

- SELECTED: SYSMOD nnn supplied a version of the element that was selected from another SYSMOD being processed during the current APPLY or ACCEPT processing.
- SMPCDS or SMPACDS: SYSMOD nnn supplied a version of the element that was selected from another SYSMOD processed during a previous APPLY or ACCEPT.

If iii is:

- RMID: mmm is the SYSMOD-ID of the last SYSMOD that supplied a total replacement (++MOD, ++MAC, ++SRC) to the named element.
- UMID: mmm is the SYSMOD-ID of a SYSMOD that supplied an update (++ZAP, ++MACUPD, ++UPDTE, or ++SRCUPD) to the named element.

System Action: This message is always issued as information. The system action can be determined by examination of the preceding HMA382 message.

Programmer Response: The relationship between the elements in the SYSMODs involved must be determined by the user. SYSMOD nnn can be rejected and modified to change the PRE and SUP operands specified. The RMID and UMID attributes of the elements on the CDS or ACDS can be modified using the UCLIN function. In addition, other SYSMODs may be required to be applied before this SYSMOD is processed to establish the correct relationship.

HMA324 type name SUBENTRY IN SYSMOD mmm REGRESSED BY SYSMOD nnn

Explanation:

- type - the subentry type
- name - the element name
- mmm - the SYSMOD-ID of the modid from the element entry
- nnn - the SYSMOD-ID of the SYSMOD being processed

SYSMOD nnn did not specify SYSMOD mmm as a prerequisite, did not supersede SYSMOD mmm, or SYSMOD mmm was applied, accepted, or concurrently being processed with a user modification to the element. Therefore, SYSMOD mmm is considered to be regressed by SYSMOD nnn.

System Action: A warning severity is indicated. Processing of SYSMOD nnn continues.

Programmer Response: None.

HMA325 SYSMOD mmm WHICH SUPERSEDES SYSMOD nnn DOES NOT CONTAIN ELEMENT zzz

Explanation:

- mmm - superseding SYSMOD-ID
- nnn - superseded SYSMOD-ID
- zzz - element name

The superseding SYSMOD does not contain all of the elements contained in the superseded SYSMOD.

System Action: The return code is set to 4 and processing continues.

Programmer Response: Review the SYSMODs and perform any necessary corrections to the indicated elements.

HMA327 INPUT TEXT FOUND AND LKLIB, TXLIB, RELFILE, OR DELETE KEYWORD SPECIFIED ON CONTROL STATEMENT

Explanation: Modification text is found following an element modification control statement that indicated that the input is on a LKLIB, TXLIB or relative file, or, if DELETE is specified, the input text should not have been found.

System Action: Processing terminates for this SYSMOD. Processing continues for any remaining SYSMODs.

Programmer Response: The SYSMOD is improperly constructed. Review the SYSMOD for an omitted element modification control statement, or incorrectly coded '++' characters, or a conflict between the placement of modification text and the specification of the LKLIB, TXLIB, RELFILE, or DELETE keywords.

HMA338 ttt MODIFICATION CONTROL STATEMENT NOT VALID IN FUNCTION SYSMOD

Explanation:

- ttt - ++UPDTE, ++MACUPD, ++SRCUPD, or ++ZAP

The specified modification control statement cannot appear in a function SYSMOD.

System Action: The SYSMOD is improperly constructed and is not received.

Programmer Response: Construct the function SYSMOD using only allowable types of modification control statements.

HMA339 RMID KEYWORD ONLY VALID IN FUNCTION SYSMOD

Explanation: The keyword RMID, optional on the ++MAC, ++MOD and ++SRC modification control statements, is allowed only in a function SYSMOD.

System Action: The SYSMOD is improperly constructed and is not received.

Programmer Response: Specify the RMID operand only on function SYSMODs.

HMA340 RELFILE KEYWORD INVALID WITHOUT FILES KEYWORD ON HEADER MODIFICATION CONTROL STATEMENT

Explanation: When you specify the RELFILE operand, you must also specify the FILES operand on the header modification control statement. The RELFILE operand is optional on the ++JCLIN, ++MAC, ++MOD, or ++SRC modification control statements.

System Action: The SYSMOD is not constructed properly and is not received.

Programmer Response: Specify the FILES operand on the header modification control statement and receive this SYSMOD again.

HMA341 KEYWORDS xxx AND yyy ARE MUTUALLY EXCLUSIVE

Explanation:

- xxx - keyword
- yyy - keyword

The keywords indicated by xxx and yyy cannot be used in the same SMP control statement. The control statement in error is listed in a previous message.

System Action: If the error occurred on a modification control statement, the SYSMOD is not received. If the error occurred on a control statement, the statement is not processed.

Programmer Response: Correct the cause of the problem by removing either keyword and run the job again.

HMA342 ONLY ONE {++JCLIN|xxx} ALLOWED IN A {SYSMOD|STATEMENT}

Explanation:

- xxx - keyword

++JCLIN results if two ++JCLIN modification control statements were specified in the SYSMOD being processed. If the keyword identified by xxx is produced in the message, this keyword was entered more than once on the control statement being processed.

System Action: The SYSMOD is improperly constructed and is not received.

Programmer Response: Either remove one of the ++JCLIN modification control statements (if ++JCLIN appears in the message), or remove any duplicate keywords from the control statement. In both cases, run the RECEIVE function again for this SYSMOD after you have corrected the errors.

HMA343 SMPPTFIN WITH RELFILES MUST BE STANDARD LABEL TAPE

Explanation: In order to receive a SYSMOD that is constructed using relative files, the SMPPTFIN data set on which the SYSMOD is contained must be a standard labelled tape.

System Action: RECEIVE processing for the SYSMOD is terminated.

Programmer Response: It is probable that the wrong tape was mounted. Rerun the job using the correct standard labeled tape.

HMA345 INVALID MODIFICATION CONTROL STATEMENT

Explanation: An invalid SMP modification control statement was encountered by the RECEIVE process.

System Action: The SYSMOD currently being processed by RECEIVE is terminated. Subsequent statements in the SMPPTFIN input stream are syntax checked until the next header modification control statement (++PTF, ++FUNCTION, ++USERMOD, or ++APAR) is encountered. When the next header modification control statement is encountered, RECEIVE processing continues normally.

Programmer Response: Ensure that the modification control statement flagged is syntactically correct and is properly positioned within a set of modification control statements. Correct the erroneous statement or SYSMOD construction, and execute the RECEIVE process again to receive the SYSMOD(s) that were terminated.

HMA346 INVALID IEBUPDTE CONTROL STATEMENT

Explanation: An IEBUPDTE control statement other than "./ CHANGE" or "./ ENDUP" was found following a ++UPDTE, ++MACUPD or ++SRCUPD modification control statement.

System Action: The SYSMOD containing the invalid statement is terminated.

Programmer Response: Correct the SYSMOD and execute the RECEIVE process again.

HMA347 INVALID RECORD. MODIFICATION CONTROL STATEMENT EXPECTED

Explanation: The SMPPTFIN data set input stream contains a non-SMP statement when an SMP modification control statement was expected. This situation can arise when input text follows a modification control statement that has a syntax error.

System Action: All subsequent non-SMP statements are ignored and the SYSMOD containing the invalid statement is terminated.

Programmer Response: Correct the problem and execute RECEIVE again.

HMA348 SYSMOD CONTAINS MORE THAN ONE ++VER MODIFICATION CONTROL STATEMENT FOR THE SAME SREL-FMID PAIR

Explanation: The SYSMOD being received contained more than one ++VER modification control statement naming the SREL and FMID. The ++VER modification control statement that caused the error is the one that immediately precedes this message. A SYSMOD constructed in this manner creates an ambiguous situation at APPLY/ACCEPT time.

System Action: The SYSMOD is terminated.

Programmer Response: Correct the problem and execute RECEIVE again.

HMA349 ++IF MODIFICATION CONTROL STATEMENT NOT ASSOCIATED WITH ANY PRECEDING ++VER MODIFICATION CONTROL STATEMENT

Explanation: A ++IF modification control statement was found that did not follow a ++VER modification control statement. The ++IF modification control statement that caused the error is the one that immediately precedes this message. ++IF modification control statements must follow a ++VER modification control statement so that SMP can associate them with the ++VER modification control statement chosen at APPLY/ACCEPT time.

System Action: The SYSMOD is terminated.

Programmer Response: Correct the problem and execute RECEIVE again.

HMA350 RELFILE GREATER THAN NUMBER OF FILES IN THE SYSMOD

Explanation: A ++JCLIN, ++MOD, ++MAC, or ++SRC modification control statement contained a RELFILE keyword that specified a relative file greater than the number of files specified in the FILES keyword on the header modification control statement. The element modification control statement that caused the error is the one that immediately precedes this message.

System Action: RECEIVE processing terminates immediately. The RECEIVE SUMMARY REPORT is generated.

Programmer Response: Correct the problem and execute RECEIVE again.

HMA351 nnn TERMINATED WHILE LOADING RELFILES. COPY RETURN CODE rc

Explanation:

- nnn - SYSMOD-ID
- rc - return code from copy processing

While relative files were being loaded, SYSMOD nnn terminated.

System Action: RECEIVE processing is terminated.

Programmer Response: Examine the copy SYSPRINT output.

HMA352 ALLOCATE SUCCESSFUL FOR xxx ON VOLUME yyy [-EXISTING DATA SET FOUND]

Explanation:

- xxx - data set name
- yyy - volume serial number

During RECEIVE processing, SMP successfully allocated data set xxx on volume yyy. The data set was allocated during the loading of a relative file for a SYSMOD. 'EXISTING DATASET FOUND' indicates that a preallocated data set was found on the specified volume and that SMP will attempt to load the relative files in the existing data set.

System Action: The allocated data set is used for loading the relative files.

Programmer Response: None.

HMA353 ALLOCATE FAILED FOR xxx {ON VOLUME yyy} - zzz

Explanation:

- xxx - data set name
- yyy - volume serial number
- zzz - reason for the error

During RECEIVE processing, an attempt was made to allocate data set xxx on volume yyy in order to load one of the relative files for a SYSMOD. However, an error occurred during allocation. The error, indicated by zzz is one of the following:

- ERROR CODE = x'nn'

  - Error code x'nn' resulted from DADSM. See _OS/VS2 DADSM Logic_, SY26-3858, or _OS/VS1 DADSM Logic_, SY26-3837, for an explanation of the error codes.

- DATASET FOUND IS NOT A PDS

  - SMP found an existing data set on the specified volume; however, the data set was not a partitioned data set and could not be used to load the relative files.

- DATASET NOT FOUND

  - This message results during APPLY or ACCEPT processing when SMP attempts to find one of the data sets that were allocated and loaded from a relative file during RECEIVE processing. The data set was not found.

- NO VOLUMES SPECIFIED

  - The SMPTLIB DD statement did not specify a list of volumes to search in order to process or allocate a RELFILE data set.

System Action: Processing is terminated for the SYSMOD associated with that data set.

Programmer Response: Determine the cause of the error by examining the DADSM return code and the volumes specified on the SMPTLIB DD statement. If the error occurred during APPLY or ACCEPT processing, ensure that the same volumes are available that were loaded at RECEIVE time.

HMA354 SCRATCH SUCCESSFUL FOR xxx ON VOLUME yyy

Explanation:

- xxx - data set name

- yyy - volume serial number

SMP scratched data set xxx from volume yyy. This data set is one of the data set allocated by SMP for processing relative files.

System Action: SYSMOD processing continues normally.

Programmer Response: None.

HMA355 ERROR PROCESSING type ENTRY FOR SYSMOD sss ON THE ddd.

Explanation

- type - Entry Type
- sss - SYSMOD Id
- ddd - SMP Dataset

An error was found during the processing for SYSMOD sss.

If the entry type is MCS, an error was detected trying to parse the MCS entry on the SMPPTS dataset. This could be the result of an I/O error or a mismatch between the MCS and SYSMOD entries on the SMPPTS.

If the entry type is other than MCS, examine the preceeding SMP output to determine the cause of the error for the named entry type.

System Action: The SYSMOD is terminated.

Programmer Response: In the case of the MCS error, the SYSMOD should be rejected and received again. In any other case, pursue the action indicated by the preceeding error message(s).

HMA356 xxx yyy TO BE ADDED TO ENTRY ALREADY EXISTS {AS zzz}

Explanation:

- xxx - subentry type
- yyy - subentry name
- zzz - the existing value of the subentry

A UCL statement requested that subentry xxx yyy be added but the subentry was already present.

System Action: The UCL statement is not processed. Any other changes requested in the same UCL statement are not done.

Programmer Response: Determine the cause of error (either wrong entry specified or incorrect subentry specified), correct the UCL statement, and rerun the job.

HMA357 xxx yyy TO BE DELETED FROM ENTRY DOES NOT EXIST

Explanation:

- xxx - subentry type
- yyy - subentry name

A UCL statement requested that subentry xxx yyy be deleted, but the subentry was not present.

System Action: The UCL statement is not processed. No changes requested in that statement will be made.

Programmer Response: Rerun the UCL statement without the specified subentry.

HMA358 xxx yyy TO BE REPLACED IN ENTRY DOES NOT EXIST - ADD ASSUMED

Explanation:

- xxx - subentry type
- yyy - subentry name

A UCL statement requested that subentry xxx replace the current value of the subentry yyy; however, no data currently exists for the specified subentry. SMP assumes that ADD was specified and processing continues.

System Action: The UCL statement is processed as if ADD were specified for the indicated subentry.

Programmer Response: No further processing is required if 'ADD' is the correct assumption. If the subentry should not have been added, use the UCL DEL function to delete the subentry. Rerun the UCL REP request, specifying the correct entry and subentry.

HMA359 nnn ttt (BYPASSED) (CAUSER=ccc)

| <u>Explanation</u>

- nnn - SYSMOD Id
- ttt - Requisite condition (see below)

This message follows message HMA302 or HMA420 (referred to as HMAxxx below) and lists the requisite conditions which were not satisfied for the SYSMOD named in HMA302 or HMA420.

The following requisite conditions may be encountered:

- PRE - nnn is a Prerequisite for the SYSMOD named in HMAxxx.
- REQ - nnn is a Requisite for the SYSMOD named in HMAxxx.
- IFREQ - nnn is a conditional requisite for the SYSMOD named in HMAxxx. The SYSMOD which caused this requisite to be generated will be named in the CAUSER= portion of the message if the causer is other than the SYSMOD named in HMAxxx.
- FMID - nnn is the FMID of the SYSMOD named in HMAxxx. This variation may occur at APPLY or ACCEPT since nnn must be APPLIED or ACCEPTED along with (or before) the SYSMOD named in HMAxxx.
- SUPING - nnn is a SYSMOD which Supersedes the SYSMOD named in HMAxxx.
- PREING - nnn is a SYSMOD which has the SYSMOD named in HMAxxx as a Prerequisite.
- FMIDED - nnn is the FMID of the SYSMOD named in HMAxxx. This variation may occur at RESTORE when nnn must be restored along with the SYSMOD named in HMAxxx.

ccc - The SYSMOD which supplied the ++IF statement which resulted in the missing requisite condition described by this message.

When BYPASSED appears in this message, the message follows HMA420 and shows the requisite conditions which were bypassed in order to APPLY, ACCEPT or RESTORE the SYSMOD named in HMA420.

Note that these un-satisfied requisite conditions can occur for a number of reasons including (1) previous failures of the named SYSMODs, (2) the SYSMODs not being found on the SMPPTS dataset at APPLY or ACCEPT, (3) the SYSMODs not being found on the SMPACDS at RESTORE and (4) incorrect specification of the set of SYSMODs to be APPLIED, ACCEPTED or RESTORED in the SELECT or GROUP list.

<u>System Action</u> See HMA302 or HMA420

<u>Programmer Response</u>: When this message follows HMA302, you must ensure that the named requisite is either successfully installed on the target system or is selected for installation during the APPLY or ACCEPT step for the SYSMOD named in message HMA302. If the requisite SYSMOD listed is being installed during the current APPLY or ACCEPT step, examine the preceding messages on SMPOUT to determine the cause of termination of the requisite SYSMOD. Note that if a SYSMOD being installed supersedes the named requisite, the termination of the superseding SYSMOD might cause the requisite SYSMOD to fail.

When this message follows HMA420, no programmer response is required.

HMA360 lib SYSTEM ENTRY NOT FOUND

<u>Explanation</u>:

- lib - ddname of data set

The specified data set, required to perform the requested SMP processing, did not have a SYSTEM entry. The SYSTEM entry must be initialized prior to performing any processing so that SMP can verify that the system and release specified are at the correct level. In addition, the DIS(WRITE) option cannot be used until a SYSTEM entry has been created on the specified data set.

System Action: Processing for the requested function is not done.

Programmer Response: Use the UCL statement to add a SYSTEM entry with the appropriate options to the specified data set.

HMA361 lib1 SREL DOES NOT MATCH lib2 SREL

Explanation:

- lib1 - ddname of data set
- lib2 - ddname of data set

SMP requires both data sets to perform the requested processing. Both data sets must have the same system and release level.

System Action: Processing for the requested function is not done.

Programmer Response: Check your JCL to ensure that both data sets were specified. Ensure that both data sets were at the same system and release level. Use the UCL statement to correct the SYSTEM entry if an error is found.

HMA362 INVALID UPDATE INPUT TEXT

Explanation: The text following a ++UPDTE, ++MACUPD, or ++SRCUPD modification control statement had one of the following errors:

- The ./ CHANGE statement was not found.
- More than one ./ ENDUP or ./ CHANGE statement was encountered.

System Action: The SYSMOD being received is terminated.

Programmer Response: Correct the problem and execute RECEIVE again.

HMA363 SYSMOD CONSTRUCTION ERROR:

{nnn APPEARS AS kkk AND zzz OPERAND}
{++IF FMID OPERAND NAMES ++VER FMID OPERAND}
{++VER MODIFICATION CONTROL STATEMENTS ARE INCONSISTENT}
{++VER mmm OPERAND INVALID ON SERVICE SYSMOD}

Explanation:

- nnn - value of kkk and zzz operands
- kkk - operand
- zzz - operand
- mmm - ++VER modification control statement operand

The following conditions are possible:

- nnn APPEARS AS kkk AND zzz OPERAND

  The same value, nnn, is specified for operands kkk and zzz.

- ++IF FMID OPERAND NAMES ++VER FMID OPERAND

  The value of the ++IF modification control statement FMID operand is the same as the value of the ++VER modification control statement FMID operand.

- ++VER MODIFICATION CONTROL STATEMENTS ARE INCONSISTENT

  The ++VER modification control statement specified an FMID, but a previously encountered, applicable ++VER modification control statement did not specify an FMID, or the ++VER modification control statement encountered did not specify an FMID operand, but a previously encountered, applicable ++VER modification control statement did.

- ++VER mmm OPERAND INVALID ON SERVICE SYSMOD

  The operand mmm is invalid for a service SYSMOD.

<u>System Action</u>: The SYSMOD being received is terminated.

<u>Programmer Response</u>: Correct the problems and execute RECEIVE again.

HMA364 DELETE ERROR, SYSMOD nnn DELETES yyy BOTH OF WHICH ARE BEING APPLIED|ACCEPTED

<u>Explanation</u>:

- nnn - SYSMOD-ID
- yyy - SYSMOD-ID

Function SYSMOD nnn is eligible to be processed, but its ++VER modification control statement specifies function SYSMOD yyy, which is also eligible to be processed, as the value of the DELETE operand.

<u>System Action</u>: SYSMOD nnn is terminated. Message HMA370 follows to indicate that the function is also terminated.

<u>Programmer Response</u>: The two SYSMODs cannot be processed concurrently. Use the SELECT, GROUP, or EXCLUDE operands on the APPLY or ACCEPT control statement.

HMA365 SELECTED xxx nnn CANNOT BE ACCEPTED UNLESS yyy IS SPECIFIED

<u>Explanation</u>:

- xxx - APAR or USERMOD
- nnn - SYSMOD-ID
- yyy - APARS or USERMODS

The select or group list on the ACCEPT control statement contained either a user modification, and the USERMODS operand was not specified, or an APAR, and the APARS operand was not specified.

System Action: SYSMOD nnn is terminated.

Programmer Response: Specify the appropriate operand, APARS or USERMODS, to process the SYSMOD.

HMA366 lib MAY NOT REFLECT TRUE STATUS OF LIBRARIES

Explanation:

- lib - ddname of data set

During the previous invocation of SMP, the DIS(WRITE) option was specified. SMP processing for the requested function was interrupted prior to attempting to rewrite the in-storage copy of the data set directory specified by lib. As a result, the directory on the direct access device will not reflect any of the processing that SMP did complete prior to its termination.

System Action: SMP processing continues normally with the first function requested in the SMPCNTL data set.

Programmer Response: Determine what function was being executed during the last invocation of SMP and re-execute that function so that the appropriate SMP data set is updated to reflect the true status of the libraries.

HMA367 lib IS NOT USABLE DUE TO PARTIAL DIRECTORY REWRITE

Explanation:

- lib - ddname of data set

During the prior invocation of SMP, the DIS(WRITE) option was specified. Processing for the function completed, but during the process of writing the in-storage copy of the directory, an error occurred that forced SMP to terminate the rewrite process. Since the data set has been partially rewritten, the status of the data within the data set is unclear. To SMP, the data set is no longer usable.

System Action: SMP processing terminates.

Programmer Response: Obtain a backup copy of the specified data set and restore the data set to a prior level.

Re-execute the control statements that modified the system or SMP data sets during the previous execution of SMP. This reupdates the system libraries and updates the SMP data sets.

HMA368 lib IN STORAGE DIRECTORY SUCCESSFULLY REWRITTEN

Explanation:

- lib - ddname of data set

When processing with the DIS(WRITE) option, either requested or as the default mode of operation, SMP successfully rewrote the updated in-storage directory for the data set. The data set now reflects all the updates that were done during processing.

System Action: The data set directory is written.

Programmer Response: None.

HMA369 SYSMOD nnn SELECTED. NOT FOUND ON SMPPTFIN.

Explanation:

- nnn - SYSMOD-ID

SYSMOD nnn was specified in the SELECT list of the RECEIVE control statement but was not encountered in the SMPPTFIN input stream.

System Action: RECEIVE processing continues for other SYSMODs specified in the SELECT list.

Programmer Response: Place the desired SYSMOD in SMPPTFIN and re-execute RECEIVE.

HMA370 xxx PROCESSING TERMINATED BECAUSE FUNCTION SYSMOD nnn FAILED

Explanation:

- xxx - type of processing being performed
- nnn - SYSMOD-ID

When a function SYSMOD fails, SMP processing is terminated.

System Action:

The function terminates.

Programmer Response: Either correct the error causing the SYSMOD to fail, or exclude the failing SYSMOD from processing.

HMA372 NPRE ERROR - SYSMOD nnn NPRES yyy BOTH OF WHICH ARE BEING zzz

Explanation:

- nnn - SYSMOD-ID of a function SYSMOD
- yyy - SYSMOD-ID of a function SYSMOD
- zzz - APPLIED or ACCEPTED

Function SYSMOD nnn specifies function SYSMOD yyy as a negative prerequisite using the NPRE operand, but both are concurrently being processed by APPLY or ACCEPT.

System Action: The message is followed by message HMA370. The SMP function terminates.

Programmer Response: Rerun the job, excluding one of the named SYSMODs from processing.

HMA373 SYSMOD nnn HAS MORE THAN ONE APPLICABLE ++VER MODIFICATION CONTROL STATEMENT

Explanation:

- nnn - SYSMOD-ID

During APPLY or ACCEPT, SYSMOD nnn had two or more ++VER modification control statements that specified, on the FMID operand, functions that were either applied or accepted or were concurrently being applied or accepted. As a result, SMP could not determine the value of the FMID to assign to this SYSMOD.

System Action: The SYSMOD is terminated.

Programmer Response: Correct the ++VER modification control statement for the named SYSMOD.

HMA374 rrr SYSMOD nnn CANNOT BE PROCESSED

Explanation:

- rrr - PREREQ or FMID
- nnn - SYSMOD-ID

This message follows message HMA226. The SYSMOD named in HMA226 was terminated because SMP element selection processing could not determine the processing order for the SYSMOD that was terminated and for the SYSMOD(s) named in this message.

The error is illustrated in the following example:

```
++FUNCTION(F000001).

++VER(Z038) PRE(P000001).

++MOD(IEYMYMOD).


++PTF(P000001).
```

++VER(Z038) FMID(F000001).

In this example, the function SYSMOD names a SYSMOD as a prerequisite that cannot be processed until the function is processed. This situation might also occur when two SYSMODs name each other as prerequisites.

System Action: The SYSMOD named by the HMA226 message is terminated.

Programmer Response: Correct the FMID/PREREQ relationship after rejecting the SYSMODs in error. RECEIVE and APPLY or ACCEPT the SYSMODs.

HMA376 COPY PROCESSING FOR SMPTLIB FAILED, SYSMOD=nnn, RC=rc

Explanation:

- nnn - SYSMOD-ID
- rc - the return code from IEBCOPY

IEBCOPY processing for elements on relative files returned a code equal to rc.

System Action: Processing for the SYSMOD is terminated if the return code is not zero or greater than the user-specified or default return code.

Programmer Response: Check the IEBCOPY output listing to determine the cause of error. Correct the error and resubmit the job.

HMA377 SCRATCH FAILED FOR lib ON VOLUME yyy - zzz

Explanation:

- lib - ddname of data set
- yyy - volume serial number

* zzz - reason for failure

An error was encountered while attempting to scratch one of the relative file data sets. The cause of the error is one of the following:

* ERROR CODE = X'nn'
  - See the SCRATCH macro return codes in _OS/VS2 System Programming Library: Data Management_, GC26-3830 or in _OS/VS1 Data Management Services Guide_, GC26-3874.
* DATASET NOT FOUND
  - The data set to be scratched was not found on any of the volumes specified by the SMPTLIB DD statement.

System Action: Processing continues for the SYSMOD for which the data set was allocated.

Programmer Response: Determine cause of error and, if required, scratch the data set by a means other than SMP.

HMA378 ttt AND uuu FOR ELEMENT eee APPEAR IN SAME SYSMOD

Explanation:

* ttt - MOD, MAC, SRC, ZAP, MACUPD, or SRCUPD
* uuu - MOD, MAC, SRC, ZAP, MACUPD, or SRCUPD
* eee - element name

Modifications ttt and uuu for the same element are invalid.

The following table illustrates the combinations of modifications to an element that are invalid in the same SYSMOD:

	MOD	ZAP	SRC	SRCUPD	MAC	UPDTE/ MACUPD
MOD	INV	INV	OK	OK	OK	OK
ZAP	INV	INV	INV	INV	OK	OK
SRC	OK	INV	INV	INV	OK	OK
SRCUPD	OK	INV	INV	INV	OK	OK
MAC	OK	OK	OK	OK	INV	INV
UPDTE/ MACUPD	OK	OK	OK	OK	INV	INV

<u>System Action</u>: The SYSMOD is not received.

<u>Programmer Response</u>: Correct the problem and execute RECEIVE again.

| HMA379 SYSMOD nnn SELECTED FOR yyy
{IS SUPERSEDED | IS IN ERROR | HAS NO APPLICABLE
++VER MODIFICATION CONTROL STATEMENTS | IS DELETED |
HAS HAD RESTORE ATTEMPTED}

<u>Explanation</u>:

- nnn - SYSMOD-ID
- yyy - APPLY or ACCEPT

SYSMOD nnn cannot be applied or accepted for the following reason:

- IS SUPERSEDED - one or more SYSMODs that supersede SYSMOD nnn are already applied or accepted.
- IS IN ERROR - SYSMOD nnn is marked with the ERROR status on the PTS.
- HAS NO APPLICABLE ++VER MODIFICATION CONTROL STATEMENT - SYSMOD nnn was specified in the select/group list, but the ++VER modification control statement it contained did not specify an SREL/FMID that matched the SREL/FMID on the CDS or ACDS.

- IS DELETED - SYSMOD nnn is DELETED and cannot be restored.

- HAS HAD RESTORE ATTEMPTED - SYSMOD nnn has been partially RESTORED and is marked as RESTORE error.

System Action: SYSMOD nnn is terminated.

Programmer Response: If the SYSMOD is marked with the ERROR status, RECEIVE or RESTORE the SYSMOD again and attempt the APPLY or ACCEPT again.

HMA380 SYSMOD nnn SELECTED FOR yyy IS IN ERROR ON THE SMPPTS

Explanation:

- nnn - SYSMOD-ID

- yyy - APPLY or ACCEPT

A SYSMOD, selected for APPLY or ACCEPT processing, was found to be in error on the PTS.

System Action: The named SYSMOD is terminated.

Programmer Response: Receive the SYSMOD again to remove the error condition on the PTS.

HMA381 ttt FROM SYSMOD nnn APPLIES TO ELEMENT eee DELETED BY ANOTHER SYSMOD BEING PROCESSED

Explanation:

- ttt - ZAP, MACUPD, or SRCUPD

- sss - SYSMOD-ID of SYSMOD supplying ttt

- eee - element name

The element to which the ZAP, MACUPD, UPDTE or SRCUPD applied does not exist.

System Action: APPLY or ACCEPT processing for the SYSMOD is terminated.

Programmer Response: Correct the cause of the problem and resubmit the job.

HMA382 ID CHECK PROCESSING SYSMOD nnn {ttt eee|ASSEMBLY aaa FOR ttt eee}

Explanation:

- nnn - SYSMOD supplying element eee.
- eee - element name
- ttt - element type
- aaa - assembly element name

An error or warning condition occurred while validity checking the relationship between an element in SYSMOD nnn and a previously installed or selected version of the same element. This message is followed by other messages describing the validity check condition(s) that failed.

If ASSEMBLY appears in the message, an ID check error exists because of the relationship between an assembly for a source module or macro and the element on the target system that will be replaced by the object module from the assembly.

The severity of this message depends upon the BYPASS options specified on the APPLY, ACCEPT or RESTORE control statement.

System Action: If BYPASS(ID) is not specified, and the SYSMOD was not previously installed on the target system:
The SYSMOD is terminated if it supplies:

- A replacement element if the SYSMOD does not specify, in the PRE or SUP operands, the RMID and all UMIDs of the previously processed version of the element.
- An update element if the SYSMOD does not specify, in the PRE or SUP operands, the RMID of the previously processed version of the element.

The SYSMOD is not terminated if it supplies an update element and specifies, in the PRE or SUP operands, the RMID of the previously processed version of the element, although it does not PRE or SUP all UMIDs of the previously processed version of the element. In this case, HMA319 follows this message, naming the updates in the previously processed version of the element that are not superseded or specified as a prerequisite. The update supplied in this SYSMOD is performed against the previously processed version of the element.

If BYPASS(ID) is not specified and the SYSMOD is a function SYSMOD previously installed on the target system, the SYSMOD is not terminated if SMP can determine that the target system is at a higher level than the SYSMOD that is being re-installed. The target system is considered to be at a higher level if one of the following is true:

(1) The RMID of the element from the SYSMOD differs from the RMID of the element on the target system, and the RMID of the element from the SYSMOD is found on the target system.
(2) The RMID of the element from the SYSMOD is the same as the RMID of the element on the target system and UMIDs are associated with the target system element.

In the first case, HMA382 is not issued. The element from the SYSMOD being re-installed is not processed and the higher level version of the element remains on the target system. In the second case, HMA382 is issued, followed by HMA319, naming the updates to the target system element that are not superseded or specified as prerequisites. The element from the SYSMOD being re-installed is not processed and the higher level element remains on the target system.

- If BYPASS(ID) was specified, the SYSMOD is not terminated for any ID checks reported by message HMA382. The element named is selected for installation, and processing of the SYSMOD continues.

<u>Programmer</u> <u>Response</u>: This message indicates that there is an invalid relationship between an element in the SYSMOD being processed and elements in other SYSMODs installed or being installed on your system. The messages following this message in the output listing should be carefully examined. If the element named is installed, it may regress IBM-supplied service and/or user-supplied modifications included in the SYSMODs named in subsequent HMA319 messages.

You can bypass termination of the SYSMOD by specifying BYPASS(ID) on the APPLY, ACCEPT or RESTORE control statement however, the modifications included in the SYSMODS named in subsequent HMA319 messages are potentially lost. An attempt should be made to re-work and re-install these modifications.

HMA383 FUNCTION xxx (FMID yyy) SUPERSEDES FUNCTION aaa (FMID bbb) BUT THE FMIDS ARE NOT EQUAL

Explanation:

- xxx - SYSMOD-ID of a function SYSMOD
- yyy - value of the FMID for SYSMOD-ID xxx
- aaa - SYSMOD-ID of a function SYSMOD
- bbb - value of the FMID for SYSMOD-ID aaa

Function SYSMOD xxx is concurrently being processed with function SYSMOD aaa, and SYSMOD xxx supersedes SYSMOD aaa. However, the two SYSMODs have different FMID values, which causes incorrect SYSMOD selection.

System Action: SYSMOD xxx is terminated. This message is followed by message HMA370, which terminates the SMP function.

Programmer Response: Correct the ++VER modification control statements for one of the function SYSMODs.

HMA384 DELETE FUNCTION xxx IS SUPERSEDED BY FUNCTION yyy

Explanation:

- xxx - SYSMOD-ID of function SYSMOD
- yyy - SYSMOD-ID of function SYSMOD

Function SYSMOD yyy supersedes function SYSMOD xxx, and both are being concurrently processed. However, SYSMOD xxx deletes other functions.

System Action: This message is followed by message HMA370, which terminates the SMP function.

Programmer Response: Exclude one of the SYSMODs from processing.

HMA385 SYSMOD nnn TERMINATED BY USER EXIT RETURN CODE

Explanation:

- nnn - SYSMOD-ID

The SMP user exit procedure returned a return code of 8 or greater.

System Action: SYSMOD nnn is not received.

Programmer Response: None

HMA386 SYSMOD nnn ALREADY RECEIVED

Explanation:

- nnn - SYSMOD-ID

A SYSMOD entry was found on the PTS for SYSMOD nnn with the ERROR indicator off; this entry represents a successfully received SYSMOD.

System Action: SYSMOD nnn is not received.

Programmer Response: To receive the new version of the SYSMOD, reject the SYSMOD from the PTS using the REJECT control statement and execute RECEIVE against the new version of the SYSMOD.

HMA387 SYSMOD nnn NOT SELECTED

Explanation:

- nnn - SYSMOD-ID

The named SYSMOD was found in PTFIN but you did not specify it as a value of the SELECT operand.

System Action: SYSMOD nnn is not received.

Programmer Response: None.

HMA388 SYSMOD nnn EXCLUDED

Explanation:

- nnn - SYSMOD-ID

The named SYSMOD was found in PTFIN but you specified it as a value of the EXCLUDE operand.

System Action: SYSMOD nnn is not received.

Programmer Response: None.

HMA389 SYSMOD nnn HAS NO APPLICABLE ++VER MODIFICATION CONTROL STATEMENT

Explanation:

- nnn - SYSMOD-ID

SYSMOD nnn did not have a ++VER modification control statement that named an SREL and/or an FMID that is in the PTS SYSTEM entry.

System Action: SYSMOD nnn is not received.

Programmer Response: To receive the SYSMOD, you might specify BYPASS(FMID) on the RECEIVE control statement, or you might update the PTS SYSTEM entry using a UCL statement to include the required SREL and/or FMID.

If SYSMOD nnn is a service SYSMOD (++PTF, ++USERMOD, or ++APAR), it must include at least one ++VER modification control statement with an FMID, or PARM='FMID=xxxxxx' must be specified when SMP is executed, where xxxxxx is the SYSMOD-ID of a function.

HMA390 SYSMOD nnn SELECTED BUT COULD NOT BE RECEIVED

| Explanation:

- nnn - SYSMOD-ID

SYSMOD nnn was specified as an operand of the SELECT keyword on the RECEIVE control statement, but it was not successfully RECEIVED.

System Action: RECEIVE processing continues.

Programmer Response: Refer to preceding messages in the output listing to determine why the SYSMOD was not received.

HMA391 SYSMOD nnn TERMINATED DURING PROCESSING OF RELFILE ELEMENTS

Explanation:

- nnn - SYSMOD-ID

Errors occurred attempting to load elements supplied in IEBCOPY unloaded data sets to the TLIB data sets.

The following conditions cause termination during RECEIVE processing of relative files:

- Unable to position the PTFIN data set because of I/O error or an invalid data set name on a PTFIN relative file. See 'Relative File Technique' in Chapter 2 for the correct naming conventions.

- Unable to allocate a data set on the TLIB volume

- Non zero IEBCOPY return code (examine IEBCOPY SYSPRINT output to determine the cause).

System Action: SYSMOD nnn is not received. RECEIVE processing terminates. The TLIBs are scratched and the SYSMOD is deleted from the PTS.

Programmer Response: Correct any errors and attempt to receive the SYSMOD again.

HMA392 SYSMOD nnn NOT RECEIVED

Explanation:

- nnn - SYSMOD-ID

There is no valid SYSMOD and/or MCS entry on the PTS that represents SYSMOD nnn.

System Action: The SYSMOD is not received.

Programmer Response: Refer to preceding messages in the output listing to determine why the SYSMOD was not received.

HMA393 SYSMOD nnn SUCCESSFULLY RECEIVED

Explanation:

- nnn - SYSMOD-ID

The SYSMOD was successfully received. A SYSMOD and an MCS entry for SYSMOD nnn have been created on the PTS.

System Action: None.

Programmer Response: None

HMA394 SYSMOD nnn RELFILE ELEMENTS LOADED [MAX COPY RETURN CODE rc]

Explanation:

- nnn - SYSMOD-ID
- rc - the maximum non zero return code from copy

The elements supplied in unloaded copy data sets for the named SYSMOD were successfully loaded to a TLIB data set for subsequent processing by APPLY and/or ACCEPT. If 'MAX COPY RETURN CODE' appears, copy processing returned a non-zero return code less than or equal to the acceptable return code that you specified in the PTS SYSTEM entry.

System Action: Processing of the SYSMOD continues. The SYSMOD entry on the PTS is updated to indicate that SYSMOD nnn is successfully received.

Programmer Response: The copy SYSPRINT output should be examined to determine the cause of the non zero return code so that subsequent processing is not adversely affected.

HMA395 SYSMOD nnn HAS RELFILE ELEMENTS

Explanation:

- nnn - SYSMOD-ID

The named SYSMOD supplied some of its elements in an unloaded IEBCOPY data set in a subsequent file on the PTFIN data set.

System Action: If no errors were encountered; that is, HMA392 SYSMOD nnn NOT RECEIVED does not appear along with this message, the elements supplied in unloaded IEBCOPY data sets will subsequently be loaded.

Programmer Response: Look for the pair of messages:

HMA394 SYSMOD nnn RELFILE ELEMENTS LOADED

and HMA393 SYSMOD nnn SUCCESSFULLY RECEIVED

after all modification control statements for all SYSMODS are listed in SMPOUT.

HMA396 SYSMOD nnn HAS NO ++VER MODIFICATION CONTROL STATEMENT

Explanation:

- nnn - SYSMOD-ID

No ++VER modification control statement was found for SYSMOD nnn.

System Action: SYSMOD nnn is not received.

Programmer Response: To receive the SYSMOD, include an applicable ++VER modification control statement and execute

RECEIVE again.

HMA397 SYSMOD nnn HAS NO ELEMENTS

Explanation:

- nnn - SYSMOD-ID

SYSMOD nnn has no ++MOD, ++MAC, ++SRC, ++ZAP, ++MACUPD, nor ++SRCUPD modification control statements. It does, however, have an applicable ++VER and may have inline JCL.

System Action: SYSMOD nnn is received.

Programmer Response: None

HMA398 SYSMOD nnn SYNTAX OR CONSTRUCTION ERROR

Explanation:

- nnn - SYSMOD-ID

A modification control statement syntax error or a SYSMOD construction error was detected by RECEIVE processing.

System Action: SYSMOD nnn is not received.

Programmer Response: More specific information can be found regarding the syntax or construction error by scanning the SMPOUT stream for SYSMOD nnn.

HMA399 ENTER JULIAN DATE OR 'U' FOR HMASMP

Explanation: The date to be used in recording this SMP job is requested.

System Action: None.

Programmer/Operator Response: Enter the date as yyddd (yy=year, ddd=day) or reply with 'U' for the system IPL date.

HMA400 SYSMOD nnn ENCOUNTERED PREVIOUSLY ON SMPPTFIN

Explanation:

- nnn - SYSMOD-ID

The named SYSMOD appeared previously in the PTFIN input stream. The previously encountered version of this SYSMOD may or may not have been successfully received.

System Action: This occurrence of the SYSMOD is not received. The previously encountered version is not affected.

Programmer Response: If the earlier occurrence of the SYSMOD is the desired SYSMOD and it was successfully received, no action is required. If this occurrence of the SYSMOD is desired, the SYSMOD must be rejected, and the PTFIN input stream must be corrected so that only the desired SYSMOD appears.

HMA401 SYSMOD nnn SELECTED FOR yyy NOT FOUND ON lib LIBRARY

Explanation:

- nnn - SYSMOD-ID
- yyy - APPLY, ACCEPT, or RESTORE
- lib - the ddname of the data set

For APPLY or ACCEPT, SYSMOD nnn was selected but was not found on the PTS. For RESTORE, SYSMOD nnn was not found on the CDS.

System Action: The SYSMOD is terminated.

Programmer Response: Ensure that the SYSMOD-ID is correctly specified on the control statement.

HMA402 xxx IS INVALID FOR {SUPERSEDED ONLY | DELETED} SYSMOD

Explanation:

- xxx - the UCL keyword

If SUPERSEDED ONLY results, UCL processing produced a SYSMOD entry that did not contain any MOD, MAC, or SRC subentries. SMP assumes that this SYSMOD entry is produced as a result of being superseded by another SYSMOD. However, your UCL request also either added or left the data specified by xxx in the SYSMOD entry. This data is only valid for SYSMODs that are not superseded.

If DELETED results, the DELETE operand was specified in the UCL request; however, you also specified other operands which is not allowed.

System Action: The requested UCL processing is not performed.

Programmer Response: Correct your UCL statement by either supplying a MOD, MAC, or SRC subentry, or delete xxx from the SYSMOD (if a DEL request), or do not specify xxx (if an ADD or a REP request), and rerun the UCL statement.

| HMA403 ENQ FAILED FOR DATASET ddd SYSMOD

Explanation:

- ddd - dataset name

SMP issued an exclusive ENQ on the datset identified, but the dataset was not available.

System Action: Processing for the SYSMOD requiring the datset is terminated.

Programmer Response: Rerun the job for the affected SYSMOD when the dataset is available for exclusive use.

HMA404 xxx yyy TO BE RESTORED TO SMPCDS FROM SMPSCDS FOR SYSMOD nnn NOT FOUND ON zzz

Explanation:

- xxx - entry type
- yyy - entry name
- nnn - SYSMOD-ID
- zzz - ddname of data set

While attempting to restore SYSMOD nnn, SMP tried to copy the specified entry type and name from the SCDS to the CDS, but the specified member was not found on the data set identified by zzz.

System Action: If the entry was not found on the CDS, the entry from the SCDS is placed on the SMPCDS. If the entry was not found on the SCDS, backup was not possible. In both

cases, processing for the SYSMOD continues.

Programmer Response: Determine the cause of the error by examining the LOG to see if UCL processing was performed for the member. Use 'LIST CDS' for the copied member to ensure that the correct version of the entry was copied. If the entry is incorrect, make the appropriate updates using a UCL statement.

HMA405 DISTLIB ERROR FOR type name FROM SYSMOD nnn

Explanation:

- type - the entry type
- name - the entry name
- nnn - SYSMOD-ID

The DISTLIB specified on the element modification control statement in SYSMOD nnn differs from the distribution library found for the element on the CDS or ACDS.

System Action: APPLY/ACCEPT processing for SYSMOD nnn is terminated.

Programmer Response: Correct the distribution library on either the element modification control statement or on the ACDS or CDS.

HMA406 START OF SMPADDIN CONTROL STATEMENTS

Explanation:

Control statements from SMPADDIN follows.

System Action: SMPADDIN statements are processed.

Programmer Response: None

| HMA407 END OF SMPADDIN CONTROL STATEMENTS

Explanation:

End of control statements from SMPADDIN follows.

System Action: UNLOAD processing continues.

Programmer Response: None

HMA408 SYSMOD nnn NOT APPLIED OR NOT ACCEPTED

Explanation:

- nnn - SYSMOD-ID

SYSMOD nnn was selected for REJECT processing, and had either been applied or accepted, but not both. The APP or ACC indicator in the PTS SYSMOD entry is not set.

System Action: Processing continues for this SYSMOD.

Programmer Response: None.

HMA409 COPY {FAILED|SUCCESSFUL} -type=name -LIBRARY=lib -SYSMOD=nnn -RETURN CODE=rc

Explanation:

- type - the entry type
- name - the entry name
- lib - ddname of data set
- nnn - SYSMOD-ID
- rc - return code

Copy processing completed for the indicated member and data set with a return code equal to rc.

Multiple members are copied to various data sets in one invocation of copy. In the event of a failure, all members must be considered to have failed. SMP marks all related SYSMODs with the ERROR status set.

System Action: Processing of the SYSMOD is terminated if the return code is not zero or greater than the user specified or default return code.

Programmer Response: If the copy failed and the SYSMODs have the ERROR status set, check the copy output listing to determine the cause of error. Correct the error and resubmit the job.

HMA410 SYSMOD nnn AND ttt ZAP MODULE mmm.

Explanation:

- nnn - SYSMOD-ID of SYSMOD supplying ZAP for module
- ttt - SYSMOD-ID of SYSMOD supplying ZAP for module
- mmm - the module name

SMP APPLY and ACCEPT will not process more than one ZAP to the same element during one APPLY or ACCEPT pass.

System Action: APPLY or ACCEPT processing terminates for SYSMOD nnn.

Programmer Response: Apply or accept SYSMOD nnn after SYSMOD ttt is applied or accepted.

HMA411 xxx yyy TO BE RESTORED TO SMPCDS FROM SMPSCDS FOR SYSMOD nnn HAS BEEN MODIFIED BY SUBSEQUENT SYSMOD mmm

Explanation:

- xxx - entry type
- yyy - entry name
- nnn - SYSMOD-ID
- mmm - SYSMOD-ID

While attempting to restore SYSMOD nnn, SMP tried to copy the specified entry type and name from the SCDS to the CDS. The entry on the CDS in the LAST UPDATE field indicated that SYSMOD mmm has been processed and has made modifications to the entry using inline JCLIN or UCLIN.

System Action: The entry is not copied from the SCDS to the CDS. Processing for the SYSMOD continues.

Programmer Response: Issue 'LIST CDS' to list the specified entry. Ensure that the entry on the CDS is valid even after the SYSMOD is restored. If any modifications are required, use a UCL statement to make the changes.

HMA412 IN THE CURRENT ENVIRONMENT THE RELATIONSHIP BETWEEN THE FOLLOWING SET(S) OF SYSMODS IS INCORRECT OR AMBIGUOUS

Explanation: When determining the order in which SYSMODS should be processed, SMP was unable to establish an order for the SYSMODs listed in a subsequent message. The current environment includes those SYSMODs already on the system and those currently being processed. Processing is determined by the information on the ++VER modification control statement (FMID, VERSION, PRE, SUP).

System Action: Processing is terminated for the function.

Programmer Response: Correct the ++VER modification control statement for the specified SYSMODs, or selectively process each SYSMOD in the order required.

HMA413 SYSMOD=nnn FMID=yyy PRE=zzz

Explanation:

- nnn - SYSMOD-ID
- yyy - the value of the FMID
- zzz - a list of prerequisite SYSMODs

This message follows message HMA412 and lists the SYSMODs for which SMP could not determine a processing order.

System Action: Processing is terminated for the function.

Programmer Response: See message HMA412.

HMA414 lib DIRECTORY SUCCESSFULLY LOADED FOR IN STORAGE UPDATE OPERATIONS

Explanation:

- lib - ddname of SMP data set

The directory for the specified data set was loaded into storage. All updates to this directory will be done only to the in-storage copy. In-storage processing does not occur until message HMA368 is issued.

System Action: Processing continues in an in-storage only mode.

Programmer Response: None.

HMA415 ELEMENT eee DOES NOT EXIST ON ccc FOR uuu FROM SYSMOD nnn

Explanation:

- eee - element name
- ccc - SMPCDS or SMPACDS
- uuu - ZAP, MACUPD, or SRCUPD
- nnn - SYSMOD-ID

The update (ZAP, MACUPD or SRCUPD) for the named element cannot be accomplished because there is no element entry on the CDS or ACDS representing the element to be updated. Note that this situation can arise if the SYSMOD that supplied the element was terminated abnormally; in this case, there may be an element entry on the ACDS or CDS that has no FMID and no RMID.

System Action: APPLY or ACCEPT processing is terminated for SYSMOD nnn.

Programmer Response: List the CDS or ACDS to determine whether there is an entry for the element. If an entry is found with no RMID, the entry represents an element that is not considered to be in the target system. Either install a SYSMOD supplying the element or use UCLIN to properly initialize the FMID and RMID fields in the CDS or ACDS entry.

HMA418 INLINE JCLIN PROCESSING {FAILED|SUCCESSFUL} FOR SYSMOD nnn

Explanation:

- nnn - SYSMOD-ID

SMP completed JCLIN processing for the indicated SYSMOD. Processing either completed successfully or failed.

System Action: If JCLIN completed successfully, processing continues for the SYSMOD. If JCLIN processing failed, processing is terminated for the SYSMOD.

Programmer Response: No action is required if processing completed successfully. If processing failed, determine the cause of failure by looking at previous messages from JCLIN processing. Restore the SYSMOD, correct the inline JCLIN element, and receive and apply the SYSMOD again.

HMA419 ++JCLIN MODIFICATION CONTROL STATEMENT NOT FOUND IN SYSMOD nnn MCS ENTRY

Explanation:

- nnn - SYSMOD-ID

The indicator that inline JCLIN is present is set in the PTS SYSMOD entry. However, SMP could not find the ++JCLIN modification control statement in the PTS MCS entry.

System Action: Processing is terminated for the SYSMOD.

Programmer Response: Probable user error. Ensure that the PTS MCS entry was not modified after the SYSMOD was received. Reject the SYSMOD and correct the error by either removing the ++JCLIN modification control statement, or adding one. Receive the SYSMOD again so that the status indicators in the PTS SYSMOD entry will reflect the contents of the PTS MCS entry.

HMA420 REQUISITE SYSMODS BYPASSED FOR SYSMOD nnn

Explanation:

- nnn - SYSMOD-ID

Specifying the BYPASS option on the APPLY or ACCEPT control statement caused SYSMOD termination to be bypassed even though certain requisite conditions were not satisfied. Message HMA359 follows this message and names the requisites which were not satisfied.

System Action: APPLY or ACCEPT processing continues for the named SYSMOD.

Programmer Response: None.

HMA421 SYSMOD nnn TERMINATED BECAUSE NEGATIVE PREREQUISITE SYSMOD mmm FOUND ON lib

Explanation:

- nnn - SYSMOD-ID of failing SYSMOD
- mmm - SYSMOD-ID of negative prerequisite SYSMOD
- lib - ddname of SMP data set

During APPLY or ACCEPT processing of SYSMOD nnn, SYSMOD mmm, which was specified as a negative prerequisite by SYSMOD nnn, was determined to have been already applied.

System Action: SYSMOD nnn is terminated.

Programmer Response: The two SYSMODs cannot both be applied or accepted. You can remove SYSMOD nnn from the PTS using the REJECT control statement, or exclude it from processing using the EXCLUDE operand. Alternately, SYSMOD mmm can be removed from the system libraries using the RESTORE or the UCLIN control statements.

HMA422 MULTIPLE NAME CARDS FOUND IN LMODIN INPUT

| Explanation:

During processing of the UCLIN ++LMODIN control statement more then one name card was encountered.

| System Action: The UCLIN changes requested are not performed.

| Programmer Response: Change the UCLIN ++LMODIN control statement so that they apply only to that one LMOD identified by the UCLIN change request. If more then one LMOD is to be changed then break the ++LMODIN statement into those applicable to only one LMOD and then rerun the UCLIN to the multiple LMOD's.

HMA423 THE DATE RANGE SPECIFIED IS INVALID

Explanation: An incorrect date range has been detected.

System Action: LIST processing is terminated.

Programmer Response: Correct the date range. Specify mm as 01 through 12, dd as 01 through 31, and yy as 00 through 99 on the LIST LOG control statement.

HMA424 HMASMP EXEC PARM=xxxx

Explanation:

- xxxx - Parameters specified on the EXEC statement

This message lists any parameters specified on the EXEC statement.

System Action: None.

Programmer Response: None.

HMA425 xxx KEYWORD REQUIRED WHEN OTHER QUALIFYING KEYWORDS ARE SPECIFIED

Explanation:

- xxx - required keyword

The xxx keyword was specified on an SMP control statement, but it is only valid when specified with another keyword. For example, LIST CDS PTF is invalid because the SYSMOD keyword is omitted. To correct the error, LIST CDS SYSMOD PTF should have been specified.

System Action: The control statement is not executed.

Programmer Response: Specify the required keyword and resubmit the job.

HMA426 DELETE ERROR - SYSMOD nnn DELETES yyy WHICH IS ALREADY {APPLIED|ACCEPTED}

Explanation:

- nnn - SYSMOD-ID
- yyy - SYSMOD-ID

During APPLY or ACCEPT processing, SYSMOD nnn, which was not in the select/group list, deleted SYSMOD yyy, which was already applied or accepted.

System Action: The SYSMOD was terminated. This message is followed by HMA370, indicating that the SMP function also fails.

Programmer Response: If you do not wish to process SYSMOD nnn, reject it using the REJECT control statement, or exclude it from processing using the EXCLUDE operand. If you want to apply or accept SYSMOD nnn, specify it using the SELECT or GROUP operands.

HMA427 NPRE ERROR - SYSMOD nnn NPRES yyy
WHICH IS ALREADY {APPLIED|ACCEPTED}

Explanation:

- nnn - SYSMOD-ID
- yyy - SYSMOD-ID

During APPLY or ACCEPT processing, SYSMOD xxx specified SYSMOD yyy as a negative prerequisite, but SYSMOD yyy was already applied or accepted.

System Action: SYSMOD xxx is terminated. This message is followed by HMA370 to indicate that the SMP function fails.

Programmer Response: If you do not want to process SYSMOD nnn, EXCLUDE it or REJECT it from the PTS. If you wish to apply or accept SYSMOD nnn, you must remove SYSMOD yyy. If APPLY is being done, you can remove SYSMOD yyy using the RESTORE control statement. However, if the function is ACCEPT, the only was to remove SYSMOD yyy is using the UCLIN control statement.

HMA428 RESTORE CANDIDATE nnn TERMINATED BECAUSE IT
{IS NOT APPLIED|HAS BEEN ACCEPTED|
IS DELETED|IS SUPERSEDED|DELETES SYSMODS}

Explanation:

- nnn - SYSMOD-ID

During RESTORE processing, SYSMOD nnn was found to be in the stated condition.

System Action: The SYSMOD was terminated.

Programmer Response: If you do not wish to process SYSMOD nnn, do not SELECT SYSMOD nnn or group SYSMODs that contain SYSMOD nnn. If you wish to restore SYSMOD nnn, note the following:

- IS NOT APPLIED - apply SYSMOD nnn so that it can be restored.

- HAS BEEN ACCEPTED - SYSMOD nnn cannot be restored.
- IS DELETED - SYSMOD nnn cannot be restored.
- IS SUPERSEDED - The SYSMOD can be restored if it is removed from the SELECT list and GROUP is specified that contains SYSMOD nnn.
- DELETES SYSMODS - SYSMOD nnn cannot be restored.

HMA429 name FOR SYSMOD nnn IS ALSO IN SYSMOD mmm WHICH IS {IN ERROR AND NOT BEING RESTORED|APPLIED BUT NOT ACCEPTED}

Explanation:

- name - the element name
- nnn - SYSMOD-ID
- mmm - SYSMOD-ID

SYSMOD mmm on the CDS contains the same element name as SYSMOD nnn but is either marked in ERROR or is applied and not being restored.

System Action: SYSMOD nnn is terminated.

Programmer Response: Correct the problem by either restoring SYSMOD mmm along with SYSMOD nnn or accept SYSMOD mmm.

HMA430 lib id list {IS DELETED | IS SUPERSEDED | IS APPLIED | IS NOT APPLIED | IS NOT BEING RESTORED | IS BEING RESTORED | IS APPLIED BUT NOT ACCEPTED | IS ACCEPTED IN ERROR | IS ACCEPTED BUT NOT APPLIED | DOES NOT PRE/SUP SMPACDS MODID}

Explanation:

- lib - SMPCDS or SMPACDS
- id - FMID, RMID, or UMID

- list - a list of modids

This message further explains the modid error defined by HMA382 during RESTORE processing. The errors are as follows:

- IS DELETED - A CDS or ACDS SYSMOD named in the modid list is deleted SYSMOD. This can only happen when a previous APPLY or ACCEPT failed while updating the CDS or ACDS entries for the SYSMOD, or when the element or SYSMOD entry was modified using the UCLIN control statement.
- IS SUPERSEDED - A CDS or ACDS SYSMOD named in the modid list is a superseded SYSMOD. This can only happen when a previous APPLY or ACCEPT fails while updating the CDS or ACDS entries for the SYSMOD, or when the element or SYSMOD entry was modified using the UCLIN control statement.
- IS APPLIED - A CDS modid SYSMOD is applied but not accepted and is not being restored.
- IS NOT APPLIED - An ACDS modid SYSMOD is accepted but not applied.
- IS NOT BEING RESTORED - A CDS modid SYSMOD is applied but not accepted and is not being restored.
- IS BEING RESTORED - A CDS modid SYSMOD is not being restored, but there is no corresponding type of modid (that is, FMID, RMID, or UMID) in the ACDS modid list.
- IS APPLIED BUT NOT ACCEPTED - A CDS modid SYSMOD is not being restored and does not appear in the ACDS modid list.
- IS ACCEPTED IN ERROR - A SYSMOD in the ACDS modid list is accepted in error.
- IS ACCEPTED BUT NOT APPLIED - A SYSMOD in the ACDS modid list is not applied.
- DOES NOT PRE/SUP SMPACDS MODID - The CDS RMID or UMID SYSMOD does not have a correct PRE or SUP relationship with any ACDS RMID or UMID. To be correct, one of the following must be true:

The named SYSMOD must PRE or SUP an ACDS RMID or UMID.

The named SYSMOD must PRE or SUP another SYSMOD that is concurrently being restored and has a correct PRE or SUP relationship with an ACDS RMID or UMID.

System Action: The SYSMOD specified in HMA382 is terminated unless BYPASS(MODID) was specified.

Programmer Response: If the wrong set of SYSMODs is being processed, change the select or group list to process the correct set. If, however, the correct set is specified, use BYPASS(MODID) to RESTORE the SYSMODs.

HMA431 INSUFFICIENT INFORMATION AVAILABLE TO DETERMINE TARGET LIBRARY - type=name - SYSMOD=nnn

Explanation:

- type - element type
- name - element name
- nnn - SYSMOD-ID

During APPLY or ACCEPT processing, an element was encountered whose target library (system library for APPLY, distribution library for ACCEPT) could not be determined from the CDS for APPLY or ACDS for ACCEPT.

System Action: APPLY or ACCEPT processing terminates for the named SYSMOD.

Programmer Response: Provide the necessary JCLIN or UCLIN information so the SMP can create the necessary CDS or ACDS entries to complete the processing for the element.

HMA432 ABNORMAL TERMINATION - CODE = ttt ccc - PROGRAM = ppp

Explanation:

- ttt - SYSTEM or USER code
- ccc - code number
- ppp - program name

An abnormal termination occurred and the SMP recovery routine has been invoked.

System Action: If directories in-storage for write processing mode was in effect, the directories of the affected SMP data sets were re-written to disk. The SMP summary reports are generated with the current element and SYSMOD status.

Programmer Response: Examine the SMP summary reports to determine the element and SYSMOD status. Examine the ABEND dump for problem determination.

HMA433 SYSMOD nnn IS SUPERSEDED BY mmm WHICH IS NOT BEING RESTORED

Explanation:

- nnn - SYSMOD-ID of superseded SYSMOD that is being restored
- mmm - SYSMOD-ID of superseding SYSMOD

SYSMOD nnn is part of a restore group. Because it is superseded by SYSMOD mmm, SYSMOD mmm is also considered to be part of the restore group. Since SYSMOD mmm is not being restored, SYSMOD nnn cannot be restored.

System Action: SYSMOD nnn is terminated.

Programmer Response: Either eliminate SYSMOD nnn from RESTORE processing, or include SYSMOD mmm.

Appendix A: Rules for Coding SMP Statements

Use the following rules to code SMP control statements and modification control statements:

1) Each statement must begin on a new logical 80-byte record.

2) The symbol '++' in the modification control statement must appear in bytes 1 and 2. Except for this restriction, the control statements and modification control statements can begin and end anywhere up to and including byte 72.

3) The statement function must be specified first, followed by any keywords.

4) The optional keywords can be coded in any sequence, except where noted in the syntax and operand descriptions.

5) At least one blank must occur between each keyword.

6) Blanks or a comma, as specified in the syntax, must separate the keywords and their options.

7) Comments are delineated by '/*' at the beginning and '*/' at the end. A comment can appear anywhere on a statement before the ending period, but should not begin in column 1.

8) Each statement must be terminated with a period(.).

9) Bytes 73-80 are ignored by SMP.

10) A statement continues until a period is encountered, and the statement can continue on more than one physical record. Continuation is assumed if no period(.) is found before byte 73.

11) SMP completes processing one statement before the next statement is processed.

| HMA434 ttt eee - SYSMOD=sss WILL NOT UPDATE SYSTEM LIBRARIES

Explanation:

- ttt - element type (MOD or ASSEMBLY)
- eee - element name
- sss - SYSMOD id

The named element will not be applied to any target system libraries.

System Action: Processing of the named SYSMOD continues.

Programmer Response: If the element should be applied to a target system library, insure that proper JCLIN has been run to define to SMP the target system library to which the element should be moved. Run the JCLIN ane re- apply the SYSMOD. If the element does not belong on a target system library, no response is required.

| HMA435 VERNUM MUST BE ENTERED BEFORE OPTION REQUIRING VERNUM

Explanation:

The VERNUM option was specified after one of the options that required VERNUM in the SYSMOD entry.

EXAMPLE:

REP SYSMOD(UZ00001) PRE(UZ00001) VERNUM(003)

This is an error because the PRE option has a VERNUM associated with it but the VERNUM option is specified after the PRE option.

System Action: The UCLIN changes requested are not performed.

Programmer Response: Change the order of options so that the VERNUM option is first.

| HMA436 UPDATE RESULTS IN SYSMOD WITH MULTIPLE VERNUM VALUES

Explanation:

The VERNUM option was specified but the resulting update caused the SYSMOD to contain subentries with different VERNUM values.

System Action: The UCLIN changes requested are not performed.

Programmer Response: Check the existing subentries in the SYSMOD for their VERNUM values, then rerun the UCLIN soecifying the existing VERNUM values; or replace all the existing subentries requiring the VERNUM value.

| HMA437 ESTAE-STAE PROCESSING TERMINATED WITH A RETURN CODE=XX.

Explanation:

The error accured in the ESTAE-STAE processor. The return code from the the ESTAE-STAE processor was XX.

System Action: System processing was terminated due to a non zero return code from the ESTAE-STAE processing.

Programmer Response: Refer to the following manuals for a further explanation of the XX return code from the ESTAE-STAE processor.

- OS/VS1 Supervisor Services and Macro Instruction Manual. (STAE PROCESSING)
- OS/VS2 Supervisor Services and Macro Instruction Manual. (ESTAE PROCESSING)

| HMA438 UNABLE TO INITIALIZE UTILITY INTERFACE SUBTASK - REASON CODE=XX (- RETURN CODE=YY)

Explanation:

- xx - Two digit code which indicates why the subtask could not
  be initialized (see below)

- yy - Two digit return code associated with reason codes 01, 02, 03.
  Reason code 04 has no associated return code.

The SMP4 subtask which interfaces with UTILITIES programs could not be initialized for one of the following reasons.

a. The IDENTIFY for the entry point of the subtask program received a return code greater than 4. The return code given with this reason code is the return code form IDENTIFY. See OS/VS2 Supervisor Services And Macro Instructions (GC28-0756) or OS/VS1 Supervisor Services Macro Instructions (GC24-5103) dependent on the system on which SMP4 is being executed for an explanation of the IDENTIFY return codes.

b. The ATTACH for the subtask received a non-zero return code. The return code given with this reason code is from ATTACH. See OS/VS2 Supervisor Services And Macro Instructions (GC28-0756) or OS/VS1 Supervisor Services Macro Instructions (GC24-5103) dependent on the system on which SMP4 is being executed for an explanation of the ATTACH return codes.

c. Subtask initialization failed since the subtask could not establish its STAE coverage when running under VS/1 or its ESTAE coverage when running under VS/2. The return code given with this reason code is from STAE or ESTAE. See OS/VS2 Supervisor Services And Macro Instructions (GC28-0756) or OS/VS1 Supervisor Services Macro Instructions (GC24-5103) dependent on the system on which SMP4 is being executed for an explanation of the ATTACH return codes.

d. Subtask initialization failed since the CONTROL PROGRAM under which SMP4 was executing was not VS/1 or VS/2. There is no return code associated with this reason code.

System Action: SMP4 Terminates.

Programmer Response: Correct the mistake and rerun the job.

| HMA439 RETRY (FAILED | SUCCESSFUL)

Explanation:

A RETRY operation was performed after a B37-04, D37-04 or E37-04 ABEND was encountered on a utility target library. If an error occurs during the RETRY operation, the RETRY

'FAILED'; otherwise the RETRY is considered 'SUCCESSFUL'.

System Action:

If the RETRY 'FAILED', the SMP function is terminated. (SMP MAY ALSO BE TERMINATED DEPENDENT ON WHERE AND HOW THE RETRY OPERATION FAILED.) If the RETRY is 'SUCCESSFUL' SMP processing continues normally.

Programmer Response:

If the RETRY was 'SUCCESSFUL', no action is required. If the RETRY 'FAILED', the size of the target library should probably be increased.

| HMA440 SMP SUBTASK ABNORMAL TERMINATION - CODE=TTTTTT CC-RR PROGRAM=PPPPPPPP - (RETRY WILL BE ATTEMPTED | RETRY WILL NOT BE ATTEMPTED - REASON=XX)

Explanation:

- TTTTTT - Indicates the type of abend encountered. It is either 'SYSTEM' or 'USER'.
- CCC - Indicates the ABEND code in Hexadecimal.
- RR - Indicates the ABEND reason code in Hexadecimal.
- PPPPPPPP- Indicates the name of the UTILITY program invoked prior to the abnormal termination.
- XX - Two digit code which indicates why the RETRY could not be attempted (see below)

The SMP SUBTASK which interfaces with UTILITY programs ABENDED. The type of abend ('SYSTEM' or 'USER'), ABEND code, ABEND reason code, and the UTILITY program name being executed are given in the message. IF a RETRY is not to be attempted, the reason code indicates why as follows:

1) No SDWA (SYSTEM DIAGNOSTIC WORK AREA) was provided to the STAE-ESTAE processor.

2) A user ABEND accurred.

3) RETRY was not indicated for the SMP function.

4) RETRY was already in progress when the ABEND accurred.

5) The ABEND code was not a B37, D37, or E37 or the ABEND reason code was not '04'.

6) The data set which caused the ABEND was not a candidate for RETRY as specified by the user.

7) The data set which caused the ABEND was not the target library of the invoked UTILITY.

System Action:

If a RETRY is to be attempted, RETRY processing occurs. If a RETRY is not to be attempted, the SMP function is terminated with a return code of 12.

Programmer Response:

If a RETRY is to be attempted, the user's action is dependent on the success or failure of the RETRY. If the RETRY is not being attempted then the corrective action is dependent on the ABEND code and the reason for not attempting the RETRY

Appendix B: Syntax Notation Conventions

This publication uses the following syntax notation conventions to define the SMP control statements and modification control statements:

1) Uppercase letters, numbers, and the set of symbols listed below should be coded in an actual statement exactly as shown in the statement definition.

apostrophe	'
asterisk	*
blank	blanks are not coded
comma	,
equal sign	=
parentheses	( )
period	.

2) Lowercase letters and symbols should not be coded; they represent variables for which you should substitute specific information in the actual statement.

   Example: If 'name' appears in a statement definition, you should substitute a specific value (for example, ALPHA) for the variable when you code the statement.

3) Hyphens join lowercase words and symbols to form a single variable, and should never be coded in an actual statement.

   Example: If 'member-name' appears in a statement definition, you should substitute a specific value (for example, BETA) for the variable when you code the statement.

4) An underscore indicates a default option, and should never be coded in an actual statement. If you select an underscored alternative, you need not specify that alternative when you code the statement.

Example: The representation

A | B | <u>C</u>

indicates that you are to select A or B or C. However, if you do not specify anything, C is chosen because it is the default.

5) Braces group required items and should never be coded in an actual statement. One of the items enclosed within the braces must be selected.

Example: The representation

ALPHA={ ADD | DEL | REP }

indicates that you must choose one of the items ADD, DEL, or REP.

6) Brackets group optional items and should never be coded in an actual statement. Only one of the items enclosed within the brackets must be selected, or you should not specify the keyword at all.

Example: The representation

ALPHA=[A | B | C]

indicates that you can choose one of the items A, B or C, or you must omit the keyword entirely.

7) An ellipsis indicates that the preceding item or group of items can be repeated more than once in succession, and should never be coded in an actual statement.

Example: The representation

ALPHA[option[,option]...]

indicates that ALPHA can appear alone or can be followed by an option any number of times in succession.

8) A slash represents 'or', and should never be coded in an actual statement.

Example: The representation

A | B | C

indicates that you are to select A or B or C.

Appendix C: PTF Compatibility Feature

This appendix describes a PTF compatibility feature that enables you to process PTFs that were created using SMP syntax from previous releases. These PTFs include the initial PTF that defines the function and subsequent PTFs that service that function.

Eligible PTFs

The PTFs that can be processed with this feature are restricted to program products that are independent of the system control program. PTFs that are SU definitions and service PTFs that are applicable to the base system control program and the SUs which modify that system control program are not eligible.

PTFs that do not define the program product initial installation package may not contain the NPRE operand in the ++VER modification control statements. If more than one ++VER modification control statement is determined to be applicable during APPLY or ACCEPT processing, the PTF is terminated.

SMP Environment

You should define the SMP environment for processing with the PTF compatiblity feature separately from the system control program and other program products. The SMP primary data sets should be allocated exclusively for the processing of the program product. The ACDS and CDS SYSTEM entries are initialized with the SREL subentry value present in the SREL operand of the ++VER modification control statement from the PTF defining the program product. If a separate PTS is to be used, the SREL subentry value is also placed in the PTS SYSTEM entry. The ACRQ, CRQ, and SCDS data sets can be null, but must be defined.

The FMID Execution Parameter

Prior to receiving the PTF that defines the program product, the SYSMOD-ID from the ++PTF modification control statement must be specified as the value following 'FMID=' in the PARM operand of the EXEC statement in the JCL statements used to invoke SMP. All subsequent executions of SMP that process PTFs or invoke the UCLIN function must also have the FMID parameter specified in the EXEC statement.

SMP Function Variations

The SMP functions RECEIVE, APPLY, ACCEPT, and UCLIN are sensitive to the presence of the FMID parameter coded on the EXEC statement. The processing variations for each of these functions is described below:

- RECEIVE

  All PTFs are received have at least one ++VER modification control statement whose SREL operand list contains a value found in the SREL subentries of the PTS SYSTEM entry. The absence of the FMID operand on ++VER modification control statements does not result in a syntax error as is normally the case. When the PTF whose SYSMOD-ID matches the FMID parameter value is received, the SYSMOD entry created has the FUNCTION indicator set so that subsequent APPLY and ACCEPT processing treats the PTF as a function type SYSMOD. The MCS entries of all PTFs received are unchanged.

- APPLY

  PTFs processed that do not contain FMID operands on their applicable ++VER modification control statements have the FMID value in the EXEC statement logically appended to them with the exception of the PTF that is treated as a function. This is the only variation in APPLY processing. The function PTF must be applied prior to or concurrent with the service PTFs.

- ACCEPT

  The processing is the same as for APPLY.

- UCLIN

  All SYSMOD and element entries that are created, updated, or replaced have the FMID parameter value from the EXEC statement placed in the FMID subentry unless one already exists or is specified on the UCL statement. If you add or replace an entry and specify the FMID operand on the UCL statement, you must ensure that it matches that specified in the EXEC statement.

Glossary

This glossary defines SMP terms used in this publication. Additional terms can be found by referring to the index of the publication, to prerequisite publications, and to the IBM Data Processing Glossary, GC20-1699.

*IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standards Vocabulary for Information Processing (Copyright 1970 by American National Standards Institute, Inc.), which was prepared by Subcommittee X3K5 on Terminology and Glossary of American National Standards Committee X3.

A

accept

In SMP, the process initiated by the ACCEPT control statement that places SYSMODs into the distribution libraries or permanent user libraries.

ACCID

The identifier of the ACDS data set, maintained as a subentry in the PTS SYSMOD entry to identify the ACDS data set on which the SYSMOD is accepted. See CDSID.

ACDS

The Alternate Control Data Set (SMPACDS) describes the SYSMODs and elements in the distribution libraries.

ACRQ

The Alternate Conditional Requisite Queue Data Set (SMPACRQ) holds the parsed ++IF modification control statements for ACCEPT processing of conditional requisite data.

APAR

Authorized program analysis report.

APAR fix

An APAR fix is a temporary correction mechanism because the correction is usually replaced at a later date by a permanent correction (PTF). In SMP, APAR fixes are identified by the ++APAR modification control statement. APARs can be fixed in PTFs and functions as denoted by the SUP operand.

APPID

The identifier of the CDS data set, maintained as a subentry in the PTS SYSMOD entry to identify the CDS data set on which the SYSMOD is applied. See CDSID.

apply

In SMP, the process initiated by the APPLY control statement that places SYSMODs into the target system libraries.

authorized program analysis report

The report of a defect in an IBM System Control Program (SCP) or Program Product (PP). The correction that results is known as an APAR fix.

B

base level system

The level of the target system modules, macros, source modules and DLIBs created by system generation (SYSGEN) to which function and service are applicable. OS/VS2 MVS Release 3.8 and OS/VS1 Release 6.7 are two examples of what would be considered base level systems.

base level function SYSMODs

SYSMODs that define elements of the base system or program products that were not previously present in the target system. They are identified to SMP using the ++FUNCTION modification control statement. Base level function SYSMODs do not have an FMID keyword in the ++VER modification control statement.

bypass

In SMP, to circumvent error conditions that would normally result in termination of SYSMOD processing using the BYPASS keyword on the ACCEPT, APPLY, RECEIVE or RESTORE control statements.

C

CDS

The Control Data Set (SMPCDS) contains information about the target system macros, modules, assemblies, load modules, source modules, libraries copied from DLIBs during SYSGEN, and the SYSMODs applied to the target system.

CDSID

A one to eight character system identifier of the CDS or ACDS data set contained in the CDS or ACDS SYSTEM entry. The identifier is placed in the SYSMOD entry on the PTS as an APPID subentry when the SYSMOD is applied, and as an ACCID subentry when the SYSMOD is accepted.

CNTL

The Control Statement Input Data Set (SMPCNTL) contains the SMP control statements.

collapse

See element version collapse.

conditional actions

Actions described by the ++IF modification control statements in terms of SYSMODs required to be applied to the target system libraries, or accepted into the distribution libraries when a specified function SYSMOD is present. The condition is described by the FMID operand; the actions are described by the REQ operand.

conditional requisite data

Data supplied in the ++IF modification control statements. This data is used to determine service requirements that are environment dependent.

CRQ

The Conditional Requisite Queue Data Set (SMPCRQ) holds ++IF modification control statements for APPLY processing of conditional requisite data.

D

dependent level SYSMODs

Function SYSMODs that introduce new elements or redefine elements of the base level system or program products. Dependent level SYSMODs cannot exist without a base level function; therefore, they must have an FMID keyword in the ++VER modification control statement, which specifies a prerequisite function SYSMOD.

deleted SYSMOD

A function SYSMOD specified as the value of a DELETE operand by the deleting SYSMOD.

deleting SYSMOD

The function SYSMOD that specifies other function SYSMODs as values of the DELETE operand.

distribution libraries

IBM-supplied partitioned data sets containing elements for subsequent inclusion in a new system. These data sets are updated by ACCEPT processing.

DLIB

Distribution library

E

element

In SMP, a module, macro or source module, identified to SMP by the element modification control statements.

element modification control statement

Consist of the

Consist of ++MAC, ++MACUPD, ++MOD, ++SRC, ++SRCUPD, ++UPDTE or ++ZAP modification control statements. They are used by SMP to identify the type of element and whether it is an update or a replacement.

element selection

The process of choosing the appropriate modification(s) to an element from the SYSMODs selected for APPLY or ACCEPT processing that have elements in common.

element version collapse

To transfer ownership of an element from one function to another, even though the elements may already be present in the function to which the elements are transferred. See VERSION.

environment

In SMP, the set of function SYSMODs successfully applied to the target system or successfully accepted into the distribution libraries.

ERROR indicator

In SMP, an indicator in a SYSMOD entry on the CDS or ACDS set prior to any SMP updating of libraries. The ERROR indicator is reset if updating completes successfully. If updating does not complete successfully, the ERROR indicator remains set in the SYSMOD entry to indicate that processing of that SYSMOD failed.

EXCLUDE

The keyword used to specify a SYSMOD not to be included in SMP processing.

F

feature level SYSMOD

See dependent level SYSMOD.

FMID

Function modification identifier.

function

In SMP, system components and program products that can be optionally installed in a user's system. Functions are identified to SMP by the ++FUNCTION modification control statement.

function SYSMOD

A SYSMOD identified by the ++FUNCTION modification control statement.

function modification identifier

An identifier in the form of a SYSMOD-ID that identifies the function to which the elements belong. It is associated with all elements installed on the user's system as part of a function system modification. It becomes the FMID subentry of the MOD, MAC, SRC, and SYSMOD entries.

functional version of an element

The functional version of an element is identified by the FMID of the SYSMOD which contains the particular element. For function SYSMODs, the FMID is the SYSMOD-ID itself. For service SYSMODs, the FMID is found in the ++VER modification control statements. When a function SYSMOD is applied to the target system libraries or accepted into the distribution libraries, the FMID from the selected SYSMOD is placed into the associated entries on the CDS or ACDS.

G

GENASM

Subentries of MAC entries that are names of ASSEM or SRC entries to be assembled when the macro is modified.

H

header modification control statement

Header modification control statements are used by SMP to identify the type of modification. They consist of the ++APAR, ++FUNCTION, ++PTF and ++USERMOD modification control statements.

hierarchy

In SMP, used to describe the top-down structure of function and service SYSMODs, where each SYSMOD is dependent on the one above it.

I

IMASPZAP

The IBM service aid used to apply superzaps. In VS1, IMASPZAP may also be invoked under the name HMASPZAP, and in VS2 under the names HMASPZAP or AMASPZAP. SMP invokes this service aid under the name IMASPZAP.

inline JCLIN

The JCL statements supplied with the ++JCLIN modification control statement in a SYSMOD. They are used to update the CDS when a SYSMOD is processed by APPLY processing.

install

In SMP, to apply a SYSMOD into the target system libraries or to accept a SYSMOD into the distribution libraries.

J

JCLIN

This term is used to describe:

- The process of creating or updating the CDS using JCL input data,
- The data set that contains the Stage I output from system generation used to update or create the CDS,
- The JCLIN control statement used to read in the JCLIN data set,

- The ++JCLIN modification control statement, packaged as part of a SYSMOD to enable SMP to perform the CDS updates during APPLY processing. See inline JCLIN.

L

load module

The output of the linkage editor; a program in a format suitable for loading into main storage for execution.

LMOD

In SMP, an abbreviation for load module. For example, an entry on the CDS that represents a load module is an LMOD entry.

LOG

The History Log Data Set (SMPLOG) contains time-and-date stamped records of all significant events that occur during modification processing, and user messages supplied using the LOG control statement.

logical deletion

Data set entries that are treated as if they do not exist but are not physically deleted.

M

MAC

In SMP, an abbreviation for macro. The element modification control statement that identifies a macro replacement is ++MAC; macro updates are identified by the ++MACUPD (or ++UPDTE) modification control statement. An entry on the CDS that represents a macro is a MAC entry.

*macro

An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language.

mass

In SMP, to process every eligible SYSMOD.

merge

In SMP, to combine source or macro updates into a temporary work data set, based on the service order relationship and type of SYSMOD.

MOD

In SMP, an abbreviation for module. The element modification control statement that identifies a module replacement is ++MOD. An entry on the CDS that represents a module is a MOD entry. module is a MOD entry.

MODID

Modification identifier.

modification

In SMP, an alteration or correction to an IBM SCP, PP or user program, also known as a system modification (SYSMOD).

modification identifier

A list of system modification identifiers consisting of the last system modification to totally replace the element and any subsequent partial updates to the element (that is, ZAPs on module elements) plus the function that owns the element. These entities are referred to as the FMID, UMID and RMID. MODIDs are part of the element entries on the CDS and ACDS.

modification text

The statements associated with the element modification control statements, such as macro definition statements, source code, and object code.

*module

A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine.

MTS

The Macro Temporary Store Data Set (SMPMTS) contains macro modifications not intended to be placed into a target system library.

N

negative prerequisite

In SMP, a SYSMOD (or SYSMODs) that must not be present in the system in order for the SYSMOD currently being processed to be successfull installed.

NPRE

The NPRE operand on the ++VER modification control statement.

null CDS

An allocated but uninitialized CDS; that is, no entries have been made on it.

O

*object module

A module that is the output of an assembler or compiler and is input to a linkage editor.

P

package

In SMP, a package consists of all of the input that comprises one system modification, including the modification control statements, modification text, relative files, or data sets containing modification text, such as TXLIB.

parse

In SMP, to examine, syntax check, and resolve a statement into component parts.

PEMAX

The maximum number of SYSMOD elements that can exist in a SYSMOD (MAC, MOD, SRC, SRCUPD, UPDTE, MACUPD, or ZAP), plus the related SYSMODs listed in the CDS or ACDS SYSMOD entry (SYSMODs listed in the PRE, SUP, REQ or merge group fields). PEMAX is used to determine the size of SMP work areas.

PP

Program product.

PRE

The PRE operand on the ++VER modification control statement.

prerequisite

In SMP, a SYSMOD (or SYSMODs) that must either be in the system or be in the process of installation on the system for the SYSMOD currently being processed to be successfully installed.

primary data set

In SMP, the SMP data sets that must be allocated after system generation.

program product

A licensed program that performs a function for the user and usually interacts with and relies upon the SCP or some other IBM provided control program. IMS and CICS are program products.

program temporary fix

A correction to a defect in an IBM System Control Program (SCP) or Program Product (PP). In the absence of a new release of a system or component that incorporates the correction, the fix is not temporary but is the permanent and official correction mechanism. New elements might also be defined in a PTF.

PTF

Program temporary fix.

PTFIN

The System Modification Input Data Set (SMPPTFIN) contains the SYSMODs to be processed by RECEIVE.

PTF tape

In SMP, the IBM-supplied tape that contains the SYSMODs.

PTS

The PTF Temporary Store Data Set (SMPPTS) is used as a temporary storage for SYSMODs that are received using the RECEIVE control statement.

purge

In SMP, to delete any SYSMOD that is successfully processed by APPLY and ACCEPT from the PTS. This process is controlled by setting the PURGE indicator in the SYSTEM entry of the PTS.

R

receive

In SMP, the process initiated by the RECEIVE control statement that reads the SYSMODs from the PTFIN Data Set and stores them on the PTS for subsequent SMP processing.

regressed SYSMOD

A SYSMOD that has one or more of its elements modified by subsequent SYSMODs that are not related to it.

regressing SYSMOD

The SYSMOD that causes regression of previous modifications when it is installed.

regression

In SMP, when a modification is made to an element by a SYSMOD that is not related to SYSMODs that previously modified the element.

reject

In SMP, the process initiated by the REJECT control statement that deletes SYSMODs processed by RECEIVE from the target system and the PTS. The REJECT process is also initiated by setting the REJECT indicator in the PTS SYSTEM entry; that is, any SYSMOD that is successfully processed by RESTORE is deleted from the PTS.

related SYSMOD

Associations between SYSMODs established by the FMID, PRE, REQ, or SUP keywords.

relative files

Files that contain modification text and JCL input data associated with a SYSMOD.

replacement modification identifier

The modification identifier of the last SYSMOD to completely replace a given module, macro, or source module. It is known as the RMID subentry of the MOD, MAC, and SRC entries.

REQ

The REQ operand on the ++VER modification control statement.

requisite

A SYSMOD (or SYSMODs) specified in either the PRE or REQ keywords on the ++VER modification control statement or in the REQ keyword on the SYSMOD's associated ++IF modification control statement. It defines a SYSMOD (or SYSMODs) that must be processed concurrently or prior to the SYSMOD being processed.

requisite SYSMOD set

The set of PTFs necessary to fix a set of APARs across a number of environments.

restore

In SMP, the process initiated by the RESTORE control statement that removes SYSMODs processed by APPLY from the target system libraries, the CDS, and optionally, the PTS.

restore group

Consists of all the SYSMODs that have a direct or indirect relationship with a SYSMOD being restored using the GROUP operand.

RMID

Replacement modification identifier.

S

SCDS

The Save Control Data Set (SMPSCDS) contains back-up copies of CDS entries that are modified during APPLY processing by inline JCLIN.

SCP

System control program.

secondary data sets

In SMP, the data sets that are allocated using JCL during the SMP job.

select

In SMP, the process of selecting a specific SYSMOD.

SELECT

The keyword that is used to specify the SYSMOD (or SYSMODs) to be included in SMP processing.

selectable unit

A functional enhancement to an IBM SCP (OS/VS1 Release 6.0 and OS/VS2 Release 3.7).

service order relationship

A relationship among service SYSMODs determined by the PRE and SUP operands, and the type of SYSMOD.

service level of an element

A set of FMID, RMID, and UMID subentry values.

service SYSMOD

Any SYSMOD identified by the ++APAR, ++PTF or ++USERMOD modification control statements.

service update process

The method for integrating PTFs into function SYSMOD packages.

SMP

System Modification Program

SMP control statements

Define the SMP processes to be performed, such as RECEIVE.

SMP modification control statements

Statements that define the type of system modification, such as ++MAC for a macro replacement. They also identify the elements to be added to, modified in, or deleted from the system libraries and distribution libraries. In addition, there are modification control statements that describe the environment and conditions that must be met in order for SMP to install the modification.

source

See source module.

source module

The source statements that constitute the input to a language translator for a particular translation.

source update

In SMP, a SYSMOD that updates a source module.

SRC

In SMP, an abbreviation for source. An entry in the CDS that represents a source module is a SRC entry. An element modification control statement that replaces a source module is ++SRC; that updates a source module is ++SRCUPD.

SRCUPD

A source module update.

SREL-ID

System release identifier

STS

The Source Temporary Store Data Set (SMPSTS) contains source code modifications that are not intended to be placed into a target system library.

SU

Selectable unit.

subentry

A field within an entry.

SUP

Supersede.

supersede

In SMP, a SYSMOD (or SYSMODs) contained in or replaced by the SYSMOD or requisite set of SYSMODs currently being processed. A superseded SYSMOD is inferior to the SYSMOD that superseded it.

superzap

A generic term for the process performed by IMASPZAP.

SYSMOD

System modification.

SYSMOD-ID

System modification identifier

SYSMOD selection

The process of determining which SYSMODs are eligible to be processed.

system modification

The input data to SMP that defines the introduction, replacement or update of elements in the operating system and associated distribution libraries, installed under the control of SMP. A system modification is defined by a set of modification control statements. It must include one header modification control statement and at least one ++VER modification control statement. It may also include ++IF modification control statements, one ++JCLIN modification control statement, and includes element modification control statements.

system modification identifier

The name that SMP associates with a system modification. It is specified as the value of the ++APAR, ++FUNCTION, ++PTF or ++USERMOD operand. A SYSMOD-ID can be any alphanumeric string of seven (7) characters, the first of which must be alphabetic. IBM reserves the characters "A" thru "K" and "U" thru "Z" for the first character of IBM

SYSMOD-IDs.

system release identifier

A four-byte value representing the system release level, such as Z038 for OS/VS2 MVS Release 3.8.

T

target system

The destination system for APPLY and RESTORE processing.

TLIB

A DD statement (SMPTLIB) pointing to a volume or set of volumes used as temporary storage for libraries loaded during RECEIVE processing when SYSMODs are packaged using the relative file technique.

U

UMID

Update modification identifier.

update

In SMP, the process of modifying, without replacement, existing modules, macros, or source modules.

update modification identifier

The modification identifier of the SYSMOD that updated the last replacement of a given module, macro or source module.

USERMOD

User modification.

user modification

A modification to IBM-supplied code that is prepared by the user and identified to SMP using the ++USERMOD modification control statement. User modifications can also define elements created by the user to interface with IBM software.

V

VERSION

used to specify one or more SYSMODs that contain elements that are functionally inferior to elements contained in the SYSMOD that specifies the VERSION operand. The VERSION operand is also used to change ownership of elements.

Index

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

OS/VS System Modification Program
(SMP) System Programmer's Guide
GC28-0673-5

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. 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.

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 comments apply to a Selectable Unit, please provide the name of the Selectable Unit ________.

If you wish a reply, give your name and mailing address:

____

____

____

Please circle the description that most closely describes your occupation.

	(Q)	(U)	(X)	(Y)	(Z)	(F)	(I)	(L)
Customer	Install Mgr.	System Consult.	System Analyst	System Prog.	Applica. Prog.	System Oper.	I/O Oper.	Term. Oper.

	(O)
	Other

	(S)	(P)	(A)	(B)	(C)	(D)	(R)	(G)	(J)	(E)	(N)	(T)
IBM	System Eng.	Prog. Sys. Rep.	System Analyst	System Prog.	Applica. Prog.	Dev. Prog.	Comp. Prog.	System Oper.	I/O Oper.	Ed. Dev. Rep.	Cust. Eng.	Tech. Staff Rep.

Number of latest Newsletter associated with this publication: ________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC28-0673-5

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 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tape

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along Line

Technical Newsletter

This Newsletter No.	GN28-2992
Date	July 1, 1979
Base Publication No.	GC28-0673-5
File No.	S370-37
Prerequisite Newsletters/ Supplements	None

OS/VS System Modification Program (SMP) System Programmer's Guide

This newsletter contains replacement pages for *System Modification Program (SMP) System Programmer's Guide.*

Before inserting any of the attached pages into *System Modification Program (SMP) System Programmer's Guide*, read ***carefully*** the instructions on this cover. They indicate when and how you should insert the pages.

Pages to be Removed	Attached Pages to be Inserted*
Cover - Edition Notice	Cover - Edition Notice
iii - iv	iii - iv
xxi - xxiv	xxi - xxiv
xxvii - xxviii	xxvii - xxviii
43 - 46	43 - 46
49 - 50	49 - 50
55 - 56	55 - 56
77 - 78	77 - 78
81 - 82	81 - 82
113 - 118	113 - 118
127 - 128	127 - 128
137 - 140	137 - 140
155 - 160	155 - 160
165 - 166	165 - 166
169 - 176	169 - 176.2
181 - 186	181 - 186
247 - 248	247 - 248.2
253 - 256	253 - 256.2
271 - 282	271 - 282
289 - 290	289 - 290
293 - 302	293 - 302.2
305 - 306	305 - 306.14
343 - 344	343 - 344
357 - 358	357 - 358
367 - 368	367 - 368
371 - 372	371 - 372
387 - 388	387 - 388

IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, New York 12602

Printed in U.S.A.

Pages to be Removed	Attached Pages to be Inserted*
391 - 392	391 - 392
413 - 416	413 - 416.2
425 - 430	425 - 430
433 - 440	433 - 440
445 - 446	445 - 446
None	452.1 - 452.6
479 - 480	479 - 480
483 - 502	483 - 502

*If you are inserting pages from different Newsletters/Supplements and ***identical*** page numbers are involved, always use the page with the latest date (shown in the slug at the top of the page). The page with the latest date contains the most complete information.

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

This update reflects support for two new functions–RETRY and UNLOAD–and also contains miscellaneous technical changes.

Note: *Please file this cover at the back of the base publication to provide a record of changes.*

OS/VS System Modification Program
(SMP) System Programmer's Guide
GC28-0673-5

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. 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.

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 comments apply to a Selectable Unit, please provide the name of the Selectable Unit ________.

If you wish a reply, give your name and mailing address:

____

____

____

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

Please circle the description that most closely describes your occupation.

	(Q)	(U)	(X)	(Y)	(Z)	(F)	(I)	(L)	(O)
Customer	Install Mgr.	System Consult.	System Analyst	System Prog.	Applica. Prog.	System Oper.	I/O Oper.	Term. Oper.	Other

	(S)	(P)	(A)	(B)	(C)	(D)	(R)	(G)	(J)	(E)	(N)	(T)
IBM	System Eng.	Prog. Sys. Rep.	System Analyst	System Prog.	Applica. Prog.	Dev. Prog.	Comp. Prog.	System Oper.	I/O Oper.	Ed. Dev. Rep.	Cust. Eng.	Tech. Staff Rep.

Number of latest Newsletter associated with this publication: ________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC28-0673-5

OS/VS System Modification Program (SMP) System Programmer's Guide (S370-37) Printed in U.S.A. GC28-0673-5

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601