GC28-0673-6
File No. S370-37

Systems

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

IBM

Seventh Edition (September, 1980)

This is a major revision of GC28-0673-5 and Technical Newsletter GN28-2992. 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 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 subsequent 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.

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 04.20.

Note that this publication is a major revision of and obsoletes GC28-0673-5, _OS/VS System Modification Program (SMP) System Programmer's Guide_ for SMP _Release 4 users._

The publication contains nine chapters, five appendices and a glossary as follows:

Chapter 1: "Introduction" - provides an overview of SMP.

Chapter 2: "SMP Processing" - provides a detailed explanation of the processing that takes place during RECEIVE, REJECT, APPLY, RESTORE, ACCEPT, JCLIN and UCLIN processing.

Chapter 3: "Installation and Use" - provides the information necessary to install and execute SMP. Examples of commonly used SMP procedures are included.

Chapter 4: "Reports" - explains the reports that might be produced during SMP processing.

Chapter 5: "Control Statements" - provides detailed descriptions of the SMP control statements, in alphabetic order by control statement.

Chapter 6: "Modification Control Statements" - provides detailed descriptions of the modification control statements, in alphabetic order by modification control statement.

Chapter 7: "Data Sets" - describes the data sets used by SMP.

Chapter 8: "Control Dataset Entries" - For user reference in reading processing descriptions and SMP listings.

Chapter 9: "SYSMOD Construction" - Contains construction rules and techniques for building SYSMODs.

Appendix A: "Rules for Coding SMP Statements" - provides syntax rules for SMP Control and Modification Control statements.

Appendix B: "Syntax Notation Conventions" - provides the syntax notation conventions used to define SMP Control 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.

Appendix D: "SMP3/SMP4 Amendments" - describes the differences between SMP Release 3 and SMP Release 4. This information was formally the "Summary of Amendments" section of the <u>SMP System Programmer's Guide</u> (GC28-0673-5).

Appendix E: "UCL Operand/Dataset Cross Reference" - provides charts which illustrate the allowable UCLIN operations to the entries in each SMP data set.

Glossary: provides definitions of SMP terms and abbreviations.

Associated Publications

- OS/VS System Modification Program (SMP) Messages and Codes, GC38-1047
- 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, OC26-3813
- 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, 5Y26-3837
- 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

SUMMARY OF AMENDMENTS

This publication represents a major re-organization of the SMP System Programmer's Guide GC28-0673-5. The material from the former publication has been re-arranged to distinguish between two different types of SMP "users": (1) the "user" who creates system modifications and (2) the "user" who manages the installation of system software and uses SMP to install these modifications.

For the first "user", a new chapter has been added, "SYSMOD Construction", which consolidates much of the construction information which was formerly spread throughout the former Programmer's Guide.

For the second "user", the "Installation and Use" chapter has been expanded to include considerations for and examples of SMP use in the management of system software.

For both sorts of "users", the "SMP Processing" chapter has been rewritten to clarify the processing descriptions in response to APARs and Reader's Comments.

- The APPLY Processing section has been completely revised with emphasis on element selection and element processing (assemblies, link edits, etc.)

- The JCLIN Processing section has been completely revised to more clearly define the affects of JCLIN and the conventions which SMP requires.

This publication is accompanied by a new publication, SMP Messages and Codes (GC38-1047), which contains SMP Return Codes, Diagnostic Techniques and Messages from the former publication. This new publication is meant to be a reference for the user who installs system software and must deal with such items.

Chapter 5, "Control Statements", and chapter 6, "Modification Control Statements", are meant to be used for syntax reference and remain essentially unchanged.

- The UCL Statement section has been re-structured such that each statement is described for the particular data set and entry to which it is applicable. A set of charts has been included (Appendix E) to illustrate the possible UCL updates allowed for each data set and entry.

- The LIST Statement section has been re-organized in a manner similar to the UCL Statement section to clarify the list options for each SMP data set.

A new chapter, "SMP Data Set Entries", is included for reference in the use of UCLIN and the analysis of SMP listings. This chapter describes the structure of the SMP data sets and describes the data contained in each of the entries.

The material formerly included in the "Summary of Amendments" describing the differences between SMP Release 3 and SMP Release 4 has been included as Appendix D in this publication.

INTRODUCTION

The System Modification Program (SMP) is a service aid used (1) to define system modification and (2) to install the modifications on an OS/VS1 or OS/VS2 operating system.

Development system programmers use SMP to define modifications that add to or replace functions in the system and that correct problems in the software.

Installation system programmers use SMP to install IBM-provided and user-developed modifications. SMP provides facilities to manage an installation's software inventory by providing extensive records of additions and modifications in the SMP control data sets.

The selective system tailoring offered by program products and the individual tailoring possible by user modifications mean that each installation's operating system is potentially unique. As a result, SMP was designed to meet the following objectives.

- To ensure that only modifications that are applicable to a system are applied to that system -- for example, to ensure that prerequisites for a program product have been installed on the system.

- To prevent an unintentional replacement of a modified module with another module that does not contain the earlier modifications (referred to as "down-leveling")-- for example, to warn the installation when a PTF updates a module that includes a user modification.

The responsibility for meeting these objectives, however, is shared by SMP and the developer of a modification (whether an IBM developer or, in the case of a user modification, a user developer).

- The developer of a modification defines the relationship of the modification to the system and to other modifications.

- The installer creates and initializes SMP control data sets on which SMP maintains records of the modules and macros that comprise the system, the modifications that have been made to the system, and the relationship of modifications to each other and to the modules and macros in the system.

This introduction describes, at a conceptual level, how SMP meets its objectives, including a description of the information the developer of a SYSMOD must provide and the data sets the installer of a SYSMOD must create. Although the tasks of the developer and installer are distinct, they should both understand the information in this introduction to use SMP effectively.

FUNCTIONAL HIERARCHY OF THE OPERATING SYSTEM

To enable developers to identify dependencies between system modifications, IBM has defined a functional hierarchy in the operating system that consists of base-level functions upon which dependent-level functions are built. Non-function modifications can provide service or user updates for each function (base or dependent-level) within the hierarchy. Figure 1 illustrates a possible hierarchy within a base-level function.

IBM defines the base-level functions by logically grouping the elements, of the system (object modules, source modules, and macros) into functions such as ISO or VTAM. An element of one base-level function should not exist in (that is, intersect) another base-level function. A base-level function has no prerequisites for SMP to install it. For example, a base-level function might be a program product that introduces a new group of elements to the system; or a new version of a program product that does not require the installation of an earlier version. Program products that modify a component in the base control program are not base-level functions unless they include all the function in the base component and, therefore, do not require the prior installation of the base component.

Dependent-level functions replace or add to the elements in a base-level function. An element can exist in more then one dependent-level function within a base-level function. Note that in Figure 1 both dependent level functions FXY1020 and FXY1030 include module B. A dependent-level function. requires the prior installation of the base-level function that it modifies.

Non-function modifications like APARs, PTFs, and USERMODs update elements in a base or dependent-level function. A single non-function modification must update only one function. SMP will not allow construction of a PTF that modifies both dependent-level functions FXY1020 and FXY1030 in Figure 1. If a change requires modifications to modules A and B, the set of three PTFs illustrated must be defined since the base-level and two dependent-level functions are involved.

The developer of a modification is responsible for identifying the modifications placement within the hierarchy. He must define:

- Whether a function modification is a new base-level or dependent-level function; and, for dependent-level functions, to which base-level function it belongs.

- To which function an APAR, PTF or USERMOD applies.

- The relationship to other SYSMODs on the same level.

The hierarchy, as illustrated, implies no relationship between SYSMODs on the same level, for example, between function SYSMODs FXY1020 and FXY1030 in Figure 1. The hierarchy shows all the dependent-level functions available for a base-level function, not necessarily all the modification that can be applied to a single system. Dependent-level functions might be mutually exclusive or one might be a prerequisite of another. If the developer indicates that a SYSMOD can coexist on one system with other SYSMODs on the same level and those SYSMODs contain common (intersecting) elements, he must also indicate the relationship between the common elements. For example, in Figure 1 both dependent-level functions FXY1020 and FXY1030 can be applied concurrently to a system; the developer must indicate which version of module B is superior (this topic is further discussed in "Defining the

Environment to which a SYSMOD Applies" later in this chapter).

```
                              ┌-----------┐              ┌-----------┐
                              |           |              |           |
                              | FUNCTION  |<-------------|    PTF    |
                              | GXY1000   |              |  UZ00001  |
                              |           |              |           |
                              └-----------┘              └-----------┘
                                    |
                                    |
                         ┌----------------------┐
                         |                      |
                         |                      |
                         |                      |
   ┌----------┐    ┌-----------┐          ┌-----------┐      ┌-----------┐
   |          |    |           |          |           |      |           |
   |   PTF    |--->| FUNCTION  |          | FUNCTION  |<-----|   PTF     |
   | UZ00002  |    | FXY1020   |          | FXY1030   |      | UZ00003   |
   |          |    |           |          |           |      |           |
   └----------┘    └-----------┘          └-----------┘      └ ----------┘

        ++FUNCTION(GXY1000) /* BASE LEVEL FUNCTION */ .
        ++VER(Z038) .
        ++MOD(A) .
        ++MOD(C) .

        ++FUNCTION(FXY1020) /* DEPENDENT-LEVEL FOR GXY1000 */ .
        ++VER(Z038) FMID(GXY1000) .
        ++MOD(B) .

        ++FUNCTION(FXY1030) /* DEPENDENT-LEVEL FOR GXY1000 */ .
        ++VER(Z038) FMID(GXY1000) VERSION(FXY1020) .
        ++MOD(B) .
        ++MOD(C) .

        ++PTF(UZ00001) /* SERVICE FOR GXY1000 */ .
        ++VER(Z038) FMID(GXY1000) .
        ++IF FMID(FXY1020) THEN REQ(UZ00002) .
        ++IF FMID(FXY1030) THEN REQ(UZ00003) .
        ++MOD(A) .

        ++PTF(UZ00002) /* SERVICE FOR FXY1020 */ .
        ++VER(Z038) FMID(FXY1020) REQ(UZ00001) .
        ++IF FMID(FXY1030) THEN REQ(UZ00003) .
        ++MOD(B) .

        ++PTF(UZ00003) /* SERVICE FOR FXY1030 */ .
        ++VER(Z038) FMID(FXY1030) REQ(UZ00001) .
        ++IF FMID(FXY1020) THEN REQ(UZ00002) .
        ++MOD(B) .
```

Figure 1 - Hierarchy of Function and Service

SYSMOD DEFINITION

A SYSMOD is defined to SMP by Modification Control statements. Modification Control statements are identified as those prefaced by the characters "++" in columns 1 and 2.

MODIFICATION CONTROL STATEMENTS (MCS)

The modification control statements can be grouped into four categories, reflecting the four types of information the developer of a SYSMOD provides.

1. Header modification control statements, by which the developer identifies the type of modification.

   - ++FUNCTION to define a base-level or dependent-level function.

   - ++PTF for PTFs that service elements of IBM software.

   - ++APAR for tempory corrections to elements of IBM software.

   - ++USERMOD for user modifications that change, replace, or interface with elements of IBM software.

   The developer also assigns the modification a unique seven character identifier, called a SYSMOD-ID, on the header modification control statement. SMP uses the SYSMOD-ID to track changes to elements introduced by this SYSMOD, and to record this SYSMOD's dependencies on other SYSMODs.

2. Environment definition statements, the ++VER and ++IF statements on which the developer identifies the system(s) and release(s) to which the modification applies and this modification's relationship to other modifications.

3. Element definition statements defining the elements affected by this modification:

   - For macros, ++MAC for a macro replacement; ++MACUPD (or for compatibility with prior versions of SMP, ++UPDTE) for a macro update.

   - For object module replacements, ++MOD. For object module updates, ++ZAP.

   - For source modules, ++SRC for a source module replacement; ++SRCUPD for a source module update.

   The developer must code an element definition statement for each element (each object module, source module and macro) in the SYSMOD.

4. Installation data provided by the developer as jobstream of linkedit, copy and assembly jobsteps (++JCLIN) to describe the structure of the elements in the operating system libraries. The JCL input data is similar to the JCL that stage I of system generation (SYSGEN) produces and that stage II uses to structure the elements from the DLIBs (distribution libraries) onto the system libraries. The ++JCLIN statement and JCL input data are necessary only if the

SYSMOD affects the structure of the elements on a system library.

DEFINING THE ENVIRONMENT TO WHICH A SYSMOD APPLIES

The ++VER and ++IF statements define the environment to which a SYSMOD applies. The ++VER (verify) statement identifies <u>absolute</u> requirements that must be met before SMP installs a SYSMOD; the ++IF statement identifies additional requirements <u>conditional</u> on the presence of other function SYSMODs in the system. Every SYSMOD must contain the ++VER statement to identify, at least, the system and release level to which a SYSMOD applies; other requirements specified on ++VER and the presence of the ++IF statement depend on the content of the SYSMOD and the possible existence of related SYSMODs in the system.

Referring again to Figure 1, the ++VER statements define the environment to which each SYSMOD applies:

- Function GXY1000 is a base-level function and specifies only the system release environment, "Z038".

- Functions FXY1020 and FXY1030 are dependent-level functions which specify their dependence upon the base-level function, GXY1000, using the ++VER FMID keyword.

- Module "B" exists in both dependent-level functions. Function FXY1030, which we will assume is supplying a functionally superior version of the module, expresses this "superiority" relationship with FXY1020 using the VERSION keyword on its verify statement.

- The set of service SYSMODs (PTFs) illustrate the construction required for a "fix" which involves elements belonging to more than one FMID.

  - The first PTF, UZ00001, applies to the base-level function and supplies only module "A".

    The ++IF conditional requisite statements supply SMP with the data required to determine other required PTFs if either of the dependent functions are installed.

  - The second PTF, UZ00002, supplies module "B" applicable to a system which has function FXY1020 installed (the ++VER FMID keyword indicates this).

    This PTF specifies the base function's PTF, UZ00001, as a REQ to ensure UZ00001's installation.

    If the environment in which this PTF is being installed has function <u>FXY1030</u> installed, the version of module "B" supplied in this PTF is not appropriate (recall that the installation of FXY1030 indicated that its module "B" was functionally superior to the module "B" in FXY1020). Although SMP will install this PTF, module "B" will be <u>excluded</u> (that is, not applied or accepted). The ++IF conditional requisite data supplied by this PTF supplies SMP the data specifying the required PTF (with the appropriate module "B") for the environment in which function FXY1030 is installed.

- The third PTF, UZ00003, supplies module "B" applicable to a system which has function FXY1030 installed.

  This PTF specifies the base function's PTF, UZ00001, as a REQ to ensure UZ00001's installation.

  Further, the ++IF conditional requisite statement ensures the installation of PTF UZ00002 if FXY1020 is installed. Because of the relationship between FXY1030 and FXY1020 in terms of the versions of the module "B" involved in this illustration, UZ00002's module "B" will not be processed; the ++IF statement is necessary to ensure that the UZ00002 service is recorded for the FXY1020 environment since subsequent FXY1020 service may need to specify it as a requisite.

SMP PROCESSING

SMP processing is invoked by SMP Control statements.

STAGES IN SMP PROCESSING

SMP processing is divided into several stages to allow the installation to do the following:

1. The first logical process, RECEIVE, introduces SYSMODs to SMP. SMP receive processing stores the modifications that the installation selects on the SMPPTS data set (referred to as the PTS) for subsequent processing. An installation can receive all modifications that it intends to install, regardless of the schedule on which it plans to install them. For example, the installation can receive PTF SYSMODs for functions that it has not yet installed.

   An installation can remove modifications from the PTS by using the REJECT, process.

3. An installation can install modifications to its set of operating system libraries using the APPLY process. APPLY processing involves four primary SMP data sets:

   • The SMPCDS contains information concerning the functions and modifications that comprise the operating system. SMP uses the information in the CDS to determine if a SYSMOD is applicable to the system. When a SYSMOD is applied, SMP updates the CDS with information about the applied SYSMOD.

   • The SMPCRQ dataset contains data saved from ++IF modification control statements included in previously applied SYSMODs. If the SYSMOD being applied is a function SYSMOD, SMP checks the CRQ to determine if a SYSMOD applied earlier had requirements conditional on the presence of the function being applied; if so, SMP ensures the presence of the SYSMOD(s) named in the earlier SYSMOD's ++IF statements.

If the SYSMOD being applied contains ++IF modification control statements, SMP will save the data from the statements in the SMPCRQ dataset. SMP will use this information if a future SYSMOD satisfies the condition specified on the ++IF modification control statement.

- The SMPSTS (source temporary dataset) and SMPMTS (macro temporary dataset) contain modifications prescribed by SYSMODs which supplied ++SRC, ++SRCUPD, ++MAC and ++MACUPD modification control statements for source and macro elements which are not reflected in the target system libraries. (For example, SYSGEN macros which do not exist on the system libraries.) The user may require these modified source and macro elements for assemblies requiring DLIB macro libraries (such as a SYSGEN).

- The SMPSCDS (save control dataset) contains data that describes the structure of the system libraries before modifications made by SYSMODs which have been applied (for example, the SCDS contains backups for CDS entries changed as a result of a SYSMOD which supplied in-line JCLIN). SMP requires this data to restore the system to its prior level (step 4) if the installation later wants to remove a modification from the system.

4. An installation can remove modifications which have been applied to its operating system using the <u>RESTORE</u> process. SMP updates the SMPCDS dataset after removing the SYSMODs; if those SYSMODs affected the structure of the system libraries, it uses the information stored in the SMPSCDS dataset to restore the libraries to their former structure.

5. Finally, after testing a modification, an installation can install it into the DLIBs (distribution libraries) using the <u>ACCEPT</u> process. Future SYSGENs using these DLIBs will produce the modification-tailored system.

   The SMPACDS (alternate control) and the SMPACRQ (alternate conditional requisite) datasets serve the same purpose for ACCEPT processing as the SMPCDS and SMPCRQ serve for APPLY processing. The SMPACDS dataset reflects the functions and modifications already on the DLIBs; SMP uses this information to ensure the applicability of a SYSMOD being accepted and updates the SMPACDS dataset with information about a SYSMOD when it is accepted. SMP checks the SMPACRQ dataset for conditional requisites relative to other accepted SYSMODs and updates the SMPACRQ dataset with conditional requirements (++IFs) supplied by SYSMODs which are accepted.

A conceptual overview of SYSGEN processing, CDS creation, and SMP processing is shown in Figure 2.

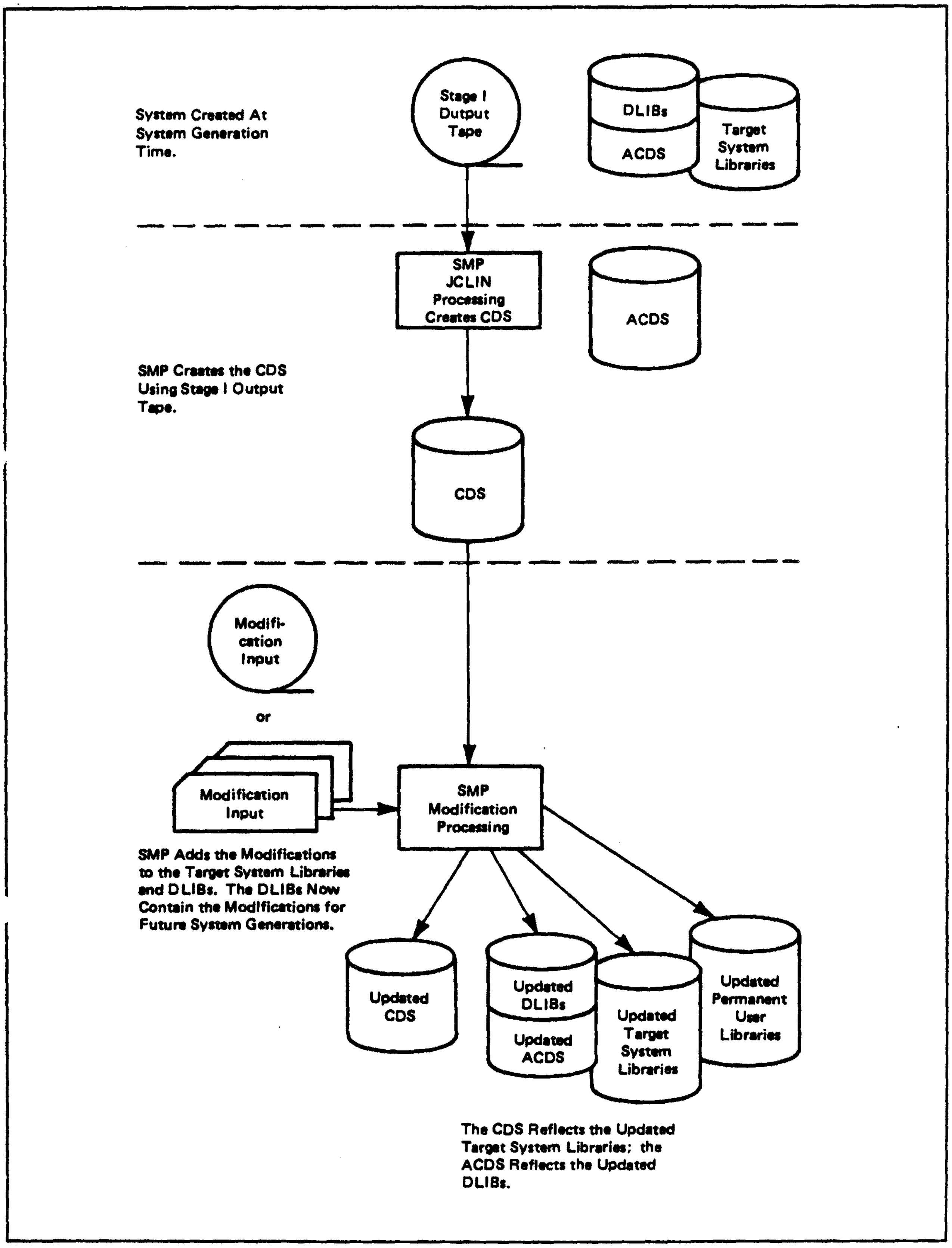

System Created At System Generation Time.
Stage I Output Tape
DLIBs
ACDS
Target System Libraries
SMP JCLIN Processing Creates CDS
ACDS
SMP Creates the CDS Using Stage I Output Tape.
CDS
Modification Input
or
Modification Input
SMP Modification Processing
SMP Adds the Modifications to the Target System Libraries and DLIBs. The DLIBs Now Contain the Modifications for Future System Generations.
Updated CDS
Updated DLIBs
Updated ACDS
Updated Target System Libraries
Updated Permanent User Libraries
The CDS Reflects the Updated Target System Libraries; the ACDS Reflects the Updated DLIBs.

Figure 2 - Processing Overview

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 on chapter Chapter 2.

A simplification of the flow of system modification processing is shown in Figure 3.

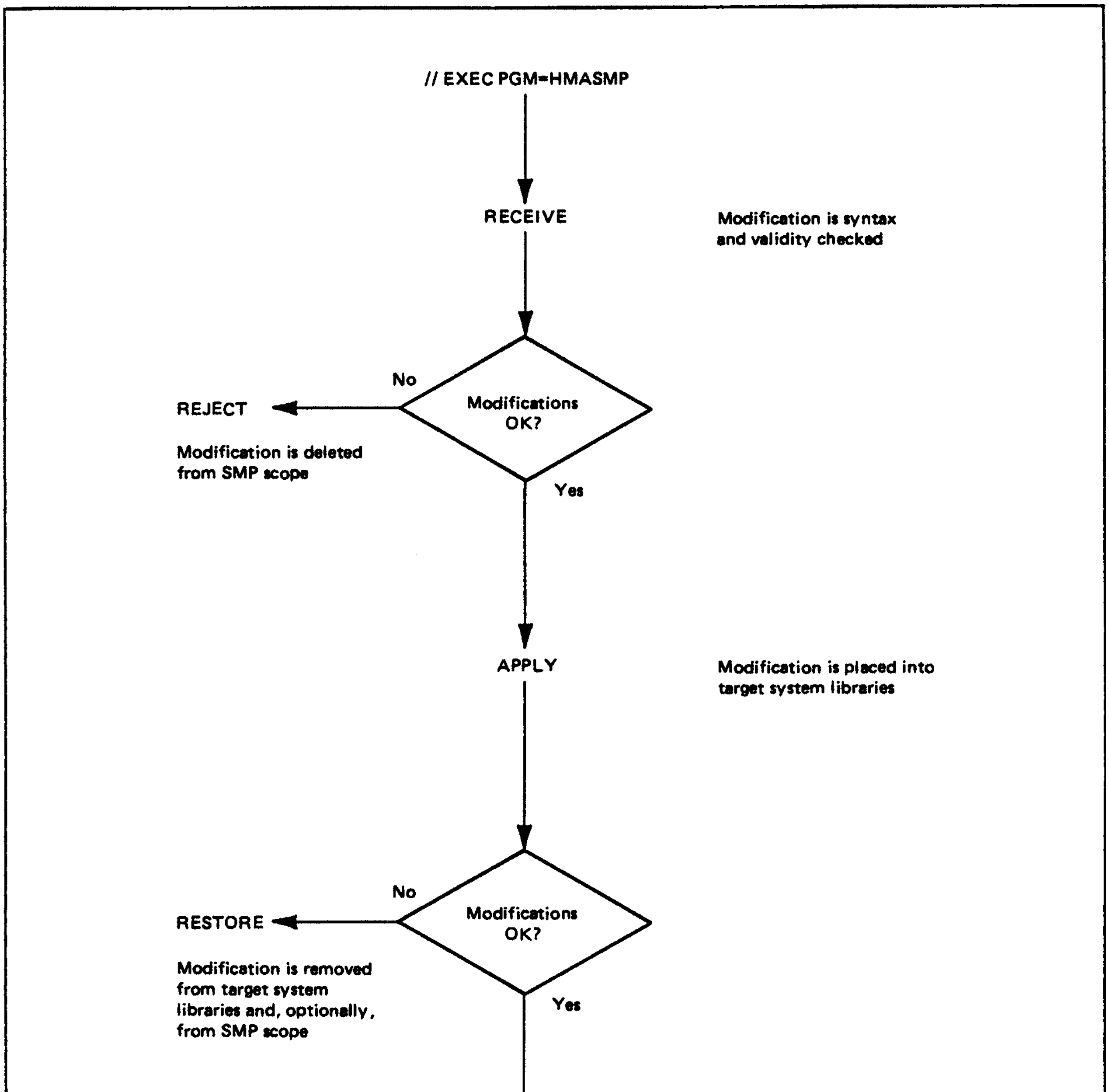

// EXEC PGM=HMASMP
RECEIVE
Modification is syntax and validity checked
Modifications OK?
No
REJECT
Modification is deleted from SMP scope
Yes
APPLY
Modification is placed into target system libraries
Modifications OK?
No
RESTORE
Modification is removed from target system libraries and, optionally, from SMP scope
Yes
ACCEPT
Modification is permanently placed into the DLIBs

Figure 3 - Simplified View of SYSMOD Processing

SMP PROCESSING INTRODUCTION

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 operating system libraries.

- RESTORE - removes SYSMODs processed by APPLY from operating system libraries and restores the elements involved to the level of those elements on the distribution 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.

RECEIVE PROCESSING

RECEIVE processing places SYSMODs in the PTS data set 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 Technique", on page 342 of Chapter 9), 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.

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.

THE RECEIVE CONTROL STATEMENT

The RECEIVE control statement invokes the RECEIVE process. Options available on this statement allow you to SELECT or EXCLUDE specific SYSMODs and allow you to BYPASS the FMID checking indicated below.

THE PTS DATA SET

The PTS serves as a staging data set for SYSMODs and is used to control which SYSMODs are received. The PTS contains a SYSTEM entry which is used to determine the applicability of the SYSMODs presented to the RECEIVE function in terms of the FUNCTIONs on your system. Further, SMP's use of the PTS enables you to receive service SYSMODs that apply to functions that have not been received.

The PTS SYSTEM Entry

The SYSTEM entry in the PTS is required for RECEIVE processing. This entry must be initialized with a system release (SREL) value and a dataset space (DSSPACE) value. The SYSTEM entry also contains FMID values (sub-entries) which indicate the functions which have been received; these FMID values are updated when a function SYSMOD is received and may also be updated using UCLIN.

SYSMOD APPLICABILITY

- A function SYSMOD is received if the SREL value on at least one of the SYSMOD's ++VER statements matches an SREL in the PTS SYSTEM entry.

- A service SYSMOD (PTF, APAR and USERMOD) is received if the SREL and FMID values on at least one of the SYSMOD's ++VER statements matches an SREL and FMID value in the PTS SYSTEM entry.

Note - If a SYSMOD is recorded in the PTS as having been RECEIVED, it cannot be received again. The REJECT function must be used to remove the SYSMOD from the PTS in order to re-receive it.

RECEIVE OUTPUT

SMPOUT

SMP messages and the modification control statements for SYSMODs processed are written to the SMPOUT data set.

When running in SELECT or EXCLUDE mode, the modification control statements for SYSMODs that were either not selected or excluded are generally not written to SMPOUT nor are they checked for syntax. Header modification control statements whose sysmodid operand does not appear on the first record of the header statement are, however, written to SMPOUT even though the SYSMOD may be not-selected or excluded.

SMP REPORTS

A RECEIVE SUMMARY REPORT, summarizing the processing of SYSMODs encountered during RECEIVE processing, is produced on SMPRPT (or SMPOUT if the SMPRPT DD card is not 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 on page 113 in Chapter 4.

PTS SYSTEM Entry

For each function SYSMOD received, the PTS SYSTEM entry is updated to include its SYSMOD-ID as an FMID subentry. This initializes the PTS such that non-function SYSMODs applicable to the function will be subsequently received.

PTS MCS Entry

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

Element text packaged in relative files, TXLIBs or LKLIBs is not moved into the MCS entry. Element text in TXLIBs or LKLIBs remains in these datasets. The processing of elements in relative files is described below.

PTS SYSMOD Entry

For each SYSMOD received, a SYSMOD entry is created containing information required for subsequent SMP processing. The SYSMOD entries do not include the text of the modification or the information from the ++IF modification control statements.

The entries on the PTS are described in Chapter 8.

SMPLOG

A time and date-stamped record of the RECEIVE processing activity is written to the history log data set, SMPLOG.

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" (on page 342 of Chapter 9), 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. 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 on in the SYSMOD entry before the files are loaded, and set off 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 return code 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, 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.

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 must reject it using the REJECT control statement, modify the contents, and receive the modified SYSMOD. You must not attempt to modify the SYSMOD or MCS entries on the PTS using any dataset utility. The SYSMOD entry has been formated by the RECEIVE process and contains information in the user-data portion of its PDS directory entry; any changes made to the MCS information (outside of SMP) will not be reflected in the SYSMOD entry.

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 3.

The purpose of this user exit is to enable you to 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; 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 deletes SYSMODs from the PTS and temporary libraries loaded during RECEIVE processing for those SYSMODs. The COMPRESS option is available on the REJECT control statement and should be used to regain unused space on the PTS after SYSMODs are rejected.

THE REJECT CONTROL STATEMENT

The REJECT control statement invokes the REJECT function. "Mass rejection" is the default if you do not specify the SELECT or EXCLUDE keywords on the REJECT control statement.

Mass Resection without PURGE

SYSMODs which have neither been applied nor accepted are rejected from the PTS. For this processing, the APPID and ACCID subentries in the PTS SYSMOD entries are used to determine whether a SYSMOD has been applied or accepted. An APPID subentry in a PTS SYSMOD entry identifies a CDS to which a SYSMOD has been applied, and an ACCID subentry in a PTS SYSMOD entry identifies an ACDS to which a SYSMOD has been accepted. If a SYSMOD has no APPID and no ACCID subentries in its PTS SYSMOD entry, it will be rejected.

To exclude SYSMODs from rejection, you may specify the SYSMODs you want to exclude as values of the EXCLUDE keyword on the REJECT control statement.

Mass Resection with PURGE

When the PURGE keyword is specified, the SYSMODs rejected are those found as successfully ACCEPTED in the ACDS. Successfully ACCEPTED SYSMODs are those on the ACDS with the ERROR indicator off. The APPID and ACCID subentries in the PTS SYSMOD entries are nee used.

To exclude SYSMODs from rejection, you may specify the SYSMODs you want to exclude as values of the EXCLUDE keyword on the REJECT control statement.

SELECT without PURGE

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 for REJECT is selected regardless of whether any APPID or ACCID subentries are present in the SYSMOD PTS 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 just applied or just 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.

SELECT with PURGE

The SYSMODs selected will be rejected if they are found as successfully ACCEPTED on the ACDS.

Note the distinction in processing when PURGE is coded: the ACDS is used (to find accepted SYSMODs) and no check is made to see if the SYSMODs have been applied.

REJECT OUTPUT

The SYSMOD and MCS entries for rejected SYSMODs are deleted from the PTS. Elements in TXLIB and LKLIB datasets are not affected.

Temporary Library Deletion

When a SYSMOD that had temporary libraries loaded during RECEIVE processing is rejected, the temporary libraries on the SMPTLIB volume are deleted. If the SMPTLIB DD statement is not present, the SYSMOD will not be rejected. If the SMPTLIB DD statement is present but the library to be deleted is not found, a warning message is issued and the associated SYSMOD is rejected. 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.

Updating the PTS SYSTEM Entry

When a function SYSMOD is rejected, the FMID subentry (in the PTS SYSTEM entry) for that function may be removed. As a result of this, non-function SYSMODs dependent upon this function will no longer be received.

The FMID subentry will be removed under the following conditions:

- PURGE is not specified.

- The function SYSMOD has neither been applied nor accepted (using PTS SYSMOD APPID and ACCID subentries to determine whether the SYSMOD has been applied or accepted).

REJECT Messages

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

APPLY PROCESSING

APPLY processing is invoked by the APPLY control statement to install the elements supplied by a SYSMOD on the operating system. The APPLY process installs the elements and updates the SMP data sets to reflect this processing. APPLY processing is controlled by information in the SMP data sets which reflect the status of the target system, information on the SYSMODs which indicates their applicability, and information from the user's APPLY control statement.

APPLY CONTROL STATEMENT

The selection of SYSMODs to be applied is controlled by the keywords specified on the APPLY control statement. These keywords, in conjunction with the applicability criteria listed below, determine which SYSMODs will be applied:

- When the SELECT keyword is used, only those SYSMODs specifically named are considered for application. SELECT may be used to re-apply a SYSMOD.

- When the EXCLUDE keyword is used, all SYSMODs found on the PTS are considered except those specifically named.

- When the GROUP keyword is used, those SYSMODs specifically named and their requisites are considered. (FMID is not a REQ)

  In order to properly determine the ++VER control information to use for SYSMOD installation, group processing will not consider requisites at a higher level in the functional hierarchy. For example, a dependent-level function's REQ for a base-level function will not be satisifed unless the base-level function is already installed or specified in the GROUP list; similarly, a PTF's REQ for a function will not be satisfied unless the function is already installed or specified in the GROUP list.

  In order to prevent the inadvertent installation of functions which delete other functions, group processing will terminate if a requisite SYSMOD (not itself specified in the GROUP list) is found to DELETE a function which is already installed.

- When none of the above keywords are specified, all SYSMODs which meet the applicability criteria will be applied.

SYSMOD APPLICABILITY

In order for a SYSMOD to be considered applicable to the target system, the SYSMOD must have a ++VER whose system release (SREL) matches the SREL of the target system's CDS and whose FMID (if present) names a function which has been (or is being) applied. Function SYSMODs which have no FMID dependency expressed on their ++VER are considered applicable if their SREL matches the SREL of the target system.

- SYSMODs which have already been successfully applied are not considered for re-application unless specifically named in the APPLY SELECT or GROUP keyword.

- SYSMODs which have been partially applied during a previous SMP run are considered applicable and need not be specifically named in the APPLY SELECT or GROUP keyword. Partially applied SYSMODs are identified by the apply ERROR indicator in their SMPCDS SYSMOD entry.

- A SYSMOD which has been partially RESTOREd during a previous SMP run cannot be applied. Partially restored SYSMODs are identified by the ERROR and RESTORE indicators in their SMPCDS SYSMOD entries.

- SYSMODs which are superseded by another SYSMOD are not applied. If a SYSMOD which has not been applied is found to be superseded by another SYSMOD during an APPLY run, no elements from the superseded SYSMOD are processed.

- SYSMODs which have been explicitly deleted cannot be applied.

- A SYSMOD which has more than one ++VER whose SREL and FMID appear to be applicable cannot be processed; SMP is unable to determine which VER statement to use.

- SYSMODs with DELETE on ++VER modification control statements must be selected.

REQUISITE CHECKING

APPLY processing ensures that the proper set of SYSMODs is installed by using requisite data supplied on the SYSMODs' ++VER and ++IF statements.

Unless BYPASS is specified for a particular requisite condition, SMP will not apply a SYSMOD whose requisites are missing.

Requisite conditions are met by any one of the following:

- The requisite SYSMODs are already applied.

- The requisite SYSMODs are superseded by a SYSMOD which is already applied.

- The requisite SYSMODs are being applied.

- The requisite SYSMODs are superseded by a SYSMOD which is being applied

UNCONDITIONAL REQUISITES

Unconditional requisites are those SYSMODs required in any functional environment. These requisites are expressed as operands of the PRE, REQ and NPRE keywords on the ++VER statement of each SYSMOD.

Negatives prerequisites (NPREs) specify that SMP should not APPLY a SYSMOD if the NPRE condition is met. NPREs are valid only on FUNCTION SYSMODS and imply a situation in which two or more function are mutually exclusive.

CONDITIONAL REQUISITES

Conditional requisites are those SYSMODs required only in a particular functional environment.

The ++IF statements immediately following the applicable ++VER modification control statement express the conditional requisite conditions. These statements specify the required SYSMODs (REQ keyword) given the presence of a particular function (FMID keyword).

If the particular function is already applied (or being concurrently applied), SMP will ensure that the ++IF's requisite conditions are met.

To ensure that these requisites are satisfied if the particular function SYSMODs are applied at a later time, SMP saves the information from the ++IF statements on the SMPCRQ dataset. When a function SYSMOD is being applied, SMP accesses the CRQ dataset to determine requisites supplied by previously applied SYSMODs and ensures that these requisites are met.

SYSMOD PROCESSING ORDER

SMP orders the processing of the set of SYSMODs being applied to ensure proper processing of JCLIN, element selection, and source/macro update merges.

The processing order of the SYSMODs being applied is determined from the prerequisite (PRE) data supplied on the SYSMODs' ++VER statements. Those SYSMODs named as prerequisites are processed before the SYSMODs which name them.

If no prerequisite order can be determined between SYSMODs, FUNCTIONs will be processed first followed, in order, by PTFs, APARs, and USERMODs.

DELETED SYSMODS

A function SYSMOD may delete another function by naming the function to be deleted as an operand of the ++VER DELETE keyword. SMP will delete the function and all FUNCTIONs, PTFs, APARs and USERMODs dependent upon the deleted function. The functions specifically named in the DELETE keyword list are considered to be "explicitly" deleted SYSMODs; all SYSMODs deleted because of their dependency upon the explicitly deleted SYSMODs are termed "implicitly" deleted SYSMODs.

SMP delete processing will scratch all macro and source elements from the target system libraries which belong to the deleted SYSMODs. Load modules will be scratched from the target system libraries if all of the modules which make up the load module are deleted.

SMPCDS MOD, MAC and SRC entries representing the deleted elements are removed. SMPCDS LMOD entries are removed if all the modules which make up the load module are deleted.

The SMPCDS SYSMOD entries for all implicitly deleted SYSMODs are removed. An SMPCDS SYSMOD entry is created for each explicitly deleted SYSMOD; this entry will have a DELBY subentry naming the function which caused the deletion. The SYSMOD entries for the explicitly deleted SYSMODs prevent the deleted function SYSMODs from being reprocessed by APPLY.

The result of this process is the 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 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 │
└───────────┘      └─────────┘
```

Figure 4 - DELETE Hierarchy For DELETE(GDE1203)

The conditional requisites (++IF statements) <u>supplied by</u> the deleted SYSMODs remain in the CRQ data set if the environment specified by the ++IF's FMID is still valid. Conditional requisites specifying the functions which are <u>explicitly</u> deleted are removed from the CRQ.

Referring to the above example, when function GDE2000 is applied, any conditional requisites specifying GDE1203 will be removed from the CRQ; conditional requisites specifying functions GDE1303 and GDE1403 will not be removed since these functions are not explicitly deleted (these environments are still valid). Further, conditional requisites supplied by the deleted SYSMODs will not be removed unless they reference the explicitly deleted function, GDE1203.

PROCESSING INLINE JCLIN

Inline JCLIN data for a SYSMOD is supplied following the ++JCLIN modification control statement. JCLIN processing is done prior to element processing in order to initialize the CDS.

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.

Inline JCLIN is not processed for superseded or deleted SYSMODs.

The NOJCLIN option on the APPLY control statement is used to prevent the processing of inline JCLIN. NOJCLIN can be used if the JCLIN contains 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.

The JCLIN function is further described in the "JCLIN Processing" section later in this chapter.

ELEMENT SELECTION

The selection of elements from a SYSMOD is based upon relationships between a SYSMOD being installed and the FMID, RMID and UMID attributes of the corresponding elements installed on the target system.

ELEMENT FMID/RMID/UMID ATTRIBUTES

The FMID, RMID and UMID attributes of an element on the target system are found in the SMPCDS element entries.

The FMID of an element is the FUNCTION-type SYSMOD which "owns" the element. Generally, an element's FMID is established and can be changed only by the installation of a FUNCTION-type SYSMOD; in this case, the element's FMID is the FUNCTION-type SYSMOD which installed the element on the target system.

The RMID of an element is the last SYSMOD which replaced the element (or caused the element's FMID to change). An element's RMID is established by the SYSMOD which first introduces the element to the target system. The RMID of an element is changed by the installation of a SYSMOD which supplies a replacement for the element. Element replacements are ++MODs, ++MACs, ++SRCs, and modules resulting from assemblies.

The UMIDs of an element are the set of SYSMODs which have applied updates to the target system element. A UMID is added to the set of the element's UMIDs for each SYSMOD which applies an update to the element (unless the update changes the element's FMID.) Whenever a new replacement for the element is applied, the set of updates (UMIDs) is cleared to start anew with subsequent updates applied to the new replacement. Element updates are ++ZAPs, ++MACUPDs, and ++SRCUPDs.

FMID OF A SYSMOD

The FMID of a SYSMOD is the function to which all the elements in the SYSMOD belong. The FMID of a SYSMOD being installed is determined as follows:

- The FMID of a FUNCTION-type SYSMOD (++FUNCTION) is the function itself (that is, the sysmod-id enclosed in the parentheses following the ++FUNCTION statement).

- The FMID of a non-FUNCTION-type SYSMOD (++PTF, ++APAR and ++USERMOD) is the FMID from the SYSMOD's ++VER (that is, the sysmod-id which is the operand of the FMID keyword).

The example below illustrates how the. FMID is determined for function and non-function SYSMODs:

```
++FUNCTION(GXY1000)  /* SYSMOD's FMID is GXY1000    */ .
++VER(Z038) .
++MOD ...
      The FMID of all elements in this SYSMOD
      is GXY1000

++FUNCTION(FXY1020) /* SYSMOD's FMID is FXY1020    */ .
++VER(Z038) FMID(GXY1000) /* DEPENDENT FUNCTION    */ .
++MOD ...
      The FMID of all elements in this SYSMOD
      is FXY1020

++PTF(P000000)       /* SYSMOD's FMID is GXY1000    */ .
++VER(Z038) FMID(GXY1000) /* PTF FOR 1ST FUNCTION */ .
++MOD ...
      The FMID of all elements in this SYSMOD
      is GXY1000
```

Element selection in SMP is divided into two cases:

1. FMID of the SYSMOD being installed differs from the FMID of the element on the target system.

2. FMID of the SYSMOD being installed matches the FMID of the element on the target system.

FMIDS DIFFER

In this case, SMP is dealing with elements belonging to different FUNCTIONS, and element selection is based upon functional relationships expressed via FMID and VERSION. Elements may be excluded (i.e., not selected) and processing of the SYSMOD continues under the assumption that a functionally higher version of the element is already installed on the target system.

An element will be <u>excluded</u> from the SYSMOD being installed unless one of the following conditions is met.

1. The function SYSMOD being installed names the FMID of the target system element in the ++VER FMID keyword. In this case, the function being installed is functionally superior to the function which owns the target system element; therefore, the element is selected.

2. The modification control statement associated with the element from the SYSMOD being installed has a VERSION keyword and the target-system element's FMID is named in the VERSION list. In this case, the element from the SYSMOD being installed is considered to be functionally superior to the target-system element, and it is selected.

   If there is no VERSION keyword associated with the element, the sysmod-ids named in the VERSION keyword on the ++VER are used as described above.

   In this situation, SMP may be dealing with either a function SYSMOD or a non-function SYSMOD which is changing the functional ownership (FMID) of the elements.

When an element is selected, its FMID becomes that of the SYSMOD from which it is selected.

FMIDS MATCH - MODID VERIFICATION

In this case, SMP is dealing with elements belonging to the same FUNCTION, and element processing is based upon service relationships expressed via PRE and SUP.

The following checks are made for the elements in a SYSMOD to ensure that a proper relationship exists between the SYSMOD being installed and previously installed SYSMODs which supplied the same elements.

- <u>All</u> elements: The SYSMOD being installed must PRE or SUP the RMID of the associated target-system element. (Note that if the target-system element's RMID is the same as its FMID, the element has not been replaced by any SYSMOD since the installation of a function; in this case, the SYSMOD being installed need not PRE the RMID.)

  If the element being installed is a ++SRC/++SRCUPD or a ++MAC/++MACUPD which results in an assembly which will replace a target-system module (element), the SYSMOD being installed must PRE or SUP the RMID of the corresponding target-system module which would be replaced by the assembly; an exception is made to this requirement if the target-system module is itself the result of an assembly (RMIDASM indicator set in the MOD entry). This exception is allowed since the re-assembly will pick up any changes caused by the SYSMOD which last replaced the module via an assembly.

- Replacement elements: The SYSMOD being installed must PRE or SUP all UMIDs associated with the target-system element.

  As above, assemblies resulting from ++SRC/++SRCUPD and ++MAC/++MACUPD elements are considered to be replacement modules; the SYSMOD being installed must PRE or SUP all UMIDs of the corresponding target-system modules which would be replaced by the assembly. There is no exception made for a SYSMOD which does not PRE or SUP all UMIDs associated with modules which were assembled. No exception is allowed here since any UMIDs associated with the module are ZAPs which would be overlaid by a new assembly.

If either of the above MODID CHECKS is not satisfied, the MODID CHECK ERROR condition is raised and the SYSMOD supplying the offending element(s) is terminated. The user may, however, bypass the termination of the SYSMOD by the specification of BYPASS(ID) on the APPLY or ACCEPT SMP Control statements; in this situation, the MODID CHECK condition is reported to the user (as a warning) and the offending element(s) are installed on the target system.

- Update elements: The SYSMOD being installed need not PRE or SUP the UMIDs associated with the corresponding target-system element. If any target-system element UMIDs are not PREd or SUPd, a MODID CHECK WARNING condition is raised and is reported to the user.

  The MODID CHECK WARNING does not result in the termination of the SYSMOD being installed and the update is installed on the target system.

ELEMENT SELECTION FOR FUNCTION RE-INSTALL

Element selection gets more complicated only for function SYSMODs which are being re-installed and have elements which intersect with corresponding elements having the same FMID as themselves (i.e., "FMIDs MATCH" above.)

The processing for this situation proceeds as in "FMIDs MATCH" above; however, when a MODID CHECK ERROR condition is detected further checks are made to determine whether the service level of the target-system element is higher than that of the element from the SYSMOD being re-installed. If SMP can determine that the service level of the target-system element is higher than that of the SYSMOD element, the element from the SYSMOD being re-installed is excluded, and processing of the SYSMOD is not terminated. If SMP cannot determine that the target-system element's service level is higher than that of the element from the SYSMOD being re-installed, the SYSMOD is terminated with a MODID CHECK ERROR.

ELEMENT PROCESSING

PROCESSING MODULES

The modules (++MODs and assemblies) selected from a SYSMOD are generally link edited to a load module library on the target system. SMP determines the load module with which a module belongs by accessing the CDS MOD entry for the module. The CDS MOD entry contains load module (LMOD) sub-entries which indicate the proper load modules. The link edit characteristics and control statements for the link edit are found in the CDS LMOD entry for the appropriate load module.

If the CDS MOD entry has no LMOD sub-entries, a check is made to determine whether the distribution library (DLIB) for the module was totally copied to a target system library. (SMP determines that a DLIB was totally copied by looking for a CDS DLIB entry describing the module's DLIB.) If this is the case, SMP will create a load module on the target library having the same name as the module's name.

When SMP links a module into a load module which was defined by "link edit" JCL, a linkage editor INCLUDE statement is generated to include the current version of tha load module from the target library. This is done to obtain the other modules which make up the load module. Link edit control statements saved in the CDS LMOD entry are passed to the linkage editor as SYSLIB input.

When SMP links a module into a load module which was defined by "copy" JCL (or was in a totally copied DLIB), no INCLUDE for the current version of the load module is generated.

If the module is supplied on a LKLIB or relative file and the load module was defined by "copy" JCL, SMP will copy the module to the target system. If aliases are specified for such a module, the aliases must exist in the LKLIB or relative file in order for them to be copied.

If SMP cannot determine the load module for a module, it is presumed that the configuration of the target system does not require the module, and no link edit or copy will be done.

Load Module Attributes and Link Edit Parameters

The parameters passed to the linkage editor include load module attributes (RENT, REUS, REFR, DC, ALIGN, SCTR, OVLY, AC=1 and NE) and linkage editor parameters (LET, LIST, XREF and NCAL). SMP determines the attributes and parameters to be passed as follows:

- If LEPARMs are specified on the ++MOD statement, the attributes supplied are used and the corresponding CDS LMOD entry is updated (or created) with these attributes.

- If LEPARMs are not specified and a CDS LMOD entry exists, the attributes from the LMOD entry are used.

- If LEPARMS are not specified and a CDS LMOD entry does not exist (or exists without link edit attribute indicators), the target library is accessed to determine the link edit attributes for the load module. If the load module is not found on the target library, no load module attributes are passed to the linkage editor (unless the user has set some in the PTS SYSTEM entry) .

The parameters passed to the linkage editor are those attributes determined above plus the LKEDPARMs found in the PTS SYSTEM entry.

PROCESSING SOURCE AND MACRO MODIFICATIONS

SMP allows you to receive and apply multiple modifications to the same macro or source elements. 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 merged based on the sequence numbers in columns 73 to 80. The order of the merge is based on the processing order expressed by the PRE keywords on the ++VER statements. If SMP finds a processing order relationship between all of the SYSMODs being processed, the merge occurs according to that order. When no processing 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 processing order relationship with other SYSMODS that do have a processing order, the updates from the unrelated SYSMODs are merged after the updates from SYSMODs that have a processing order relationship.

Library Updating

Macro and source replacements are generally copied to the operating system libraries using IEBCOPY. However, IEBUPDTE is used to process macros which have aliases and source or macro elements having SSI data.

Macro and source updates are performed using IEBUPDTE.

The operating system library for macros and source is found as a SYSLIB sub-entry in the CDS MAC or SRC entries for the respective element types. If no SYSLIB sub-entry is present (and no SYSLIB is specified on the element's modification control statement), a check is made to determine whether the element's distribution library was totally copied to an operating system library. If the distribution library was totally copied to an operating system library (determined by the presence of a CDS DLIB entry for the distribution library), the operating system library to which the DLIB was copied is determined to be the operating system library for the macro or source element. Finally, if no operating system library can be determined, the SMPMTS and SMPSTS libraries are used to store macros and source respectively.

Macro and source elements stored on the MTS or STS remain there until they are replaced by elements from subsequent APPLY processing or until the SYSMODs which modified them are accepted. In this way, the MTS and STS serve as a "macro" libraries for assemblies during APPLY processing and must be in the concatenation of the SYSLIB DD statement for APPLY ("SMP Cataloged Procedure" in Chapter 3 further discusses the concatenation of SYSLIB libraries).

Assembly of Source Text

A SYSMOD may supply both the source (++SRC/++SRCUPD) for an element and an object deck (++MOD) for the same element. SMP will determine whether the object deck may simply be link-edited into the target system or whether the source must be assembled. This determination is made by considering the following questions:

- Has the user specified ASSEM on the APPLY/ACCEPT control statement?
- Are there any UPDATES to the SOURCE which the SYSMOD supplying the OBJECT does not know about? (UMIDs in the CDS Source entry)
- Has another SYSMOD assembled the module which this SYSMOD does not know about? (RMID of the CDS MOD entry for the module to be assembled)
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of the above questions is "yes", the assembly of the module will be done if SMP can determine where the resultant assembled object should go. SMP "knows" where to put the assembled object if there is a CDS MOD entry for the resultant object and a load module into which the module should be link edited (see the discussion of "Processing Modules" above).

Assemblies for MACROs

A SYSMOD may supply macros which require the assemblies of modules. The required assemblies are found as GENASM sub-entries in the CDS MAC entry and in the ASSEM and PREFIX keywords on the ++MAC/++MACUPD modification control statement. The source for these assemblies is found by looking for a CDS ASSEMBLY entry matching the name of the module; if an ASSEMBLY entry is not found, the SOURCE entry matching the name of the module is used as the source for the assembly. If neither an ASSEMBLY nor a SOURCE entry can be found, no assembly will be done.

The SYSMOD may also supply an object deck for the modules to be assembled. SMP will determine whether the assembly should be done rather than using the object decks supplied in the SYSMOD. This determination is made by considering the following questions:

- Has the user specified ASSEM on the APPLY/ACCEPT control statement?

- Are there any UPDATES to the MACRO which the SYSMOD supplying the OBJECT does not know about? (UMIDs in the CDS Macro entry)
- Has another SYSMOD assembled the module which this SYSMOD does not know about? (RMID of the CDS MOD entry for the module to be assembled)
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of the above questions is "yes", the assembly of the module will be done if SMP can determine where the resultant assembled object should go. SMP "knows" where to put the assembled object if there is a CDS MOD entry for the resultant object and a load module into which the module should be link edited (see the discussion of "Processing Modules" above).

Whenever a macro modification in a APAR or USERMOD type SYSMOD causes the assembly of a module, the ASSEMBLE indicator in the modules CDS MOD entry is set on. Subsequent PTF, APAR, or USERMOD type SYSMODs containing modifications to macro or source elements affecting a module whose ASSEMBLE indicator has been set will cause the re-assembly of the module in spite of the presence of an object module in the SYSMOD. The re-assembly will prevent regression of the assembled module during the installation of subsequent SYSMODs which might replace the affected module but not contain the macro modification introduced by the earlier SYSMOD.

If the user wishes to prevent future re-assemblies of modules which have had their ASSEMBLE indicator set, the UCLIN function must be used to set it off (DEL).

REUSE of Previous Assemblies

If SMP is run after a previous failure, assemblies are re-done to ensure that the proper source and macros are used. If the same set of SYSMODs is being processed after a failure, the assemblies done before the failure need not be re-done. The previously assembled objects for the failed SYSMOD will be used if the REUSE keyword is specified in the APPLY control statement.

Assembled object decks are stored on the SMPWRK3 dataset and remain there until the SYSMODs causing the assemblies are successfully processed; in order to be able to reuse assemblies, SMPWRK3 must not be scratched after an apply step.

SMP does not make any checks to ensure that the same set of SYSMODs are being re-done after a failure; a certain amount of user care must be exercises in order to take advantage of the REUSE facility.

Usage of DISTSRC, ASSEM and DISTMOD Operands

Because SMP cannot determine from the data processed by JCLIN what source modules era 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 and PREFIX keyword values specifies a list of source modules and/or CDS ASSEMBLY entries 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 and PREFIX operand lists are performed. If no member is found in either the source target system library or STS, or 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.

ALIAS PROCESSING

When an element with aliases is processed, both the element and its aliases will be updated. SMP does not check the aliases against elements maintained in the CDS. The user must ensure that an element's alias does not match the name of an element maintained by SMP in the CDS.

Aliases for an element are determined as follows:

- REPLACEMENT Elements (MODs and MACs):

1. If a list of aliases is specified on the SMP Modification control statement, these aliases will be used. Aliases (on the target system) which existed before this new list of aliases was presented to SMP will 'not be replaced. Further, this new list of aliases will replace any alias sub-entries in the CDS element entry.

2. If no list of aliases is specified on the SMP Modification control statement, the aliases found as alias sub-entries in the CDS element entry will be used.

- UPDATE Elements (ZAPS and MACUPDs):

  1. If a list of aliases is specified on the SMP Modification control statement, these aliases will be used. Any alias sub-entries in the CDS element entry are ignored for update processing of the element. Macro aliases (on the target system) which existed before this list of aliases was presented to SMP will not be updated. Alias sub-entries in the CDS element entry are not updated or replaced by the aliases in this list.

  2. If no list of aliases is specified on the SMP Modification control statement, the aliases found as alias sub-entries in the CDS element entry will be used.

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, than 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.

COMPRESS PROCESSING

If COMPRESS is specified on the APPLY control statement, the libraries to be affected by the APPLY are compressed by calling IEBCOPY. Before IEBCOPY is called, all load modules and source elements to be replaced are deleted from their target system libraries. Macros are not deleted since termination of the SYSMOD supplying the macros might affect assemblies in other SYSMODs.

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 target system libraries does not occur.

During CHECK processing, the CDS directories are maintained in storage; data is written to the CDS as a temporary storage media. CHECK processing does not re-write the in-storage directories; consequently, no permanent updates are made to the CDS.

APPLY OUTPUT

CDS SYSMOD ENTRIES

Processed SYSMODs

For each SYSMOD processed, an SMPCDS entry is created. If a SYSMOD entry existed previously (as in the case of reapplication of the SYSMOD), the previous entry is replaced. The entry includes 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.

A SYSMOD is considered "successfully" processed when all of its elements have been applied to the appropriate system libraries <u>and</u> all of its requisites have, themselves, been successfully processed. Since SMP will process any number of SYSMODs with elements in common, it is possible that some SYSMODs have elements which need not be applied to a system library; when this is the case, such SYSMODs are not considered "successfully" processed until the SYSMODs supplying the higher level versions of the corresponding elements are successful.

If the SYSMOD is not successfully processed, an ERROR status indicator is set in the entry.

Superseded SYSMODs

When a SYSMOD is superseded by another SYSMOD, a record of this is made by adding the name of the superseding SYSMOD to the SUPBY subentries of the superseded SYSMOD. If the superseded SYSMOD had not been previously applied, the SMPCDS

SYSMOD entry for the superseded SYSMOD will contain only the SUPBY information.

Regressed Element Subentries

A potential regression of modifications in an element exists in the situations described in the MODID VERIFICATION checks described above. When potential regressions of elements are detected, a record is kept for the SYSMOD which had previously modified the element; this record is kept by marking the element sub-entries in the SYSMOD entry as "regressed".

CDS ELEMENT ENTRIES

Apply processing creates, modifies and may delete CDS element entries.

Entry Update Indicator:

When an entry is added by a SYSMOD being processed or modified by in-line JCLIN, the SYSMOD's sysmodid is placed in the LASTUPD subentry of the CDS element entry.

ALIAS Sub-Entries:

The updates to an element's ALIAS sub-entries are discussed under the "Alias Processing" topic on page 34 in this chapter.

LMOD Sub Entries:

When the LMOD operand is specified on a ++MOD modification control statement, the values in the operand list are added to the CDS MOD entry as LMOD sub-entries.

MODID Sub Entries:

- 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 changed when a replacement element (++MOD, ++MAC, ++SRC or assembly) is applied. The RMID becomes the SYSMOD-ID of the SYSMOD supplying the element.

  If a MOD entry is being updated as the result of an assembly for a macro or source the RMID is replaced with the SYSMOD-ID of the SYSMOD supplying the MAC or SRC, and the RMIDASM indicator is set to reflect this occurrence The RMIDASM indicator will be set for the module even if the actual assembly was suppressed because the SYSMOD supplied an assembled version of the module (see "Assembly of Source Text" and "Assemblies for MACROs" above).

  If the replacement element's modification control statement specified an RMID for the element (the RMID keyword), the specified value is used.

- All UMID subentries are deleted when a replacement element is applied.

  If the replacement element's modification control statement specified a list of UMIDs for the element (the UMID keyword), these UMIDs replace any existing UMIDs for the element.

- UMID subentries are added when updates (++ZAP, ++MACUPD, ++SRCUPD) are applied to the element. The UMIDs are the SYSMOD-IDs of the SYSMODs supplying the updates.

  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 is deleted from the element entry.

SMPCRQ

SMP processing adds data to the CRQ dataset for every SYSMOD applied which contains ++IF conditional requisite statements. Two types of entries are maintained on the CRQ:

- FMID Entries

  These entries are organized by function name and contain the names of other SYSMODs which have supplied IF statements which refer to the function.

- SYSMOD Entries

  These entries are organized by SYSMOD and contain the IF statement data supplied by the SYSMOD.

Generally, this data must be kept indefinitely since SMP cannot predict when a function SYSMOD will be installed for which CRQ data has been provided. The only situation in which SMP "knows" that a function SYSMOD will never be subsequently installed is when that SYSMOD is explicitly DELETED by another SYSMOD. Therefore, the management and cleanup of CRQ data is driven by the explicit deletion of function-type SYSMODs.

When a function-type SYSMOD is DELETED by another function-type SYSMOD, the CRQ data will be examined in order to determine what data is still applicable.

- The CRQ FMID entry for each explicitly deleted SYSMOD is scratched.

  The CRQ SYSMOD entries are scratched when all the FMIDs referred to by the IF data in these entries are explicitly deleted.

The description of the CRQ entries on page 323 contains an example of the CRQ entries created by a SYSMOD supplying ++IF statements.

PTS SYSMOD ENTRIES

The CDSID (from the CDS SYSTEM entry) is added as an APPID subentry to the PTS SYSMOD entry of each successfully processed SYSMOD. The APPID sub-entries in the PTS thus reflect the target systems to which each SYSMOD has been applied.

SMPSCDS

Entries are created on the SCDS so that SMP RESTORE processing can restore modifications made to CDS entries if a SYSMOD is restored.

REPORTS AND MESSAGES

Four reports can be produced as a result of APPLY processing and are explained in Chapter 4. In addition, messages are issued to inform you of error and warning conditions that are detected before the reports are produced. The reports 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.

SYSMOD PROCESSING TERMINATION

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

- Missing Requisites:

  The requisite SYSMOD is not available on the PTS (it has not been received).

- The requisite SYSMOD has been EXCLUDED.

- The requisite SYSMOD was terminated (possibly due to other missing requisites).

- The requisite SYSMOD did not meet the applicability criteria.

- The requisite SYSMOD was not included in the SELECT list.

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

• MODID Error Conditions

• DISTLIB operand checking failure.

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

• Utility Return Codes

  Return codes from the utilities invoked to update, assemble, copy and link edit elements to the target system are examined to determine the success or failure of an operation. If these return codes exceed a pre-defined value, the SYSMODs whose elements are involved in the operation are terminated.

• Related SYSMOD Failure

  When SMP excludes an element from a SYSMOD because another SYSMOD being processed supplies a higher level of the element, SMP does not consider the first SYSMOD "successfully" processed until the SYSMOD supplying the highest (selected) level element completes successfully. If the SYSMOD supplying the highest level element fails, all SYSMODs from which elements have been excluded will be terminated because of a "related SYSMOD failure".

AVOIDING TERMINATION OF A SYSMOD

BYPASS

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 SYSMODs should be processed even though their MODID verification checks have failed.

• PRE - specifies that SYSMODs should be processed even though their PRE requisite conditions are not met.

- REQ - specifies that SYSMODs should be processed even though their REQ requisite conditions are not met.

- IFREQ - specifies that SYSMODs should be processed even though their conditional requisite conditions (IFREQs) are not met.

Utility Return Code Thresholds

The values SMP uses to determine the success or failure of an invoked utility program are kept in the SMPPTS SYSTEM entry and may be changed by UCLIN. The default values are shown in the "SMP Data Set Entries" chapter.

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.

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 U200001 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.

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 last accepted 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.

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:

<u>SELECT</u>

In SELECT mode, you must specify the SYSMOD to be RESTORED and all of the requisite SYSMODS that are interrelated which have been APPLIED and not ACCEPTED. When restoring a SYSMOD that superseded a previously APPLIED SYSMOD and that SYSMOD is not ACCEPTED then the superseded SYSMOD is required in the RESTORE list.

Only the SYSMODs that you specify are selected for RESTORE and an incomplete list will result in a RESTORE failure. Therefore a RESTORE CHECK run is recommended prior to the actual RESTORE. In order to synchronize the target system to the level of the distribution libraries, SYSMODs other then those specified in the RESTORE list may be required.

<u>GROUP</u>

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 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 (LASTUPD subentry) 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 is APPLIED and not ACCEPTED, 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 4. 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.

ACCEPT CONTROL STATEMENT

The selection of SYSMODs to be accepted is controlled by the keywords specified on the ACCEPT control statement. These keywords, in conjunction with the applicability criteria listed below, determine which SYSMODs will be accepted:

- When the SELECT keyword is used, only those SYSMODs specifically named are considered for processing. SELECT may be used to re-accept a SYSMOD. SELECT also allows you to accept a SYSMOD which has not been applied.

- When the EXCLUDE keyword is used, all SYSMODs found on the PTS are considered except those specifically named.

- When the GROUP keyword is used, those SYSMODs specifically named and their requisites are considered.

- When none of the above keywords are specified, all SYSMODs which meet the applicability criteria will be accepted.

SYSMOD APPLICABILITY

In order for a SYSMOD to be considered applicable to the target system, the SYSMOD must have a ++VER whose system release (SREL) matches the SREL of the target system's ACDS and whose FMID (if present) names a function which has been (or is being) accepted. Function SYSMODs which have no FMID dependency expressed on their ++VER are considered applicable if their SREL matches the SREL of the target system.

- SYSMODS which are not applied will not be processed unless NOAPPLY is coded or they are specifically called for in the SELECT or GROUP lists. NOAPPLY is required when SYSMODs are initially accepted into the distribution libraries in preparation for a SYSGEN; the SMPCDS dataset is not required when NOAPPLY is coded.

- SYSMODs which have already been successfully accepted are not considered for processing unless specifically named in the ACCEPT SELECT or GROUP keyword.

- SYSMODs have been partially accepted during a previous SMP run are considered applicable and need not be specifically named in the ACCEPT SELECT or GROUP keyword. Partially accepted SYSMODs are identified by the apply ERROR indicator in their SMPACDS SYSMOD entry.

- SYSMODs which are superseded by another SYSMOD are not accepted. If a SYSMOD which has not been accepted is found to be superseded by another SYSMOD during an ACCEPT run, no elements from the superseded SYSMOD are processed.

- SYSMODs which have been explicitly deleted cannot be accepted.

- A SYSMOD which has more than one ++VER whose SREL and FMID appear to be applicable cannot be processed; SMP is unable to determine which VER statement to use.

INLINE JCLIN AND DISTLIB CHECKS

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.

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 Subentry" under "ACCEPT OUTPUT", below).

ELEMENT PROCESSING

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

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.

Load Module Attributes and Link Edit Parameters

The determination of the parameters passed to the linkage editor at ACCEPT differs slightly from that at APPLY.

At ACCEPT, SMP determines the attributes and parameters to be passed as follows:

- If LEPARMs are specified on the ++MOD statement, the attributes supplied are used and the corresponding ACDS MOD entry is updated (or created) with these attributes.

- If LEPARMs are not specified and a ACDS MOD entry exists, the attributes from the MOD entry are used.

- If LEPARMS are not specified and a ACDS MOD entry does not exist (or exists without link edit attribute indicators), the distribution library is accessed to determine the link edit attributes for the module. If the module is not found on the distribution library, "RENT, REUS, REFR, DC" attributes are used.

The parameters passed to the linkage editor are those attributes determined above <u>plus</u> the LKEDPARMs found in the PTS SYSTEM entry.

ACCEPT OUTPUT

SMPSCDS BACKUP ENTRIES

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.

SMPACDS ELEMENT ENTRIES

DISTLIB Subentry:

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

SMPPTS SYSMOD ENTRIES

If the PTS SYSMOD entry is not to be deleted (see below), the CDSID (from the ACDS SYSTEM entry) is added as an ACCID to the PTS SYSMOD entry when processing is successful. The ACCID subentries in the PTS thus reflect the target systems to which the SYSMOD has been accepted.

DELETION OF TEMPORARY LIBRARIES AND SMPPTS ENTRIES

When a SYSMOD which has been successfully applied (indicated by an APPID in the PTS SYSMOD entry) is accepted, the temporary libraries loaded at RECEIVE and the PTS SYSMOD and MCS entries will be deleted if the following conditions are met:

The PURGE indicator is set in the PTS SYSTEM entry and NOAPPLY was <u>not</u> specified.

SMPMTS AND SMPSTS

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

- All of the updates which have been APPLIED have also been ACCEPTED. When this is true, the distribution library version of the element may be used for subsequent assemblies. This situation is detected when the MODID subentries in the CDS element entry match the MODID subentries in the ACDS element entry.

- The NOAPPLY operand was omitted on the ACCEPT control statement.

- The SAVEMTS and/or SAVESTS indicators in the CDS SYSTEM entry are reset.

UCLIN PROCESSING

UCLIN is the SMP function designed to allow the user to make manual modifications to entries on the SMP datasets. Its function, in relation to the SMP datasets, is analogous to the function of IMASPZAP for library load modules.

UCLIN does contain some error checking code. This checking will ensure that the data in each modified entry is consistent with the data that would result from normal SMP processing. Note that the checking done is done only for the one entry changed by each UCL statement. Its affect on other entries is not verified. Before attempting to use UCLIN the user should fully understand SMP processing and the relationships between the various types of entries to ensure that the specified changes are complete.

For example, you may delete an IMOD entry from the SMPCDS via UCLIN but that will not result in the load module being deleted from the target system library. In addition, it will not delete the references to that LMOD from the various other SMPCDS entries. This will result in an error condition later when a SYSMOD is being processed that affects an entry that refers to the deleted LMOD. The SYSMOD can not be installed.

UCLIN is applicable to the following SMP datasets:

- SMPCDS and SMPACDS
- SMPSCDS
- SMPCRQ and SMPACRQ
- SMPPTS
- SMPMTS and SMPSTS

The following terms are referred to in this discussion of UCLIN processing:

SMP dataset	This refers to one of the <u>datasets</u> defined above that are applicable to UCLIN processing.
entry	The term <u>entry</u> refers to a member of an SMP dataset. With the exception of the MCS entry in the SMPPTS dataset, these member names are encoded and cannot be easily accessed by utilities other than SMP. SYSMOD and MACRO entries are examples of the types of entries maintained in the SMPCDS dataset.
sub-entry	A <u>sub-entry</u> is a field within an entry. Each sub-entry has associated with it a type and a value. Multiple occurrences of the same sub-entry type may exist in an entry each with a different value. For example, the modules supplied by a PTF are saved as "MOD" type sub-entries within the PTF's SYSMOD entry. Some sub-entries may occur only once within an entry; for example, the CDSID sub-entry in a CDS SYSTEM entry.

indicator An <u>indicator</u> is a field in an SMP dataset entry that does not have a data value associated with it. An example of an indicator is the APP indicator in the SMPCDS SYSMOD entry. An indicator is either "on" or "off".

UCL statement A <u>UCL statement</u> is the command to SMP that results in a change to one of the SMP dataset entries. UCL statements come between the UCLIN and ENDUCL commands. The UCL statement specifies the action to be taken (ADD, REP or DEL), the entry to be modified and possibly the indicators and sub-entries to be affected.

<u>FUNCTIONS PROVIDED BY UCLIN</u>

The functions that UCLIN can perform are;

- ADD - add

- REP - replace

- DEL - delete

Not all functions are supported for all datasets, nor are all functions supported for all sub-entries and keywords within an entry in a dataset.

The <u>ADD</u> function of SMP is designed to allow the user to either add an entry that did not exist previously or to add a new tub-entry or indicator to an existing entry that did not previously contain that sub-entry or indicator.

- If the ADD function is requested for an entry that already exists then SMP assumes that the request is actually to ADD the sub-entry or indicator specified.

- SMP will check to ensure that each sub-entry value specified does not already exist in the entry. If the sub-entry value specified is found, SMP will issue a message identifying the sub-entry value found and indicating that ADD was requested for an existing sub-entry value.

- For each indicator specified, SMP will check to see if the indicator is "on". If not "on" then it will be set "on". If the indicator is already "on" than an error message will be issued.

The <u>REP</u> function of SMP is designed to allow the user to either replace or add sub-entries or indicators to either an existing or new entry. In the strict sense REP should be used only to replace sub-entries or indicators in existing entries. However, UCLIN REP allows the following;

- If the entry specified is not found then SMP will issue one warning message for that UCL statement indicating that the specified entry was not found and that ADD is assumed.

- If the specified entry is found but no sub-entries of the specified type are found than SMP will issue one warning message for each sub-entry list specified (not for *each* value specified in the sub-entry list) indicating that no sub-entries of that type were found and that ADD is assumed. For sub-entry list the REP function will replace all sub-entry items of the specified type with the list of items specified in the UCL statement.

- For indicators, the indicator is set "on" regardless of the current status of that indicator.

The DEL function of SMP is designed to allow the user to.either delete an entry or delete a sub-entry or indicator from an existing entry.

- If DEL is specified for an entry that does not exist then SMP will issue an error message indicating that the entry could not be found.

  If DEL is specified for a sub-entry in an existing entry SMP will issue an error message for each sub-entry value specifed that was not found.

- If DEL is specified for an indicator in an existing entry but the indicator is not "on" then an error message will be issued. If the indicator was "on" then the indicator will be reset to "off".

- Note that in the general case in order to delete sub-entries in an existing entry that the user must know and specify the existing values for the sub-entry. There is a special provision for sub-entry lists to delete all values in the list without being aware of the current contents of the list. This can be done by specifying the sub-entry operand followed by a left and right parenthesis.

  For example to delete all MOD sub-entries, restore date, time, status and error indicator from SYSMOD UR11111 you could specify:

  ```
  UCLIN CDS.
  DEL SYSMOD(UR11111) MOD() RESDATE() RESTIME() RES ERR.
  ENDUCL.
  ```

UCL STATEMENT PROCESSING FLOW

When SMP°s processing a UCL statement the following actions take place:

- SMP scans the UCL statement and processes the sub-entries and indicators as they are encountered.

- First to be encountered is the type and name of the entry to be processed. At this point the entry is located on the specified dataset and checking is done to determine if the request type (ADD, REP, or DEL) is valid based on whether the entry is found or not.

- Then, as each sub-entry or indicator is encountered SMP checks the in-storage copy of the entry and attempts to make the specified change. At this time error messages may be issued.

- After all requested operands have been processed the resulting in-storage copy of the entry is checked for validity. This checking is done based on the data-set and entry type. The basic intent of this checking is to verify consistency within the entry being updated. Complete checking is not provided with respect to consistency between entries; as a result, improper UCL updates can adversely affect SMP processing.

For example, a SMPCDS SYSMOD entry may not be left with the APP indicator off as that is not a condition that could have resulted via SMP processing and it is inconsistent with the function of the SMPCDS.

During this checking phase SMP will attempt to create a valid entry for the specified dataset by either setting or resetting specified operands. These changes may then result in requirements for other operands which will then generate an error message.

For example, the statement:

```
UCLIN CDS.
DEL SYSMOD(UR11111) APPDATE() APPTIME() APP.
ENDUCL.
```

will result in an entry without the APPLY date or time data. During the checking phase SMP will reset the APP indicator because APP must be on in a CDS entry, and than will issue an error message indicating that APPDATE and APPTIME are required fields that are missing from the resulting entry.

- If after the checking phase no error messages have been issued for this UCL statement then the resulting in-storage copy of the entry is written out to the specified dataset.

- If any error messages are produced during processing of a UCL statement no changes for the specified entry are made.

ENTRY UPDATE INDICATION IN ACDS AND CDS ENTRIES

When an entry is updated by UCLIN, the character string 'UCLIN ' is placed in the LASTUPD subentry.

UCLIN MESSAGES

UCLIN processing produces messages on SMPOUT.

JCLIN FUNCTION

The JCLIN function initializes the CDS dataset with the data required to install elements on the target system. This data is derived by scanning a jobstream containing assembly, copy and link-edit job steps.

The purpose of this section is to describe JCLIN processing and the requirements that SMP has on the input to enable the user to better understand how to construct JCLIN for his own use.

The JCLIN function is invoked either by the SMP JCLIN function control statement or as part of a SYSMOD containing in-line JCLIN statements. When the JCLIN function is invoked, SMP will only update the SMPCDS dataset. It does not invoke the utilities specified in the jobstream. It does not update, modify or look at any of the system libraries.

The following terms are used in this section of the document:

JCLIN input is a jobstream of assembly, link edit and copy jobsteps. When the JCLIN function is invoked by the JCLIN function control statement, this input is pointed to by the SMPJCLIN data definition (DD) statement.

Inline JCLIN refers to JCLIN input within the body of a SYSMOD. Inline JCLIN is identified by the ++JCLIN modification control statement.

The description of the JCLIN function here applies to both the processing for the JCLIN function control statement and the processing of inline JCLIN during SMP APPLY processing. All parameters which may be specified on the JCLIN control statement are also valid as operands on the ++JCLIN modification control statement.

JCLIN PROCESSING OVERVIEW

When the JCLIN function of SMP is invoked SMP begins to read the JCLIN input. It scans the JCL looking for those job steps (i.e. EXEC PGM=xxx or EXEC procname) that contain information that SMP may need. Updating of the SMPCDS is done in the order that the job steps are found in the SMPJCLIN input.

SMP looks for certain program names and proc names that it recognizes as valid assembler, copy, link-edit, and update steps. All other program names and proc names are ignored. If the JCLIN input contains program or procedure names not recognized by SMP, these names can be passed to SMP using operands on the JCLIN control statement or ++JCLIN modification control statement as follows:

- ASM(PGM=asmpgm) or ASM(asmproc)

- COPY(PGM=copypgm) or COPY(copyproc)

- LKED(PGM=lkedpgm) or LKED(lkedproc)

- UPDATE(PGM=updpgm) or UPDATE(updproc)

Note that only one additional program name or procedure name may be specified for each utility. These names are in addition to the names standard names built into SMP and are lower in the search order than the built in names. Thus, if you specify "JCLIN UPDATE(PGM=ASMBLR)", SMP will still treat a job step specifying PGM=ASMBLR as an assembly.

ENTRIES AFFECTED BY JCLIN

SCDS BACKUP ENTRIES

When a CDS entry is updated by in-line JCLIN, a copy of the CDS entry (before any updates) is saved on the SCDS dataset.

ENTRY UPDATE INDICATION IN CDS ENTRIES

When a CDS entry is updated with JCLIN invoked with the JCLIN control statement, the character string 'JCLIN ' is placed in the LASTUPD subentry of each entry updated.

When a CDS entry is updated with in-line JCLIN, the SYSMOD-Id is placed in the LASTUPD subentry of each entry updated.

The LASTUPD sub-entry is used by RESTORE processing to ensure that CDS entries modified by SYSMOD's which are not being restored are not inadvertently overlaid by SCDS backup information.

SMPCDS ASSEM ENTRY

ASSEM entries are created or replaced by JCLIN processing of assembly steps and contain the assembly steps' assembler input statements.

The operation fields (OP codes) of the assembler input are scanned for macro references. SMP will detect a macro reference as an OP code of 6 to 8 characters. These macro references will cause the creation of an SMPCDS MACRO entry as described below.

If an ASSEM entry already exists for a particular assembly, the assembler statements previously saved will be totally replaced by the new set of statements. MACRO entries created or modified by the previous assembly will not be affected.

SMPCDS DLIB ENTRY

DLIB entries are created or updated when SMP processes copy steps for a libraries which are either totally copied or copied with the EXCLUDE option.

The DLIB entry name matches the distribution library that was copied; the entry contains up to two system libraries to which the DLIB was copied.

If a DLIB entry already exists when such a copy step is encountered and the existing DLIB entry has only one SYSLIB, SMP will add a new system library (SYSLIB) to the existing DLIB entry. If an existing DLIB entry has two SYSLIBs, SMP will overlay the second of the two with the new system library determined from the copy step.

SMPCDS MACRO ENTRY

MACRO entries are created or updated for each macro found in the OP code fields of the assembler statements processed in the assembler steps of JCLIN input. Note that MACRO entries are not created from copy steps; SMP assumes that elements found in copy steps are copied modules.

The MACRO entry created contains GENASM sub-entries which reference each assembly that used the macro. After JCLIN processing which creates a new MACRO entry, the only data present in the MACRO entry is the set of GENASM sub-entries. Additional data, such as the distribution library, will be added to the MACRO entry during the installation of the SYSMOD which supplies the actual macro.

If a MACRO entry already exists for a particular macro, a new GENASM sub-entry will be added to the existing MACRO entry referencing the new assembly that used the macro.

SMPCDS MOD ENTRY

MOD entries are created or updated from the processing of copy and link-edit steps.

Each MOD entry contains the module's distribution library (DISTLIB sub-entry) and load modules (LMOD sub-entries) that that module gets link-editted or copied into.

If the module is defined by a copy step, a LMOD entry having the same name as the module will be created (see below).

If a MOD entry already exists for a particular module, link-edit and copy steps will add new LMOD sub-entries for load modules which the module is link edited with or copied to.

SMPCDS LMOD ENTRY

LMOD entries are created during processing of either copy or link-edit steps of JCLIN input.

The LMOD entry name is the load module name; the entry contains the names of the load module's system libraries (SYSLIB sub-entries), link edit attributes and link edit control cards. The LMOD entry can contain up to two system libraries.

If a LMOD entry already exists when a copy or linkedit step is encountered and the existing entry has only one SYSLIB, SMP will add a new system library to the existing LMOD entry. If an existing LMOD entry has two SYSLIBs, SMP will overlay the second of the two with the new system library determined from the copy or linkedit step. Further, any control statements in an existing LMOD entry (except CHANGE and REPLACE) are replaced by the control statements from the new linkedit step (or are deleted altogether if the load module is being redefined by a copy step).

JCLIN JOB-STEP CODING CONVENTIONS

It is not the intent of SMP to be able to support the wide variety of options and facilities supported in job control language (JCL); thus, SMP has some rules as to how the JCL must be coded in order for SMP to process it correctly. If these conventions are not followed the results are unpredictable; JCLIN processing may or may not complete successfully, and follow-on processing may be affected. The conventions that must be followed are:

- DD names pointing to distribution libraries or target libraries must be equal to the lowest level dataset name. This is required because, in the SMP datasets, all references to libraries are by the DD name that points to the library to be updated. If the DD name and the lowest level DSN are kept the same then it will be easier for you to determine what datasets SMP requires. This is also the convention used by IBM when it distributes functions or PTFs, thus if user JCLIN is run that modifies SMP dataset entries, the DDname of the SYSMODs may not match the DDnames in the SMP dataset entries, which will make those SYSMODS unprocessable.

  For example, to point to "SYS1.LINKLIB" the DD statement must be specifies as:

  ```
  //LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
  ```

- Concatenated datasets must not be used for input. For example a link-edit step has an INCLUDE card in its input that says

  ```
  INCLUDE DD1(MODA,MODB,MODC)
  ```

  and has the following DD statements:

  ```
  //DD1 DD DSN=SYS1.USRDLIB1,DISP=SHR
  //     DD DSN=SYS1.USRDLIB2,DISP=SHR
  //     DD DSN=SYS1.USRDLIB3,DISP=SHR
  ```

  In this case SMP will be able to process the JCL but the updates to the SMPCDS will not be sufficient to enable service to be applied. There are two problems with this example. First, the DD names are not equal to lowest level DSN, and second, datasets are concatenated. The correct format for the INCLUDE statement and DD statements are:

  ```
  INCLUDE USRDLIB1(MODA)
  INCLUDE USRDLIB2(MODB)
  INCLUDE USRDLIB3(MODC)


  //USRDLIB1 DD DSN=SYS1.USRDLIB1,DISP=SHR
  //USRDLIB2 DD DSN=SYS1.USRDLIB2,DISP=SHR
  //USRDLIB3 DD DSN=SYS1.USRDLIB3,DISP=SHR
  ```

- Where possible continued utility control statement should not be used. Although SMP attempts to support all existing formats of the utility control statements, it is not able to completely duplicate the syntax checking of the utility. Thus the safest method is to use the simplest format of the utility control statement.

For example, rather than code:

```
 INCLUDE AOS12(HMASMDRV,HMASMDRI,HMASMDR2,           X
              HMASMDC1,HMASMDC2)
```

a better (i.e. safer) method would be to code:

```
 INCLUDE AOS12(HMASMDRV,HMASMDR1,HMSAMDR2)
 INCLUDE AOS12(HMASMDC1,HMASMDC2)
```

Although the linkage editor may accept both formats, the safer format for SMP would be the simplest.

- The DD statement identifying the input control statements for a utility <u>must be</u> the last DD statement in that job step.

  For example:

```
//STEP1 EXEC PGM=IEWL
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12 DD DSN=SYS1.AOS12.DISP=SHR
//SYSLIN DD
  INCLUDE AOS12(HMASMDRV,HMASMDRI,HMASMDR2,...)
  ENTRY HMASMDRV
  SETCODE AC(1)
  NAME HMASMP(R)
/*
```

  would be valid but

```
//STEP1 EXEC PGM=IEWL
//SYSLIN DD *
  INCLUDE AOS12(HMASMDRV,HMASMDR1,HMASMDR2....)
  ENTRY HMASMDRV
  SETCODE AC(1)
  NAME HMASMP(R)
/*
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12 DD DSN=SYS1.AOS12.DISP=SHR
```

  would not be. This is because, for each step encountered, SMP saves all the JCLIN records in a work area until the first non-JCL (i.e. first statement not starting with "//") is found. The saved records are than searched for any information that must be obtained from the JCL. If a JCL statement follows the utility input DD statement, that JCL statement will then not be in the work area and may cause problems to SMP.

ASSEMBLER JOB STEP CODING CONVENTIONS

Assembler steps are identified by one of the following:

- EXEC PGM=IFOX00

- EXEC PGM=IEUASM

- EXEC PGM=ASMBLR

- EXEC ASMS

- EXEC PGM=asmpgm ... from ASM(PGM=asmpgm)

- EXEC asmproc   ... from ASM(asmproc)

As the assembler step is processed all the assembler control statements (i.e. all cards after the "//SYSIN DD *" statement until the end of input ("/*" or "//")) are written out as an SMPCDS ASSEM entry. Note that this requires that the assembler input must be inline, not pointing to another dataset or member of a dataset.

The name of the ASSEM entry is determined by looking at the assembly step JCL. If the step is of the "EXEC PGM=name" type then SMP looks for the

```
//SYSPUNCH DD DSN=...
or
//SYSGO    DD DSN=...
or
//SYSLIN DD DSN=...
```

statements and looks for the member name of DSN or DSNAME library.

For example

```
//SYSPUNCH DD DSN=high.next.low(member)
or
//SYSPUNCH DD DSNAME=high.next.low(member)
```

The SYSPUNCH, SYSGO, or SYSLIN DD statement must point to a PDS and the member must be specified.

If the step is of the "EXEC procname" type, SMP looks for the "MOD=name" operand of the PROC. If the "MOD=name" operand is found, that name is used as the SMPCDS ASSEM entry name. If the "MOD=name" operand is not found, either the "//SYSPUNCH"" or "//SYSGO" or "//SYSLIN" DD statement must be present and the SMPCDS ASSEM entry name will be obtained from it.

While writing the cards out SMP scans each card looking for macro invocations. The following will be recognized by SMP as macros:

- An OP code of 6 to a characters in length.

- The operand of the assembler COPY operation.

For each macro found in the assembly input SMP does the following:

- Locates MACRO entry on SMPCDS. If entry is found then it will be modified. If entry is not found a new entry will be created.

- Update MACRO entry by adding the name of the assembly module(as described above) as a GENASM subentry. This now means that each macro used during the assembly will have a reference back to the SMPCDS ASSEM entry. Thus, when a SYSMOD is processed that modifies that macro, SMP will know what SMPCDS ASSEM entries should be re-assembled in order to include the new macro.

COPY JOB STEP CODING CONVENTIONS

Copy steps are identified by one of the following:

- EXEC PGM=IEBCOPY
- EXEC PGM=copypgm ... from COPY(PGM=copypgm)
- EXEC copyproc   ... from COPY(copyproc)

When a copy step is encountered SMP searches thru the JCL looking for the "//SYSIN DD *" statement. All records from that point till end of input ("/*" or "//") are assumed to be the copy input control statements. Note that this requires that the copy input must be inline , not pointing to another dataset or member of a dataset.

COPY INDD=ddname1,OUTDD=ddname2

When scanning the copy input SMP assumes that the ddnames of the statement are equal to the lowest level dataset name of the dataset referenced. If the COPY statement is set up without this convention, SMP will generate incorrect DLIB and SYSLIB names in the SMPCDS MOD and LMOD entries.

If a COPY statement is followed either by another COPY statement or an EXCLUDE statement, SMP assumes that the INDD library is totally copied to the OUTDD library. Since SMP does not look at either the DLIB or target library datasets during JCLIN it has no way of knowing what modules are contained in the INDD library. Therefore an SMPCDS DLIB entry, with a name equal to the INDD ddname, is created which indicates that the library was totally copied to the library specified by the OUTDD ddname. If the INDD ddname DLIB entry already existed then SMP will add the OUTDD ddname to the DLIB entry. Note that SMP has a restriction that a DLIB can only be copied to two target libraries. Thus if the SMPCDS DLIB entry already has two SYSLIB (i.e. target library ddnames) sub-entries SMP will replace the second entry with the ddname specified by the OUTDD operand.

Now, when a SYSMOD is processed that contains a ++MOD with a DISTLIB equal to the INDD ddname, SMP can create an SMPCDS MOD and LMOD entry with the correct SYSLIB and reference information. Also, if a SYSMOD is processed that contains a ++MAC, ++MACUPD, ++SRC or ++SRCUPD, SMP can fill in both the DISTLIB and SYSLIB fields of the SMPCDS MACRO or SOURCE entry.

If the COPY statement is followed by a SELECT statement, SMP knows exactly which members of the DLIB are being copied (the IEBCOPY select member statements), which DLIB they are being copied from (the IEBCOPY INDD ddname), which target libraries they are going to (the IEBCOPY OUTDD ddname) and the name of the load module in the target library (the IEBCOPY select member statements.)

For each member in the SELECT list SMP will build:

- An SMPCDS MOD entry containing the DLIB module name, DLIB name, and load module name.

- An SMPCDS LMOD entry containing the load module name, target library ddname, and the fact that the load module was copied at SYSGEN. Note that the same restriction of two target libraries is true for the LMOD entry as for the DLIB entry. If two SYSLIB sub-entries exist in the LMOD entry then the second value will be overlaid by the OUTDD ddname.

Note that because SMP only looks at the JCLIN input and has no knowledge about the DLIBs or target libraries built into it, SMP is not able to determine what type libraries are being copied. SMP assumes that all libraries that are selectively copied are load libraries; therefore, MOD and LMOD entries are built for all selectively copied elements. Thus if a macro or source DLIB is selectively copied SMP will not build MACRO entries and SOURCE entries, but will build extraneous MOD and LMOD entries.

LINK EDIT JOB STEP CODING CONVENTIONS

Link-edit steps are identified by one of the following:

- EXEC PGM=IEWL

- EXEC PGM=HEWL

- EXEC PGM=lkedpgm ... from LKED(PGM=lkedpgm)

- EXEC lkedproc   ... from LKED(lkedproc)

When a link-edit step is encountered SMP reads thru the JCL control statements looking for the link-edit control card input. This is done by looking for the "//SYSLIN DD *" statement. All records from that point till end of input ("/*" or "//") are assumed to be linkage editor control statements (note, these must be control statements, not object modules). The "//SYSLIN" input must be inline, it cannot be pointing to a member of another dataset, as SMP would than be unable to analyze the data.

When scanning the input SMP looks for and performs special processing for selected link-edit control statements. The following describes the processing for each link-edit control statement:

- "IDENTIFY" - should not be used.

- "INCLUDE ddname(member,member...)"

  The member names are assumed to be distribution library modules that exist in library 'ddname'. SMP will build SMPCDS MOD entries for each member name specified and will set the DLIB field within each MOD entry to 'ddname'. Note that SMP does not refer back to the 'ddname' DD statement to determine the actual library referred to; the ddname is assumed to match the lowest level dataset name of the library.

The "INCLUDE" cards are not saved in the SMPCDS LMOD entry as they are not necessary when SMP linkedits the load module. All link-edits done by SMP are CSECT replaces so SMP just includes the new version of the updated CSECT and the existing load module from the library.

The ddname, <u>SYSPUNCH</u>, is reserved for includes of object decks produced by assembly steps which are not to be link edited to a DLIB at ACCEPT.

- "INSERT" and "OVERLAY"

  If a load module is to be linked in overlay structure you must supply an "INSERT" control statement for each CSECT in the load module, including INSERT statements for those CSECTs within the root segment. It is not sufficient to properly place the "INCLUDE" and "OVERLAY" control statements.

- "ENTRY"

  Each load module that consists of <u>more than one</u> distribution library module must have an "ENTRY" statement, else the entry point of the load module will change each time the load module is re-linked by SMP.

- "ORDER"

  If the order of CSECTs within a load module is necessary then "ORDER" statements are required to define the load module structure. Simply ordering the "INCLUDE" statements is not sufficient as SMP does CSECT replaces when re-linking the load module and will thus change the order of the CSECTs.

- "CHANGE"

  "CHANGE" cards are saved in the LMOD entry and are associated with the DLIB module name that contains the name to be changed. These statements are passed to the link-editor only when the associated DLIB module is to be replaced in the load module.

- "REPLACE"

  "REPLACE" cards are saved in the LMOD entry and are associated with the DLIB module name that contains the name to be replaced. These statements are passed to the link-editor only when the associated DLIB module is to be replaced in the load module.

- "NAME lmodname(R)" or end of input with no "NAME"

  When SMP encounters either the "NAME" control statement or end of input SMP builds an SMPCDS LMOD entry. The name of the LMOD is determined in different ways depending on the JCL being scanned.

  - If the "NAME" statement is found the LMOD name is determined by the 'lmodname' field of the "NAME" statement.

  - If no "NAME" statement is found and a "//SYSLMOD" DD statement is present SMP gets the LMOD name from the member name of the dataset specified. If no member name is specified SMP will issue an error message identifying the JOBNAME and STEPNAME and reason for the error.

- If no "NAME" statement is found and no "//SYSLMOD" DD statement is found SMP searches for the "MOD=name" keyword in the JCL and uses that name as the LMOD name. If no "MOD=name" keyword is found SMP issues an error mes¯ sage.

- All other statements found in link-edit input

  All other link-edit control statements found will be saved in the SMPCDS LMOD entry in the order that they are encountered and will be passed to the linkage editor whenever SMP needs to re-link this load module.

The link-edit JCL is then scanned looking for the link-edit attributes used to link this load module. These are the options in the "FARM=" field. The attributes that SMP recognizes and saves for future processing are:

- RENT
- REUS
- SCTR
- OVLY
- REFR
- DC
- NE
- AC=1
- ALIGN2

When none of the above attributes are found, the STD indicator is set in the LMOD entry to indicate that the load module should be link edited without any particular attributes.

The target library name for the load module is determined in the following manner:

- If a "//SYSLMOD" DD statement is present the target library DD name is determined by using the lowest level dataset name specified in the "//SYSLMOD" DD statement.

- If no "//SYSLMOD" DD statement is present the name is determined by looking at the "NAME=dsname" option on the pros statement. The DD name used is the lowest level dataset name specified in the "NAME=" option.

- If no "//SYSLMOD" or "NAME=dataset" is found SMP will issue an error message.

Note that the same restriction on the number of libraries that a LMOD can exit in is true for link-edit steps as for copy steps and that SMP will overlay the second library if a third one is presented to it.

The SMPCDS LMOD entry thus constructed contains the load module name, library(s) that it resides in, link¯edit attributes and those link-edit control statements required to re-link the load module.

UPDATE JOB STEP CODING CONVENTIONS

Update steps are identified by one of the following:

- EXEC PGM=IEBUPDTE
- EXEC PGM=updpgm ... from UPDATE(PGM=updpgm)
- EXEC updproc ... from UPDATE(updproc)

When SMP recognizes an UPDATE step it will skip all the JCL until the "//SYSIN" DD statement is encountered or until the next "// EXEC statement is encountered. If the "//SYSIN" statement is found SMP skips all further input until either:

"/*" is found if "//SYSIN DD *"
or
"//" is found if "//SYSIN DD DATA"
or
"xx" is found if "//SYSIN DD DATA,DLM=xx" where xx may be any two characters

This is done so that if the UPDATE step is adding JCL to a library that JCL will not be scanned as part of the JCLIN input.

OTHER JOB STEP JCL

When SMP encounters an EXEC statement that does not execute one of the programs or catalogued procedures documented than it skips all further JCL statements until the next EXEC statement is found.

JCLIN EXAMPLE

Given the job steps shown below, CDS entries will be created as follows:

```
//DEFINE JOB
//A1 EXEC PGM=IEUASM
//SYSLIB   DD DSN=SYS1.AMACLIB,DISP=SHR
//SYSPUNCH DD DSN=&&PUNCH(TESTMOD1),
//            SPACE=(TRK,(1,1,1)),DISP=(,PASS)
//SYSIN DD
TESTMOD1 CSECT
         TESTMAC1 --- INVOKE MACRO
         END TESTMOD1
/*
//L1 EXEC PGM=IEWL,PARM='LET,LIST,NCAL,RENT'
//SYSLMOD DD DSN=SYS1.LPALIB,DISP=SHR
//SYSPUNCH DD *.A1.SYSPUNCH,DISP=(SHR,PASS)
//SYSLIN DD *
 INCLUDE SYSPUNCH(TESTMOD1)
 NAME LOADMOD1(R)
/*
//L2 EXEC PGM=IEWL,PARM='LET,LIST,NCAL'
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12   DD DSN=SYS1.AOS12,DISP=(SHR,PASS)
//SYSLIN DD
 INCLUDE AOS12(TESTMOD2)
 INCLUDE AOS12(TESTMOD3)
 NAME LOADMODX(R)
/*
//C1 EXEC PGM=IEBCOPY
//AMACLIB DD DSN=SYS1.AMACLIB,DISP=SHR
//MACLIB DD DSN=SYS1.MACLIB,DISP=SHR
 COPY INDD=AMACLIB,OUTDD=MACLIB
/*
//C2 EXEC PGM=IEBCOPY
//AOS14   DD DSN=SYS1.AOS14,DISP=SHR
//LINKLIB DD DSN=SYS1.LINKLIB,DISP=SHR
 COPY INDD=AOS14,OUTDD=LINKLIB
 SELECT MEMBER=((LOADMODC,,R))
```

Assembly Step A1

An ASSEM entry named "TESTMOD1" will be created, where the name is derived from the member name on the SYSPUNCH DD statement. This entry will contain the assembler SYSIN data shown.

A MAC entry named "TESTMAC1" will be created since SMP will detect the invocation of the macro in the assembler SYSIN. The macro entry created will contain the name of the assembly as a GENASM sub-entry (which will be used by SMP to determine that this assembly should be performed if and when "TESTMAC1" is updated).

Link Edit Step L1

This step shows the link edit of the previous assembly. From the link edit INCLUDE statement, SMP derives the data required to create a CDS MOD entry representing the module, "TESTMOD1". The module name is determined from the member name operand and the distribution library, "SYSPUNCH", is determined from the INCLUDE statement's DDNAME.

Further, the link edit defines the operating system library for a load module, "LOADMOD1". A CDS LMOD entry is created from the link edit NAME statement and contains the DDNAME of the operating system library derived from the link edit SYSLMOD DD statement. In this case, the LMOD entry name is "LOADMOD1" and the system library (saved as a SYSLIB sub-entry) is determined to be "LPALIB". The load module attribute, "RENT", is saved in the LMOD entry for use in subsequent link edits of this load module; the parameters, LET, LIST and NCAL are not saved.

Link Edit Step L2

The second link edit step defines two modules and one load module to SMP.

Modules "TESTMOD2" and "TESTMOD3" are defined by the link edit INCLUDE statements; the distribution library for each of these is determined to be "AOS12". A CDS MOD entry will be created for each of these modules with a LMOD sub-entry naming the load module, "LOADMODX", and a DLIB sub-entry, "AOS12".

The operating system load module, "LOADMODX", is represented by a CDS LMOD entry. This LMOD entry will be created and entry will contain the operating system library, "LINKLIB", as a SYSLIB sub-entry. No link edit <u>attributes</u> are specified in this step; therefore, the STD indicator is set in the LMOD entry.

Coos/ Step C1

This copy step informs SMP that an entire distribution library is copied to an operating system library. From the INDD operand, SMP determines the DDNAME of the distribution library and creates a CDS DLIB entry named "AMACLIB". From the OUTDD operand, SMP determines the DDNAME of the operating system library and adds a SYSLIB sub-entry, "MACLIB", to the DLIB entry. SMP will use this entry to determine the operating system library for subsequent modifications which specify an element's DISTLIB as "AMACLIB".

Copy Step C2

This copy step illustrates a selective copy of elements from a distribution library to an operating system library.

When a selective copy step is encountered, SMP assumes that the elements involved are <u>modules</u>. As such, SMP will create a CDS MOD entry for each selectively copied element. The module name is derived from the SELECT MEMBER statement; the distribution library for such modules is determined from the operand of the copy INDD statement; the load module name is simply the module name. In this case, a MOD entry named "LOADMODC" is created; the distribution library is determined to be "AOS14", and the LMOD is "LOADMODC".

Finally, a CDS LMOD entry will be created to represent the operating system library to which the module is copied. The load module name (and the CDS LMOD entry name) is the module name, "LOADMODC". The operating system library (saved as a SYSLIB sub-entry) is determined from the operand of the copy OUTDD keyword; in this case, "LINKLIB". LMOD entries created from copy job-steps do not have any link edit attributes. Rather, an indicator (COPY) is set in the entry to inform SMP that the link edit attributes must be obtained by examining the operating system library when the module must first be link edited.

SMP INSTALLATION AND USE

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

- IBM Operating Systems and Distribution Libraries
- Executing SMP
- SMP Primary Data Set Initialization
- SMP Processing Parameters and Options
- Special Processing Considerations
- Examples of Using SMP to Install System Modifications
- User Written Exit Routines
- Primary Data Set Allocation Guidelines (MVS and VS1)

IBM OPERATING SYSTEMS AND DISTRIBUTION LIBRARIES

An IBM operating system consists of a set of function SYSMODS made up of modules, macros and source elements. 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 Chapter 2 and Chapter 6.

The system is generated by a system generation process (SYSGEN) using a set of distribution libraries (DLIBs) containing modules that are assembled or link edited into system data sets. The system configuration is determined by parameters you supply the SYSGEN process. 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.

The distribution libraries required for a SYSGEN are built by SMP from a set of distribution tapes containing modules, macros and source elements. In order to build the distribution libraries with SMP,

1. the SMPPTS, SMPACDS and SMPACRQ must be initialized as described in the "Primary Data Set Initialization" section below, and
2. the SYSMODs on the distribution tape must be ACCEPTED.

The system generation process uses these distribution libraries to create a system control program tailored to the data processing and machine configuration requirements of 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 the operating or target system, and the level of the system created is referred to as a base level system.

Following the SYSGEN, the SMPCDS and SMPCRQ data sets must be initialized for subsequent installation of modifications to the operating system built by the SYSGEN.

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.

SMP CONTROL STATEMENTS

SMP Control statements are used to invoke SMP processing and are placed in the data set described by the SMPCNTL DD statement.

SMP MODIFICATION CONTROL STATEMENTS

Modification Control statements describe the SYSMODs to be processed by RECEIVE, APPLY and ACCEPT. These statements all begin with '++', such as ++PTF, and are placed in the data set described by the SMPPTFIN DD statement.

SMP OUTPUT

Output from SMP may go to one of three files, SMPOUT, SMPRPT and SMPLIST.

- SMPOUT - lists the control statements entered in SMPCNTL and displays SMP messages.
- SMPRTP - contains the REPORTS produced by RECEIVE, APPLY, ACCEPT and RESTORE.
- SMPLIST - contains the listings produced by the SMP LIST function.

SMPOUT is required. SMPRPT and SMPLIST are optional; if either is not present, the associated output will be diverted to SMPOUT.

SMP Messages and Return Codes

The return code set by SMP is determined by the conditions encountered during the processes invoked by the user. Since many different conditions can be encountered during an SMP invocation, the final return code presented by SMP is the return code associated with the most severe condition encountered.

Various situations and unusual conditions are reported by messages to the SMPOUT file. These messages fall into one of five categories associated with the severity of the condition detected:

- Informational - Messages in this category generally indicate stages of SMP processing or accompany other messages to further explain unusual conditions.

- Warning - Messages in this category indicate that SMP has detected a situation which may be invalid. A return code of 4 is associated with such situations. The user should examine these messages to determine whether the action taken by SMP was appropriate.

- Error - Messages in this category indicate that some SMP processing did not complete properly. A return code of 8 is associated with such situations. For example, the termination of a SYSMOD during APPLY is always *an* error condition and results in a return code of at least 8.

- Severe - Messages in this category indicate that an entire SMP function failed. A return coda of 12 is associated with such situations. For example, the termination of the APPLY function due to insufficient storage results in a return code of at least 12.

- Terminating - Messages in this category indicate that a situation occurred which forces SMP to terminate. A return code of 16 is associated with such a situation.

SMP JCL AND EXEC STATEMENT PARAMETERS

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 (page 361) for a description of the usage of this parameter.

  'NORECOVERY'
    specifies that SMP not invoke its STAE/ESTAE recovery environment. This parameter may be specified only when SMP is executing on a non-VS1 or non-MVS operating system. SMP functions which use instorage directories should be invoked specifying DIS(READ) or DIS(NO) since no recovery environment is established to re-write updated directories if the SMP function ABENDS.

- The DD statements specify the data sets that are required by or are optional for the SMP function. See Chapter 7 for information about the data sets and the ddnames associated with each.

SMP Cataloged Procedure

Figure 5 is a sample SMP cataloged procedure that can be placed in a cataloged procedure library and used during the execution of an SMP job.

SMP cannot use the same cataloged procedure for both APPLY and accept (see Note 1 below).

```
//SMPJOB   PROC
//SMPSTEP EXEC PGM=HMASMP
//SYSPRINT DD   SYSOUT=A
//SMPOUT   DD   SYSOUT=A
//SMPRPT   DD   SYSOUT=A
//SMPLIST  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
//SMPPUNCH DD   SYSOUT=B
//SMPTLIB  DD   DISP=OLD,UNIT=3330,VOL=SER=(333001,333002)
//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         SEE NOTE 1 (below) *********
//* ---- Include DD Statements for Distribution Libraries ----
//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 DD Statements for Target Libraries
//LINKLIB  DD   DSN=SYS1.LINKLIB,DISP=OLD
                  .
                  .
//* ---- Include DD Statements for Utility Datasets
//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)
// PEND
```

Figure 5: Sample SMP4 Cataloged Procedure (see notes below)

Notes - SMP4 Cataloged Procedure

1. The proper SYSLIB concatenation is required for assemblies done at APPLY and ACCEPT.

   The SMPMTS dataset contains macros from SYSMODs which are applied; therefore, the proper SYSLIB concatenation for APPLY includes the SMPMTS dataset as shown below:

   ```
   //*        Include SMPMTS, Operating System and DLIB Macro
   //*        Libraries
   //SYSLIB    DD  DSN=SYS1.SMPMTS,DISP=OLD
   //          DD  DSN=SYS1.MACLIB,DISP=OLD
   //          DD      .
   //          DD      .
   //          DD      .
   //          DD  DSN=SYS1.AMACLIB,DISP=OLD
   //          DD  DSN=SYS1.AMODGEN,DISP=OLD
   //          DD      .
   //          DD      .
   //          DD      .
   ```

   At ACCEPT time, the macros on the SMPMTS dataset must not be used; therefore, the proper SYSLIB concatenation for ACCEPT does not include the SMPMTS dataset as shown below:

   ```
   //*        Include All Distribution Library Macro Libraries
   //SYSLIB    DD  DSN=SYS1.AMACLIB,DISP=OLD
   //          DD  DSN=SYS1.AMODGEN,DISP=OLD
   //          DD      .
   //          DD      .
   //          DD      .
   ```

2. An SMPCNTL DD statement describing the SMP control statement input is required for all SMP processing.

3. An SMPPTFIN DD statement describing the input containing SYSMODS is required for RECEIVE processing.

4. An SMPTLIB DD statement is used by SMP to define a volume, or volumes (up to five), to be used for the allocation of space needed to load partitioned data sets during RECEIVE processing of SYSMODs packed in RELFILE format.

5. DD statements describing datasets containing elements in LKLIBs and TXLIBs are required for APPLY and ACCEPT processing.

6. The SMPACDS DD statement is not required for an APPLY, RECEIVE, or REJECT procedure.

7. The SMPCDS DD statement is not required for an ACCEPT, RECEIVE, or REJECT procedure.

8. The SMPMTS DD statement is required for modifications to macros not residing in a target system library.

9. The SMPSTS DD statement is required for modifications to source modules not residing in a target system library.

10. The SMPPUNCH DD statement is required for the UNLOAD function.

11. The DD statements for target system data sets are not required for a RECEIVE. REJECT, or ACCEPT procedure.

12. The DD statements for distribution library data sets are not required for a RECEIVE, REJECT, or APPLY procedure.

13. The SMPWRK3 dataset may be permanently allocated to take advantage of the assembly REUSE facility. (See "Assembly REUSE Facility" later in this chapter.)

INCLUDING THE REQUIRED SYSTEM PROGRAMS

The following system programs are required for SMP processing:

- 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 and "Special Processing Considerations" later in this chapter)
- IMASPZAP (See OS/VS1 Service Aids or OS/VS2 System Programming Library: Service Aids)

These programs must be in an authorized library on the system under which SMP is executing. They must be accessible to SMP using LOAD (that is, in a STEPLIB, JOBLIB, LINKLIB or LPALIB). If the required programs are not available, SMP will terminate. When SMP is executed under VS/1, SMP will lose its authorization if any of the above utilities are loaded from a non-authorized library. This may result in SMP termination since SMP uses some authorized functions.

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 the SCDS.

WARNING

The SMP Primary data sets contain encoded member names and data. They must not be modified by any partitioned data set utility or editor!

Primary Data Set Requirements - RECEIVE

For the RECEIVE function, the only required primary data set is the SMPPTS. The PTS must have an initialized SYSTEM entry for RECEIVE.

Primary Data Set Requirements - APPLY

The PTS must have a SYSTEM entry.

The CDS must have a SYSTEM entry (initialized by SMP Release 4) with a CDSID sub-entry and an SREL sub-entry matching one of the SREL sub-entries in the PTS.

The CRQ and SCDS data sets must be present and should correspond to the operating system defined in the CDS. SMP will ensure their presence; the user must, however, ensure that they match the CDS.

Primary Data Set Requirements - ACCEPT

The PTS and the CDS (if present) must meet the requirements outlined for APPLY (above).

The ACDS must have a SYSTEM entry (initialized by SMP Release 4) with a CDSID sub-entry and an SREL sub-entry matching one of the SREL sub-entries in the PTS.

The ACRQ data set must be present and should correspond to the operating system defined in the ACDS. SMP will ensure its presence; the user must, however, ensure that it matches the ACDS.

Primary Data Set Requirements - RESTORE

The PTS, CDS, ACDS, CRQ and SCDS must meet the requirements outlined for both APPLY and ACCEPT (above).

SMP PRIMARY DATA SET INITIALIZATION

The ACRQ, CRQ, MTS, STS and SCDS need only be allocated as empty partitioned data sets as described in 'SMP Data Set Allocation Requirements' later in this chapter. You must, however, initialize ACDS, CDS and PTS data sets using SMP as illustrated below.

NULL PTS

The SMP RECEIVE function places SYSMODs into the SMPPTS data set for subsequent processing and is thus the first SMP process in the installation of system software.

The PTS data set must be allocated as a partitioned data set (see Chapter 8) and initialized using UCLIN (as illustrated below):

```
UCLIN PTS .
ADD SYS SREL(Z038) DSSPACE(10,20,30) .
ENDUCL .
```

- SREL(Z038) sets the system release for MVS 3.8. The program directory supplied with the distribution tapes will specify the appropriate system release value.

- DSSPACE(10,20,30) sets values for SMPTLIB dataset allocation required for receive processing of SYSMODs packaged using RELFILES. SMP will attempt to allocate a data set on the SMPTLIB volume(s) for each relative file on the RELFILE tape; as illustrated, the allocation request will be 10 primary tracks, 20 secondary tracks and 30 directory blocks.

  The values used for DSSPACE depend upon the sizes of the libraries supplied on the RELFILE tape; guidance in the determination of the appropriate values should be found in the program directory accompanying the distribution tapes.

Many other processing options may be set in the PTS SYSTEM entry. The UCL SYS statement description on page 211 in Chapter 5, provides an explanation of all subentries and the default indicator settings for this entry.

NULL ACDS

In order to ACCEPT SYSMODs into the distribution libraries in preparation for a SYSGEN, the SMPACDS data set must be allocated (see Chapter 8) and initialized using UCLIN as illustrated below:

```
      UCLIN ACDS .
      ADD SYS SREL(Z038) NUCID(2) CDSID(SYSTEM1) .
      ENDUCL .
```

- SREL(Z038) sets the system release for MVS 3.8.
- NUCID(2) sets the alternate nucleus identifier.
- CDSID(SYSTEM1) places an identifier in the ACDS system entry which is displayed by the SMP list facilities. Further, when a SYSMOD is accepted, this indicator is placed in the PTS SYSMOD entry for the SYSMOD indicating that the SYSMOD has been accepted on "SYSTEM".

The UCL SYS statement description on page 211 in Chapter 5, provides an explanation of all subentries and indicators in the ACDS SYSTEM entry.

CREATING THE CDS AND CRQ

After you have created a target system by performing a SYSGEN, you must create the CDS and CRQ control data sets which maintain the status of the operating system libraries.

The data in the ACDS and ACRQ was initialized by the SMP ACCEPT of the SYSMODs into the distribution libraries and thus also reflects the status of the newly built operating system libraries. The CDS is created by copying all of the entries from the ACDS to the CDS; The CRQ is created by copying all of the entries from the ACRQ to the CRQ.

The CDS must also be initialized with data which describes the load module structure of your particular operating system. This initialization is accomplished by the SMP JCLIN function using the SYSGEN Stage I output jobstream as input to SMP. JCLIN is invoked by the JCLIN control statement (see page 140) with the SMPJCLIN DD statement pointing to the data set containing the SYSGEN Stage I jobstream. (JCLIN processing is further described in Chapter 2.)

SMP PROCESSING PARAMETERS AND OPTIONS

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 below. 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.

PEMAX DETERMINATION - RECEIVE

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.

PEMAX DETERMINATION - APPLY/ACCEPT/RESTORE

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.

SYSTEM PROGRAMS AND PARAMETERS

The system programs invoked (see "Including the Required System Programs", on page 79) and the parameters passed to these programs may be changed by modifications to the PTS SYSTEM entry.

The "UCL - SMPPTS SYSTEM Entry" section on page 229 describes the system program and parameter defaults.

ESTABLISHING SYSTEM PROGRAM 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 - SMPPTS SYSTEM Entry" section on page 229 describes the return code 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 5 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 CHTL 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 5.

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 mach 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 three cases:

  1. 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.

  2. When the amount of storage necessary to hold the directory is not available.

3. Non-VS/1, non-VS/2 environment - no STAE/ESTAE (See page 76).

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 generally 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 2 of the OS/VS SMP Messages And Codes describes this problem in detail.

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 a successful 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 description.

The RETRY facility is activated by the RETRY keyword on the APPLY, ACCEPT and RESTORE control statements.

The utility routines (IEBCOPY, IEWL, IEBUPDTE, ETC) are invoked as attached tasks. SMP is not terminated if one of these utility routines fails during processing. If one of the utilities fails with an X37-04 ABEND, SMP will attempt to compress the failing dataset and continue processing.

The user may specify the name of the program to be used for compress during RETRY recovery (RETRYNAME in the PTS SYSTEM entry). If no RETRYNAME is specified, 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 will terminate if the SYSUT4 DD statement is not present. Tha 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.

- The datasets eligible for RETRY processing may be controlled by the RETRYDDN subentries in the CDS and ACDS SYSTEM entries:

  The list of DD names eligible for X37 RETRY processing must be 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 by removing all DD names from the appropriate system entry.

You 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' on page 105 of this chapter) or to reinvoke the failing utility without doing the COMPRESS operation.

RELFILE TAPE TLIB DATA SET ALLOCATION (DSSPACE)

The DSSPACE sub-entry in the PTS SYSTEM entry defines the primary, secondary and directory block allocation required for receiving SYSMODs packaged using RELFILES. DSSPACE specifies the amount of space SMP will attempt to allocate for <u>each</u> relative file on the RELFILE tape and therefore must be large enough to contain the largest unloaded relfile data set.

ASSEMBLY REUSE FACILITY

When assemblies are done during processing a set of SYSMODs, the assembler output (object deck) is stored on the SMPWRK3 data set. If SMP fails before the SYSMODs are successfully processed, these assemblies need not be re-done when the same set of SYSMODs is re-installed.

The reuse facility provides the option of not re-assembling the previously successful assemblies and can be very valuable if certain precautions are heeded:

- The SMPWRK3 data set must be allocated as a "permanent" data set; otherwise, the objects saved there will be lost after a failure.

  It is recommended that the WRK3 data set be scratched and re-allocated before any "normal" (non-REUSE) SMP run. For the "normal" SMP run, the data sat should specify a disposition of KEEP.

  If this "normal" run fails and you wish to re-use the successful assemblies, modify the APPLY or ACCEPT control statement for the failing run by including the REUSE keyword.

- When re-processing the failing SYSMODs, the user must ensure that exactly the same set of SYSMODs are being processed as was being processed before the failure. This ensures that no new modifications are introduced which could affect the previous assemblies. SMP does not check to ensure that this condition is met!

  Re-run the failing set of SYSMODs immediately without RECEIVing or RESTORing any SYSMODs which could affect the previous assemblies.

SPECIAL 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.

SYS1.NUCLEUS STORAGE REQUIREMENTS - MVS/VS1 CONTROL PROGRAM

Before modifications are made to IEANUC01 in SYS1.NUCLEUS, SMP renames IEANUC01 to IEANUCxx, where xx is a number specified in the NUCID sub-entry of the CDS SYSTEM entry. This facility allows you to IPL your system with the old nucleus if the modifications made to IEANUC01 prevent an IPL.

The SYS1.NUCLEUS data set should be large enough to contain at lease three copies of the IEANUC01 member. This will provide space for IEANUC01 and IEANUCxx and allow space for one additional link edit of the load module.

If the miniumum amount of space is allocated for the nucleus data set, it must be compressed after each modification to it.

IEHIOSUP FOR VS1

For V51, 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. This dataset must be authorized on the system that SMP is executing on

  - 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.

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).', 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.

- Execute the jobstream produced by Stage I.

- 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:

```
S [SMPIO | SMPNUC],id,aaaaa,vvvvvv,,dsn=dddddddd
```

where

id - specifies the partition number.

aaaaa - specifies the device address or device type of the CDS. The default is SYSDA.

vvvvvv - specifies the volume serial number of the CDS device. This parameter is required.

dddddddd - specifies the data set name of the CDS. The default is SMPCDS.

You can use the following JCL to invoke SMPIO or SMPNUC:

```
//    EXEC [SMPIO | SMPNUC]
//IEFPROC.IEFRDER DD VOL=SER=vvvvvv,UNIT=aaaaa,DSN=dddddddd
```

where

vvvvvv - specifies the volume serial number of the CDS device. This parameter is required.

aaaaa - specifies the device address or device type of the CDS. The default is SYSDA.

dddddddd - 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.

BACKUP OF SMP DATASETS

From the backup point of view, it is important that backups of SMP4 datasets be maintained in synchronism so that all the datasets reflect the same status. This may mean restoring several datasets at once in the event of an error in a single dataset. If the backups are recent, this is generally simpler than recovering the situation using SMP unless the user is confident that he understands the cause of the error and the steps to perform the recovery.

The following datasets should be backed up and restored together to insure synchronism:

- SMPCDS, SMPCRQ, SMPSCDS, SMPMTS, SMPSTS and the target system libraries.
- SMPACDS, SMPACRQ and the distribution libraries.

Two satisfactory options for backup are described below.

IEHDASDR DUMP

Ideally the SMP4 datasets should be maintained in synchronism with the system itself (the target and distribution libraries.) IEHDASDR may be used to dump the volumes which contain the system and SMP datasets. If a problem is encountered, the entire system may be restored to a point before the problem, and therefore one is sure that the system and SMP datasets are properly synchronized.

Although a complete system restore can be rather disruptive, this sort of backup is desirable when the system must be brought back to the known backed-up level. It is advisable to create a system restore backup before a large set of modifications (such as a PUT or function install) is attempted.

IEBCOPY UNLOADS OF INDIVIDUAL SMP DATASETS

IEBCOPY backups can be maintained of all SMP4 datasets on tape. Although back-up does not provide the synchronism with the target system described above, it does provide insurance against loss of the SMP datasets due to problems such as un-recoverable I/O errors.

The SMPLOG dataset should be excluded from the data set restore process so that it can maintain a correct log of all processing including the error situation. Use should be made of the 'write-to-log' facility (LOG control statement) so that a comment about the restore of the SMP4 datasets can be recorded in the log. The restore of the datasets and the 'write-to-log' can easily be automated via a JCL procedure.

SMP USE EXAMPLES

This section presents some examples to illustrate the use of SMP in the installation and maintenance of system software.

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 data and time stamped. See the 'LOG Control Statement' in Chapter 5 for further information.

RECEIVE

The SMP receive function places SYSMODs into the SMPPTS data set for subsequent processing and is thus the first SMP process in the installation of system software.

RECEIVE ALL SYSMODS FROM SMPPTFIN

In order to receive all SYSMODs which are applicable to your SYSTEM, the RECEIVE control statement is coded with no additional keywords,

```
      RECEIVE .
```

SYSMODs which have not been received on the PTS and meet the following criteria will be received:

- All function SYSMODs whose system release (++VER SREL operand) matches an SREL in the PTS SYSTEM entry.

- All non-function SYSMODs whose system release matches an SREL and an FMID in the PTS SYSTEM entry.

RECEIVE SELECTED SYSMODS

The following RECEIVE control statement illustrates how only a certain set of SYSMODs from SMPPTFIN input may be received:

```
      RECEIVE SELECT(UZ00001,UZ00002) .
```

In this case, SYSMODs UZ00001 and UZ00002 will be received if they are in the PTFIN input and they are not already received in the PTS data set.

RECEIVE SYSMODS WHICH HAVE ALREADY BEEN RECEIVED

SMP will not receive a SYSMOD which is found on the PTS as successfully received. If you desire to re-receive a SYSMOD, the SYSMOD must be deleted from the PTS using the reject function:

```
      REJECT S(UZ00000) /* DELETE SYSMOD UZ00000 FROM PTS    */ .
```

RECEIVE PTFS PRIOR TO FUNCTION INSTALLATION

In preparation for the installation of new function, it may be desireable to begin receiving SYSMODs for the function before the function itself is received. To accomplish this, the PTS SYSTEM entry must be modified to include the FMID of the function for which PTFs are to be received:

```
      UCLIN PTS .
      ADD SYS FMID(F000002) .
      ENDUCL .
```

FMID(F000002) places an FMID value in the PTS to enable SMP receive processing for SYSMODs belonging to function F000002.

RECEIVE PTFS FOR VARIOUS SYSTEMS

Additional system release values can be specified such that SYSMODs applicable to various systems may be stored in one PTS dataset.

```
      UCLIN PTS .
      ADD SYS SREL(X070) .
      ENDUCL .
```

SREL(X070) places an SREL value in the PTS to enable SMP receive processing for SYSMODs applicable to X070 (VS1 Release 7) systems. Checks during APPLY and ACCEPT will ensure that the SYSMODs are installed on the appropriate system.

APPLY

The SMP apply function installs SYSMODs on the target operating system.

MODIFICATION OF CDS IDENTIFIER (CDSID)

The CDS Identifier (CDSID) may be changed to distinguish between various systems being maintained in one installation. The CDSID may be changed using UCLIN:

```
      UCLIN CDS .
      REP SYS CDSID(TEST1) .
      ENDUCL .
```

When a SYSMOD is applied, the indicator is placed in the PTS SYSMOD entry for the SYSMOD indicating that the SYSMOD has been applied to "TEST1" .

MASS APPLY

In order to apply all SYSMODs which have been received but not yet applied, the APPLY control statement is coded as

```
      APPLY .
```

GROUP APPLY

To apply one particular PTF, P000002, and all its requisites.

```
      APPLY GROUP(P000002) .
```

Since PTF P000002 is specifically named in the GROUP list, it will be applied even though it may already have been applied. Any requisite PTFs (PRE or REQ) will be applied only if they have not already been applied.

ACCEPT

The SMP accept function puts SYSMODs into the system's distribution libraries. The system distribution libraries are used to generate the operating system using a system generation (SYSGEN) process outside the scope of SMP. Once the initial operating SYSTEM has been generated, the accept process is generally performed for SYSMODs which have been applied and tested on the target operating system.

DELETION OF SYSMODS FROM THE PTS AT ACCEPT

The PTS data set SYSTEM entry PURGE indicator is used to control whether SYSMODs are deleted from the PTS after they are successfully accepted.

When "set", SYSMODs will be deleted from the PTS and temporary libraries loaded for these SYSMODs will be scratched when the SYSMODs are successfully applied and accepted.

When this indicator is not "set", the SYSMODs will remain on the PTS. Such SYSMODs left on the PTS may be applied and accepted to this same or other target and distribution libraries.

- The PURGE indicator may be "set" as follows,

```
UCLIN PTS .
ADD SYS PURGE /* DELETE SYSMOD FROM PTS AT ACCEPT */ .
ENDUCL .
```

- The PURGE indicator may be "reset" as follows,

```
UCLIN PTS .
DEL SYS PURGE /* DO NOT DELETE SYSMOD FROM PTS AT ACCEPT */ .
ENDUCL .
```

When the PURGE indicator is "reset" such that SYSMODs are not deleted from the PTS at ACCEPT, the user controls the deletion of PTS SYSMODs using either UCLIN or REJECT PURGE.

ACCEPT ALL SYSMODS WHICH HAVE BEEN SUCCESSFULLY APPLIED

```
ACCEPT .
```

This variation of ACCEPT is used to accept only those SYSMODs which have been successfully applied (found on the CDS). SYSMODs on the PTS which have not been applied will not be accepted (unless they are found to be superseded by another SYSMOD which is being accepted.)

ACCEPT SYSMODS IN PREPARATION FOR A SYSGEN

Generally, ACCEPT processes SYSMODs which have applied; however, in the situation where a system is first being built by a SYSGEN process, the SYSMODs which make up the system must be moved to the distribution libraries. In order to accept all SYSMODs, whether they have been applied or not, the NOAPPLY keyword is used as shown,

```
ACCEPT NOAPPLY .
```

All SYSMODs which have not been ACCEPTED will be accepted into the distribution libraries. The use of the NOAPPLY will cause the SYSMODs to be retained on the PTS after they have been accepted; The REJECT PURGE function may be used to clean up the PTS.

In order to accept specific SYSMODs which may or may not have been applied, the SELECT keyword is used as shown (NOAPPLY is not necessary),

```
      ACCEPT SELECT(F000002) .
```

In this illustration, the elements from the specified SYSMOD will be put in the distribution libraries. If NOAPPLY is not coded, an SMPCDS dataset will be required; if NOAPPLY is coded, no SMPCDS is required and the SYSMOD will be retained on the PTS after it is accepted; the REJECT PURGE function may be used to cleanup the PTS.

PTS LISTINGS FOR SYSMOD STATUS

PTS SYSMOD entry listings provide not only the "receive" status of SYSMODs but also contain information relative to the "apply" and "accept" status of a SYSMOD.

```
     LIST PTS SYSMOD .
```

will produce a listing of all SYSMODs on the PTS.

Each entry listed will have a "STATUS" field which will show "APP" if the SYSMOD has been applied to one or more operating systems and will show "ACC" if the SYSMOD has been accepted on one or more DLIBs.

The "APPLY CDSID" field of each entry will show the CDSID of each CDS to which the SYSMOD has been applied and the "ACCEPT ACDSID" field will show the CDSID of each ACDS to which the SYSMOD has been accepted.

These fields can be used to determine which SYSMODs have been applied and accepted and to which systems they have been so processed.

The NOAPPLY and NOACCEPT PTS listings can be used to determine which SYSMODs have not been applied or accepted to a particular system or distribution library. The NOAPPLY and NOACCEPT listings do not use the APP and ACC indicators in the PTS SYSMOD entries; rather, the listing is produced by comparing the PTS SYSMODs with the SYSMODs found on the CDS or ACDS when the listing is produced. For example,

```
     LIST PTS PTF NOAPPLY .
```

will produce a listing of all PTF -type SYSMODs which have not been applied to the CDS operating system.

CDS/ACDS LISTINGS FOR SYSMOD STATUS

The listings of the CDS and ACDS SYSMOD entries are very valuable in determining the status of the modifications which have been made to your system.

LISTING ONLY SYSMOD ENTRIES

```
LIST CDS SYSMOD .
```

will list all CDS SYSMOD entries.

LISTING SPECIFIC SYSMODS

```
LIST CDS SYSMOD(UZ00001,UZ00002) .
```

will list the CDS SYSMOD entries for only UZ00001 and UZ00002.

LISTING SPECIFIC SYSMOD TYPES

```
LIST CDS SYSMOD USERMOD .
```

will list the CDS SYSMOD entries whose type is USERMOD. The other types which may be specified are FUNCTION, PTF and APAR.

LISTING SPECIFIC SYSMOD STATUS

- To determine whether any SYSMODs have been _partially_ applied,

  ```
  LIST CDS SYSMOD ERROR .
  ```

- To determine whether any SYSMODs have been _partially_ restored,

  ```
  LIST CDS SYSMOD RESTORE .
  ```

- To list all SYSMODs on the CDS which have not been accepted,

  ```
  LIST CDS SYSMOD NOACCEPT .
  ```

- To list all SYSMODs on the ACDS which have not been applied,

  ```
  LIST ACDS SYSMOD NOAPPLY .
  ```

LISTINGS FOR ELEMENT INFORMATION

The listings of the CDS and ACDS ELEMENT entries provide both processing information and status (function-level and maintenance-level) of all elements on your system.

LISTING ONLY ELEMENT ENTRIES

```
     LIST CDS MOD .
```

will list all the CDS MOD entries. Other variations of element types may be specified singularly or together:

```
     LIST CDS MAC SRC .
```

will list all MACRO and SOURCE entries.

ELEMENT MODIFICATION HISTORY LISTINGS

The cross-reference (XREF) option of LIST will provide a useful historical record of modifications made to the elements on your system.

```
     LIST CDS MAC(MACRO01,MACRO02) XREF .
```

will list the CDS MACRO entries for the two macros specified and provide a list of all SYSMODs which have modified these macros.

The XREF option is useful for the other CDS element-type entries such as MOD (for modules) and SRC (for source).

MACRO PROCESSING INFORMATION

The CDS MAC entry provides the data required to process a macro modification. This entry shows the operating system macro library in which the macro resides (SYSLIB); when no SYSLIB is present, the macro is presumed to have no operating system library, and the SMPMTS data set will be used to to maintain the modified macro until it is accepted into its distribution library (DLIB).

In addition to the SYSLIB data the MAC entry may contain a list of source elements to be assembled whenever the macro is modified. These source elements are listed as GENASMs; SMP will access the CDS for this source as an ASSEM entry (or SRC entry if no ASSEM entry is found). These GENASM sources will be assembled and link edited when the macro is updated providing an associated CDS MOD entry (one having the same name as the GENASM) with load module data is available (see "Module Link Edit Processing" below).

SOURCE PROCESSING INFORMATION

The CDS SRC entry and a corresponding MOD entry provide the data required to process a source modification. The SRC entry itself shows the operating system library in which the source resides (SYSLIB); when no SYSLIB is present, the source is presumed to have no operating system library, and the SMPSTS data set will be used to to maintain the modified source until it is accepted into its distribution library (DLIB).

Associated with each SRC entry, there is generally a CDS MOD entry which SMP will use to determine how to process the assembled source modification (see "Module Link Edit Processing" below). If there is no corresponding MOD entry (that is, a MOD entry with the same name as the source element), SMP presumes that the source need not be assembled or link edited to an operating system library. A JCLIN job step shcwing the link edit of the source module will properly initialize the CDS to enable SMP to assemble and link edit the source.

MODULE LINK EDIT PROCESSING INFORMATION

In order to determine how a module will be link edited into your operating system. both the MOD entry and any associated LMOD entries must be examined.

```
     LIST CDS MOD(module) .
```

will display the CDS MOD entry. In this entry, the "LMODS" field identifies the load modules which contain this module. In order to determine how the module will be link edited, the CDS load module entries found in the MOD entry must be listed.

```
     LIST CDS LMOD(lmodxx) .
```

REJECT FACILITIES

The SMP Reject function removes SYSMODs from the PTS which are no longer desired by the user. Generally, SYSMODs are removed when they have been successfully accepted (see "Deletion of SYSMODs from the PTS at ACCEPT" on page 94). This method of operation is not always satisfactory, especially when multiple systems are being maintained from the same PTS data set.

Two forms of "mass" REJECT are available:

1. The first form removes SYSMODs which have been <u>neither</u> applied nor accepted:

   ```
           REJECT .
   ```

   This form of REJECT determines whether a SYSMOD has been applied or accepted from the APPID and ACCID sub-entries in the PTS SYSMOD entry.

2. The second form removes SYSMODs which been accepted into the ACDS present when reject is invoked:

   ```
           REJECT PURGE .
   ```

   In contrast with the first form of REJECT, this form determines whether a SYSMOD has been accepted by examining the ACDS SYSMOD entries; the presence of an ACDS SYSMOD entry (not "in-error") indicates that the SYSMOD has been accepted.

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, two user exits can be defined in HMASMUXD. They are described as "User Exit 1" and "User Exit 2" below.

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	Offset (DEC)	Len	Description
UXPUXNUM	+0	2	User exit number (hexadecimal)
	+2	2	Unused
UXPUXNAM	+4	8	User exit name
UXPUXAD	+12	4	Address of user exit
UXPFUNCT	+16	8	SMP function name
UXPPRMAD	+24	4	Address of user exit parameter list
UXPLOJAD	+28	4	Address of work area common to user exits
UXPLOEAD	+32	4	Address of work area for this exit
UXPCTBAD	+36	4	Reserved
UXPMODAD	+40	4	Reserved

Figure 6 - HMASMUXP - Parameter List to HMASMUXD

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 entry point address of the exit routine can be placed in field UXPUXAD or the name of the exit module can be placed in UXPUXNAM. If the address is passed back, it is used as the entry point for the user exit. If the name is passed back, the exit module will be loaded if UXPUXAD is zero. 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 without an entry point in UXPUXAD, the user exit to be loaded module must exist as a load module in LINKLIB or in a data set defined by the STEPLIB or JOBLIB DD statements. The user exit must exist in an authorized library.

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 execution, it will be loaded during SMP initialization and deleted during SMP termination. If the user exit applies to a specific SMP function, it will be loaded during initialization for 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 SMP 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.

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 function level exits (RECEIVE and APPLY/ACCEPT/RESTORE) are supported and they cannot be activated at the same time. 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 (RECEIVE)

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 2 for information on when this exit is called.

When the exit is called, the fields in the parameter list have the following values:

```
UXPUXNUM - X'0001'     - user exit number (hex)

UXPFUNCT - 'RECEIVE ' - the RECEIVE function (character string)

UXPPRMAD - address of 81 byte buffer area (see below)
```

Figure 7 - Values Passed in HMASMUXP - User Exit 1

FIELD NAME	Offset (DEC)	Length	DESCRIPTION
none	+0	1	X'00' - PTFINBUF contains record to be processed. X'04' - End-of-file on SMPPTFIN.
PTFINBUF	+1	80	PTFIN input record next to be processed.

Figure 8 - Buffer Passed by UXPPRMAD - User Exit 1

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 rein-yoked 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 buff-er, the exit clears the buffer area and returns a 0 in register 15.

- 24 - Delete record in buffer area

USER EXIT 2 (RETRY)

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 dataset followed by a reinvocation of the failing UTILITY. Note that the dump for the failure has been cancelled before the user exit is called.)

When the exit is called, the fields in its input parameter list have the following values:

```
UXPUXNUM - X'0002' - user exit number (hex)

           'APPLY   '   SMP function being
UXPFUNCT - 'ACCEPT  '   performed (character
           'RESTORE '   string)

UXPPRMAD - address of 25 byte parameter list
           (see below)
```

Figure 9 - Values Passed in HMASMUXP - User Exit 2

FIELD NAME	Offset (DEC)	Length	DESCRIPTION
UX002DDN	+0	8	Target DDNAME on which the 837-04, D37-04 or E37-04 occurred.
UX002PGM	+8	8	The program name of the utility invoked which caused the failure.
UX002ACH	+16	3	Abend code encountered (hex) (same format as SWDA field SDWACMPC)
UX002RCH	+19	1	Abend reason (hex)
UX002ACP	+20	3	Abend code (printable EBCDIC)
UX002RCP	+23	2	Abend reason code (printable EBCDIC)

Figure 10 - Parameter List Passed by UXPPRMAD - User Exit 2

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.

SMP DATA SET ALLOCATION GUIDELINES

This section provides some guidance for the initial allocation of the SMP data sets required for MVS and VS1 system installation and maintenance.

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

See Chapter 7 for detailed descriptions of the use and purpose of each data set.

Primary Data Set Requirements

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 11 and 12 provide guidelines for estimating the storage requirements of the primary SMP data sets. Figure 11 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 12 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				
DATA SET NAMES	2305	2314/ 2319	3330/ 3333	3340	3350
SMPACDS	41 x 62 y	409 x 614 y	225 x 338 y	475 x 551 y	155 x 231 y
SMPACRQ	11 x 16 y	109 x 164 y	60 x 90 y	98 x 147 y	41 x 62 y
SMPCDS	55 x 82 y	545 x 818 y	300 x 450 y	489 x 734 y	206 x 309 y
SMPCRQ	11 x 16 y	109 x 164 y	60 x 90 y	98 x 147 y	41 x 62 y
SMPLOG	70	140	76	126	52
SMPMTS	52	104	57	94	39
SMPPTS	580 x 870 y	153 x 730 y	634 x 950 y	040 x 560 y	433 x 650 y
SMPSCDS	5 x 8 y	55 x 82 y	30 x 45 y	49 x 73 y	21 x 31 y
SMPSTS	52	104	57	94	39
LINKLIB	46	93	51	91	35

x	VS1 Systems
y	VS2 Systems

Figure 11: - 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 12: - 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	• SMPWRK1	• SYSUT1
• SMPJCLIN	• SMPWRK2	• SYSUT2
• SMPLIST	• SMPWRK3	• SYSUT3
• SMPOUT	• SMPWRK4	• SYSUT4
• SMPPTFIN	• SMPWRK5	• distribution libs.
• SMPRPT	• SYSLIB	• LKLIB libraries
• SMPTLIB	• SYSPRINT	• target libraries
• SMPADDIN	• SMPPUNCH	• TXLIB libraries

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:

```
         530K   (Size of SMP program)
+          8K   (SMP fixed-size storage areas)
+         20K   (reserved for system use ... open,close)
+          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
+         36 x largest PEMAX value
+        252 x number of directory blocks in ACDS
+        252 x number of directory blocks in CDS
+          1 x largest size of programs invoked by SMP
+          1 x calculated storage for processing SYSMODs
  (see calculations below)
---------
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	x	number of SYSMODs being processed
+	160	x	number of elements in SYSMODs being processed
+	160	x	number of ASSEM and SRC entries referenced by macros modified by SYSMODs being processed
+	160	x	number of ASSEM and SRC entries that are to be assembled
+	64	x	number of unique load modules that are to be link edited or modified by IMASPZAP
+	8	x	number of DELETE, NPRE, PRE, REQ, SUP, and VERSION operands in the ++VER modification control statements in SYSMODs being processed
+	16	x	number of REQ operands in ACRQ/CRQ and ++IF modification control statements in SYSMODs being processed that are applicable to your environment ... APPLY/ACCEPT only.
+	8	x	number of sysmods which are applied (on CDS) but not accepted (on ACDS) ... RESTORE only.
+	16	x	(number of elements being RESTOREd)
		x	(number of SYSMODs which have supplied each element) ... RESTORE only.
+	22	x	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.

The maximum amount of storage that SMP attempts to obtain for internal entries is 4000k. If your calculations show that the processing of a set of SYSMODs might exceed this amount, you should process smaller subsets of that set.

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' on page &dirst. in this chapter for additional information.

CHAPTER 4: SMP REPORTS

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 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 data-set. 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 data-set.

   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 SMPTLIB 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.

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".

```
                         RECEIVE SUMMARY REPORT

SYSMOD    STATUS        TYPE        — — — — — — — — — — — —

EXY1101   RECEIVED      FUNCTION
UZ00001   RECEIVED      PTF
UZ00002   NOT RECEIVED  PTF         SYNTAX/CONSTRUCTION
UZ00003   RECEIVED      PTF
UZ00004   NOT RECEIVED  PTF         USER EXIT
```

Figure 13 - RECEIVE SUMMARY Report

APPLY, RESTORE AND ACCEPT REPORTS

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 eligible SYSMOD. The SYSMODs are listed in alphanumeric sequence (see Figure 14).

The fields in the report are as follows:

- SYSMCD - The identifier of the system modification

- STATUS - Describes what has happened to the SYSMOD. The possible values of this field are as follows:

  - APPLIED, ACCEPTED, or RESTORED - The SYSMOD was successfully processed.

  - SUPD - The SYSMOD is superseded by one or more SYSMODs being processed. The superseding SYSMODs are shown in the "REQUISITE AND SUPEDBY SYSMODS" field.

  - 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.

  - ERROR - The SYSMOD was terminated after some target system or SMP libraries were updated but before the SYSMOD could be considered completely processed. A SYSMOD is considered "completely" processed when all of its elements have been processed and all of the SYSMOD's requisites have been "completely" processed. The message output should be checked to determine the cause of the error. This condition does not appear when the CHECK operand is specified.

  - DELETED - The SYSMOD was explicitly or implicitly deleted.

  - INCMPLT - The processing for this SYSMOD is "incomplete" due to some failure. The SYSMOD has not updated any target system libraries (as distinguished from the ERROR status described above).

- TYPE - The system modification type (APAR, FUNCTION, PTF, or USERMOD).

- FMID - The SYSMOD's FMID. This field is not filled in for superseded SYSMODs.

- REQUISITE AND SUPEDBY SYSMODS - shows the requisite and superseding SYSMODs. The lists are preceded by the type of requisite as follows:

  - 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.

  - PRE - The SYSMODs are prerequisites of the SYSMOD.

  - REQ - The SYSMODs are requisites of the SYSMOD.

  - SUPBY - The SYSMODs that supersede the SYSMOD.

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 can be processed.

```
     DATE 80.092 TIME 09=25 47/HMASMP LVL 04.18 SMPRPT OUTPUT

SYSMOD STATUS REPORT FOR APPLY PROCESSING

  NOTE:   '-' INDICATES THE REQUISITE SYSMOD CONDITION IS NOT SATISFIED
          '*' INDICATES THE NON SATISFIED REQUISITE SYSMOD CONDITION IS
              BYPASSED

SYSMOD     STATUS    TYPE        FMID       REQUISITE AND SUPBY SYSMODS
----------------------------------------------------------------------

AZ00124    APPLIED   APAR        GXY1000    PRE    UZ00010
GXY1000    APPLIED   FUNCTION    GXY1000
HXY1010    APPLIED   FUNCTION    GXY1000    PRE    UZ00010
UZ00010    APPLIED   PTF         GXY1000
UZ00001    SUPD                             SUPBY  UZ00015
0200012    APPLIED   PTF         GXY1000    PRE    UZ00010
U200014    APPLIED   PTF         GXY1000    IFREQ  UZ00015
                                            PRE    UZ00010
UZ00015    APPLIED   PTF         HXY1010    REQ    UZ00014
XY10001    APPLIED   USERMOD     GXY1000    IFREQ  XY10101
XY10101    APPLIED   USERMOD     HXY1010
```

Figure 14 - 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 15). 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 ERR - 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 - This version of the element was not selected. If multiple versions of the same element are being processed concurrently, a superior version may have been chosen from another SYSMOD. If none of the versions of an element are selected, a superior version existed previously on the target system.

     Often, an element is not selected because its FMID did not match the FMID of the element on the target system. The selection and exclusion of elements is discussed in the APPLY Processing section of Chapter 2.

  8. SRC SEL - The object module (++MOD) was not processed. This situation occurs when the object is created by assembling the source for the module as a result of a source or macro modification.

- 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.

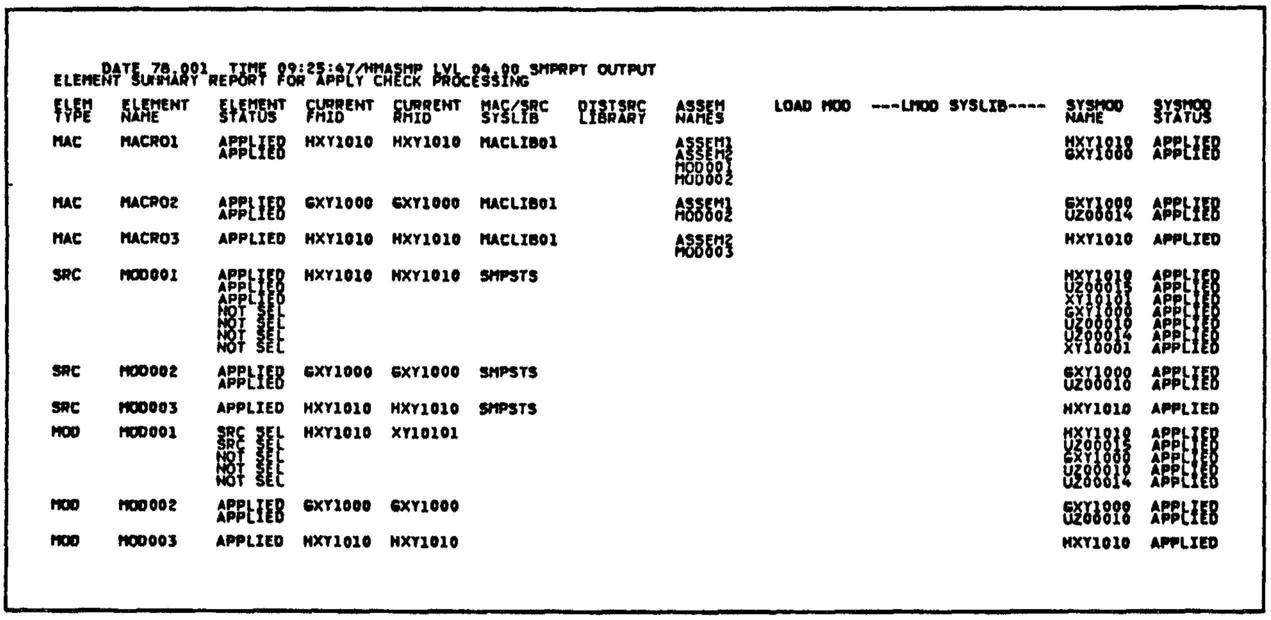
- LOAD MOD - A list of load modules that were link edited and/or copied using the module named in the ELEMENT NAME field. This field is not present for ACCEPT processing.

- 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    ELEMENT   ELEMENT   CURRENT   CURRENT   MAC/SRC   DISTSRC   ASSEM       LOAD MOD  ---LMOD SYSLIB----  SYSMOD    SYSMOD
TYPE    NAME      STATUS    FMID      RMID      SYSLIB    LIBRARY   NAMES                                     NAME      STATUS

MAC     MACRO1    APPLIED   HXY1010   HXY1010   MACLIB01            ASSEM1                                    HXY1010   APPLIED
                  APPLIED                                           ASSEM2                                    GXY1000   APPLIED
                                                                    MOD001
                                                                    MOD002

MAC     MACRO2    APPLIED   GXY1000   GXY1000   MACLIB01            ASSEM1                                    GXY1000   APPLIED
                  APPLIED                                           MOD002                                    UZ00014   APPLIED

MAC     MACRO3    APPLIED   HXY1010   HXY1010   MACLIB01            ASSEM2                                    HXY1010   APPLIED
                                                                    MOD003

SRC     MOD001    APPLIED   HXY1010   HXY1010   SMPSTS                                                        HXY1010   APPLIED
                  APPLIED                                                                                     UZ00015   APPLIED
                  APPLIED                                                                                     XY10101   APPLIED
                  NOT SEL                                                                                     GXY1000   APPLIED
                  NOT SEL                                                                                     UZ00010   APPLIED
                  NOT SEL                                                                                     UZ00014   APPLIED
                  NOT SEL                                                                                     XY10001   APPLIED

SRC     MOD002    APPLIED   GXY1000   GXY1000   SMPSTS                                                        GXY1000   APPLIED
                  APPLIED                                                                                     UZ00010   APPLIED

SRC     MOD003    APPLIED   HXY1010   HXY1010   SMPSTS                                                        HXY1010   APPLIED

MOD     MOD001    SRC SEL   HXY1010   XY10101                                                                 HXY1010   APPLIED
                  SRC SEL                                                                                     UZ00015   APPLIED
                  NOT SEL                                                                                     GXY1000   APPLIED
                  NOT SEL                                                                                     UZ00010   APPLIED
                  NOT SEL                                                                                     UZ00014   APPLIED

MOD     MOD002    APPLIED   GXY1000   GXY1000                                                                 GXY1000   APPLIED
                  APPLIED                                                                                     UZ00010   APPLIED

MOD     MOD003    APPLIED   HXY1010   HXY1010                                                                 HXY1010   APPLIED
```

Figure 15 - The ELEMENT SUMMARY Report

The SYSMOD REGRESSION Report

The SYSMOD regression report summarizes the MODID CHECK conditions described under "FMIDs Match - MODID Verification" on page 28 with respect to elements in the target system. A regression situation occurs when SMP APPLY or ACCEPT processes *an* element from a SYSMOD which did not express a proper PRE/SUP relationship with the RMID and/or UMID attributes of the corresponding element in the target system. This situation can only occur when the MODID CHECK ERROR termination is bypassed using BYPASS(ID). This situation will not occur when a new function is being installed since elements from a FUNCTION are selected based upon functional "superiority" (see "FMIDs Differ on page 28); SMP assumes that service (PTFs and USERMODs) installed on the functionally inferior elements has provided ++IF conditional requisite data to bring the functionally superior SYSMOD up to the proper service level.

If no regressions are detected, this report is generally not produced. There are certain circumstances, however, in which the MODID verification detects a regression situation which is resolved by other SYSMODs being processed during the same APPLY or ACCEPT; in this case, the regression report will be produced and consist 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 modified the element(s) listed in the COMMON ELEMENTS fields. These are the SYSMODs whose modifications were potentially overlaid.

- 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.

```
     DATE 80.092 TIME 09:25:47/HMASMP LVL 04.18 SMPRPT OUTPUT
SYSMOD REGRESSION REPORT FOR APPLY CHECK PROCESSING

REGRESSING     REGRESSED     COMMON ELEMENTS     OTHER POTENTIALLY
SYSMOD         SYSMOD        TYPE    NAME        REGRESSED SYSMODS
_______        _________     ______  ________    ________________________

UZ00099        UZ00001       MODULE  HMAB0123    AZ00050 AZ00051 AZ00052

               UZ00002       MACRO   HMAMAC01    AZ00055
                             MODULE  HMAB0456

               UZ00003       MODULE  HMAB0789    AZ00056 AZ00057
                             MODULE  HMAB0012
                             MODULE  HMAB0345

UZ00111        UZ00004       MODULE  HMAB0678    AZ00058 AZ00059 AZ00060
                             MODULE  HMAB0987
                             MODULE  HMAB0124
                             MODULE  HMAB0135
```

Figure 16 - 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 17). It is not produced for RESTORE processing or when no DELETE processing has occurred.

The fields in the report are as follows:

- DELETING SYSMOD - The identifier of the SYSMOD containing the DELETE operand in its ++VER modification control statement.
- DELETING TYPE and SYSMOD - 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:
  - 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.
  - 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.
  - 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.
  - 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 80.092 TIME 09:25:47/HMASMP LVL 04.18 SMPRPT OUTPUT
DELETED FUNCTION REPORT FOR APPLY CHECK PROCESSING

DELETING       DELETED       SYSMODS
SYSMOD         TYPE
               --------      ------------------------------------

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 17 - The DELETED FUNCTION Report

CHAPTER 5: 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
- DEBUG - enables SMP debug facilities
- 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
- UCLIN - used in conjunction with the UCL and ENDUCL statements to update SMP data sets.
- UCL - update control language statements used to describe update processing to be done by the UCLIN function.
- 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 2 of this manual.

The control statement syntax is arranged in this chapter as follows:

The SMP ACCEPT control statement invokes ACCEPT processing which installs 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)]
    [REUSE]
    .
```

ACCEPT OPERANDS

SELECT(sysmodid[,sysmodid]...)
: specifies one or more SYSMODs to be placed into the DLIBs or permanent user libraries. A SYSMOD which is specifically selected will be ACCEPTED even though it may not be APPLIED. A SYSMOD which is already ACCEPTED will be reprocessed if it is specifically selected. 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. A SYSMOD which is specifically selected will be ACCEPTED even though it may not be APPLIED. A SYSMOD which is already ACCEPTED will be re-processed if it is specifically selected. Requisite and prerequisite SYSMODs (which are not already ACCEPTED) are automatically included in the processing (including any requisites and prerequisites of the requisite and prerequisite SYSMODs). This operand can also be specified as 'G'.

Note: When neither SELECT, GROUP nor NOAPPLY keywords are coded, only those SYSMODs that have been APPLIED will be considered for ACCEPT processing.

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'.

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 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'.

- If the CHECK and COMPRESS operands are both specified, the COMPRESS operand is ignored and no compression is performed.
- The SMPACDS and SMPCDS data sets cannot be compressed. If specified, they are ignored.

DIS(READ | NO | WRITE)
specifies if 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 DISCREAD), the default value, is used. The directory entries are not updated in CHECK mode.

NOAPPLY
specifies that all SYSMODs found on the PTS should be considered even though they have not been APPLIED.

Note: If the NOAPPLY operand is specified, then the CDS data set is not required during ACCEPT processing. However, if the SMPCDS DD is present, it must be a valid CDS.

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.

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.

RETRY(YES | NO)
where 'YES' indicates that SMP is to attempt a RETRY for each X37 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.

REUSE
specifies that when an object module is found on SMPWRK3 from a previous successful SMP generated assembly for that module, the new assembly will be suppressed and the existing object from SMPWRK3 will be used instead of the one which would have been produced by the new assembly.

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

- CAUTION: SMP cannot remove a SYSMOD from the target system after ACCEPT processing.

- 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.

- A data set can be specified in the COMPRESS operand list even if is not affected by any modification in the same ACCEPT pass.

  COMPRESS does not process keyed or unmovable data sets.

  The COMPRESS function should not be performed on a running operating system; an alternate system should be used to apply the service or function.

- 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 require the macro for assemblies.

- 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.

- When modules are link edited into the distribution libraries, external references might be unresolved; therefore, ignore message IEW0461.

- 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).

- 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.

- 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.

- Use the DIS(NO) option only when the number of SYSMODs and their elements is small or when the trade off between storage utilization and performance has to be made in favor of storage.

- 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.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

The SMP APPLY control statement invokes APPLY processing which installs SYSMODs into the operating 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)]
    [REUSE]
    .
```

APPLY OPERANDS

SELECT(sysmodid[,sysmodid]...)
specifies one or more SYSMODs to be placed into the target system libraries. If a SYSMOD which is already APPLIED is specifically selected, it will be reprocessed. This operand can also be specified as 'S'.

GROUP(sysmodid[,sysmodid]...)
specifies one or more SYSMODs to be placed into the target system libraries. Any requisite and prerequisite SYSMODs (which are not already APPLIED) are automatically included in the processing (including any requisites and prerequisites of the requisite and prerequisite SYSMODs). This operand can also be specified as 'G'.

Note: If neither SELECT nor GROUP is coded, all SYSMODs that have not been successfully APPLIED are considered for processing.

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'.

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 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.

Note: 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 if 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.

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.

RETRY(YES | NO)
where 'YES' indicates that SMP is to attempt a RETRY for each X37 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 net specified and a RETRYDDN is available.

REUSE
specifies that when an object module is found on SMPWRK3 from a previous successful SMP generated assembly for that module, the new assembly will be suppressed and the existing object from SMPWRK3 will be used instead of the one which would have been produced by the new assembly.

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 only)
txlib	(one for each different TXLIB 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 the 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 trade off 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.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

ENDUCL CONTROL STATEMENT

The SMP ENDUCL control statement identifies the end of the update control language (UCL) statements and signifies the and 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" in Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047-0) manual.

The SMP DEBUG control statement enables SMP debug facilities to assist in the debugging of SMP problems.

DEBUG SYNTAX

```
DEBUG MSGMODID(ON | OFF) .
```

DEBUG OPERANDS

MSGMODID(ON)
: enables the message debug facility.

MSGMODID(OFF)
: disables the message debug facility.

DEBUG DDNAMES

None

DEBUG PROGRAMMING CONSIDERATIONS

When the MSGMODID debug facility is enabled, SMP messages are prefixed by the character string

```
@yyy+X'zzzzzz'
```

where

yyy is the last three characters of the SMP module name which issued the message

zzzzzz is the offset (in hexadecimal) into the module where the message was issued.

DEBUG RETURN CODES

None

The SMP JCLIN Control statement invokes JCLIN processing for the JCLIN data supplied in the data set described by the SMPJCLIN DD statement. JCLIN processing initializes entries in the CDS for subsequent APPLY processing. Any number of JCLIN statements can be included in an SMP job step.

JCLIN SYNTAX

```
JCLIN [ASM({PGM=asmpgm | asmproc})]
   [COPY({PGM=copypgm | copyproc})]
   [DIS( NO | READ | WRITE )
   [LKED({PGM=lkedpgm | lkedproc})]
   [UPDATE({PGM=updpgm | updproc})]
   [RC(function=code[,function=code]...}]
   .
```

JCLIN OPERANDS

ASM({PGM=asmpgm | asmproc}}
: specifies an assembler program, asmpgm, or procedure, asmproc, in addition to those recognized by SMP. SMP recognizes programs ASMBLR, IFOX00, IEUASM and procedure ASMS.

COPY({PGM=copypgm | copyproc})
: specifies a copy program, copypgm, or procedure, copyproc, in addition to those recognized by SMP. SMP recognizes only program, IEBCOPY.

DIS(NO | READ | WRITE)
: specifies if 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 specifies 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=lkedpgm | lkedproc})
: specifies a linkage editor program, lkedpgm, or procedure, lkedproc, in addition to those recognized by SMP. SMP recognizes programs IEWL and HEWL and procedure LINKS.

UPDATE({PGM=updpgm | updproc})
: specifies an update program, updpgm, or procedure, updproc, in addition to those recognized by SMP. SMP recognizes only program IEBUPDTE.

Note: Although JCLIN does not actually derive any CDS initialization data from update job steps, update job steps are recognized so that the update-step SYSIN data (which may itself contain JCL) is not processed.

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

- 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. The job step coding conventions are described in "JCLIN Processing" section of Chapter 2.
- Any step using a PGM or Procname other than those recognized is ignored and not considered as an error. The return code it not affected.

- 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.

- 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.

- Specification of DIS(NO) or DIS(READ) when processing JCLIN might cause the processing time to increase significantly.

- 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.

- JCLIN from a superseded SYSMOD is not processed when both the superseded SYSMOD and the superseding SYSMOD are being processed in the same APPLY step.

- The linkage editor control statement IDENTIFY should not be used as input for JCLIN.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

The LIST control statement invokes SMP LIST processes which provide listings of:

- All data or selected data from the ACDS, ACRQ, CDS, CRQ, PTS and SCDS data sets.
- The contents of the LOG data set.

These 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 (GENERAL)

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 dataset [entry[,entry]...] [option[,option]...] .
```

LIST Operands (General)

data set
: specifies the SMP data set to be listed. The allowable datasets are ACDS, CDS, ACRQ, CRQ, LOG, PTS and SCDS.

  There is no default data set.

entry
: specifies the entry or entry types to be listed. The valid entry types are described below for each SMP data set.

option
: specifies options which control the data listed. The syntax and explanations of the options are described below for each SMP data set.

If no entries or options are specified, a complete listing of all entries in the data set is produced with the exception of XREF information. ACDS and CDS XREF information must be explicitly requested.

The LIST syntax descriptions in this chapter are arranged by data set as follows:

DATA SET	Page
ACDS	146
ACRQ	157
CDS	146
CRQ	157
LOG	160
PTS	161
SCDS	167

LIST Output (General)

The listing output goes to the file defined by the SMPLIST DD statement (or to SMPOUT if SMPLIST is not present).

The fields that appear in these listings (with the exception of the ACDS/CDS XREF fields) are described in Chapter 8, "Control Data Set Entries".

Indicators in SYSTEM entries are shown as "YES" if the indicator is set and "NO" if it is not set.

Indicators in the SYSMOD entries that are defined as status indicators appear in the listing with their associated abbreviation.

LIST DDnames (General)

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 (General)

- 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.

  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.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

LIST ACDS/CDS SYNTAX

```
   LIST { ACDS | CDS }

        [ASSEM[(asmname[,asmname]...)]]

        [DLIB[(dlibname[,dlibname]...)]]

        [FORFMID(sysmodid)]

        [LMOD[(modname[,modname]...)]]

        [MAC[(macname[,macname]...)]]

        [MOD[(modname[,modname]...)]]

        [SRC[(srcname[,srcname]...)]]

        [SYSMOD[(sysmodid[,sysmodid]...)]
           [APAR] [DELETE] [ERROR] [FUNCTION]
           [NOACCEPT|NOAPPLY] [NOSUP] [PTF] [SUP]
           [RESTORE] [USERMOD]]

        [SYS]

        [XREF]

        .
```

LIST ACDS/CDS Operands

ASSEM[(asmname[,asmname]...)]
specifies that information for all ASSEM entries or the specified ASSEM entries is to be listed.

ASSEM entries exist only on the CDS.

DLIB[(dlibname[,dlibname]...)]
specifies that information for all DLIB entries or the specified DLIB entries are to be listed.

DLIB entries exist only on the CDS.

FORFMID(sysmodid)
specifies only those entries who's FMID subentry field match the value specified, will be listed.

LMOD[(modname[,modname]...)]
specifies that information for all LMOD entries or the specified LMOD entries are to be listed.

LMOD entries exist only on the CDS.

MAC[(macname[,macname]...)]
specifies that information for all MAC entries or the specified MAC entries are to be listed.

MOD[(modname[,modname]...)]
specifies that information for all MOD entries or the specified MOD entries are to be listed.

SRC[(srcname[,srcname]...)]
specifies that information for all SRC entries or the specified SRC entries are to be listed.

SYSMOD[(sysmodid[,sysmodid]...)]
    [APAR] [DELETE] [ERROR] [FUNCTION]
    [NOACCEPT | NOAPPLY] [NOSUP] [PTF] [SUP]
    [RESTORE] [USERMOD]
specifies that information for all SYSMOD entries or the specified SYSMOD entries is to be listed.

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 or ACDS 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 applied but not accepted are to be listed. This option compares SYSMODs found on the CDS with the SYSMODs on the ACDS; both the CDS and the ACDS must be available. This operand can be abbreviated as 'NOACC'.

<u>NOACCEPT can be specified for CDS listings only.</u>

NOAPPLY
specifies that SYSMODs that have been accepted but not applied are to be listed. This option compares SYSMODs found on the ACDS with the SYSMODs on the CDS; both the CDS and the ACDS must be available. This operand can be abbreviated as 'NOACC'.

NOAPPLY can be specified for ACDS listings only.

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'.

RESTORE can be specified for CDS listings only.

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'.

XREF
generates cross reference information as part of the listing for each MAC, MOD, SRC, and SYSMOD entry (see "List ACDS/CDS XREF Fields" below).

You should be aware that SMP uses extra time and more storage to generate the additional data requested by the XREF keyword.

SYS
specifies that system entry is to be listed.

LIST ACDS/CDS Output

The figures on the following pages illustrate the output produced for various CDS listings. The fields that appear (with the exception of the XREF fields) are described in Chapter 8, "Control Data Set Entries".

Note: The date field in the CDS SYSMOD entries reflects the date the SYSMOD was applied. If the CDS was created by copying the ACDS to the CDS, the apply data will appear as zero and the SYSMOD status will be RGN.

LIST ACDS/CDS XREF Fields

The following fields appear for ACDS and CDS XREF listings:

- "MACROS USED" - appears for CDS ASSEM entries and indicates the CDS MAC entries with a GENASM sub-entry for the ASSEMbly.
- "MODULES" - appears for CDS LMOD entries *and* indicates the CDS MOD entries with a LMOD sub-entry for the MODule.
- "SYSMOD HISTORY" - appears for ACDS and CDS MAC, MOD and SRC entries. The "SYSMOD HISTORY" lists all other SYSMODs in the control data set which have modified the respective element and indicates the SYSMOD type and status.
- "NPREBY (XREF)" - appears for ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which specify the particular SYSMOD as *an* NPRE (negative prerequisite).
- "PREBY (XREF)" - appears for ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which specify the particular SYSMOD as an PRE (prerequisite).
- "REQBY (XREF)" - appears for ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which specify the particular SYSMOD as an REQ (requisite).
- "VERSIONBY (XREF)" - appears for ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which specify the particular SYSMOD in the VERSION keyword of their ++VER.
- "DELBY (XREF)" - appears for ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which specify the particular SYSMOD in the DELETE keyword of their ++VER.
- "IFREQBY (XREF)" - appears foes ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which specify the particular SYSMOD as an IFREQ (conditional requisite).
- "SUPBY (XREF)" - appears for ACDS/CDS SYSMOD entries and indicates all other SYSMODs in the control data set which SUP (supersede) the particular SYSMOD.

```
     DATE 80.092  TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT
SMPCDS        ASSEMBLER ENTRIES
  NAMES

ASSEM1        LAST UPDATE     = GXY1000 TYPE=ADD
              ASSEMBLER INPUT = .
                                .
                                .
              MACROS USED     = MACRO1  MACRO2

ASSEM2        LAST UPDATE     = HXY1010 TYPE=REP
              ASSEMBLER INPUT = .
                                .
                                .
              MACROS USED     = MACRO1  MACRO3
```

Figure 18 - LIST CDS ASSEM XREF

```
     DATE 80.092  TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCDS        DLIB ENTRIES
  NAMES

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 19 - LIST CDS DLIB

```
     DATE 80.092  TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCDS        LOAD MODULE ENTRIES
  NAMES

LMOD1         LAST UPDATE     = HXY1010 TYPE=REP
              SYSTEM LIBRARY  = SYSLIB01  SYSLIB02
              LKED ATTRIBUTES = NCAL
              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 MOD00#
              MODULES         = MOD003  ASSEM2
```

Figure 20 - LIST CDS LMOD XREF

```
     DATE 80.092  TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCDS     MACRO ENTRIES
  NAME

MACRO1     LAST UPDATE     = HXY1010 TYPE=REP
           LIBRARIES       = DISTLIB=DLIBMAC2  SYSLIB=MACLIB01
           FMID            = HXY1010
           RMID            = HXT1010
           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            = GXT1000
           UMID            = UZ00014
           GENASM          = ASSEM1  MOD002
           SYSMOD HISTORY  = SYSMOD   TYPE      DATE   MCS ---STATUS---
                             GXY1000  FUNCTION 77.301  MAC  APP    ACC
                             UZ00014  PTF      77.357  MAC  APP
```

Figure 21 - LIST CDS MAC XREF

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

 SMPCDS     MODULE ENTRIES
   NAME

 MOD001     LAST UPDATE     = UZ00010 TYPE=ADD
            LIBRARIES       = DISTLIB=DLIB01
            ASSEMBLE
            FMID            = HXY1010
            RMID            = XY10101 RMIDASM
            LMODS           = LMOD1
            SYSMOD HISTORY  = SYSMOD   TYPE       DATE   MCS ---STATUS---
                              GXY1000 FUNCTION 77.301   MOD    APP   ACC
                              AZ00124 APAR     77.318   SRCUPD APP
                              XY10101 USERMOD  77.318   SRCUPD APP

 MOD002     LAST UPDATE     = UCLIN TYPE=ADD
            LIBRARIES       = DISTLIB=DLIB01
            FMID            = GXY1000
            RMID            = UZ00010
            LMODS           = LMOD1
            SYSMOD HISTORY  = SYSMOD   TYPE       DATE   MCS ---STATUS---
                              GXY1000 FUNCTION 77.301   MOD    APP   ACC
                              UZ00010 PTF      77.312   MOD    APP
```

Figure 22 - LIST CDS MOD XREF

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

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---
                               HXY1010 FUNCTION 77.345  SRC    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
            MACRO USED       = MACRO1 MACRO2
            SYSMOD HISTORY   = SYSMOD   TYPE       DATE   MCS ---STATUS---
                               GXY1000 FUNCTION 77.301  SRC    APP   ACC
                               UZ00010 PTF      77.312  SRCUPD APP
```

Figure 23 - LIST CDS SRC XREF

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCDS     SYSMOD ENTRIES
  NAME

AZ00123    TYPE             = SUPERCEDED
           SUPBY            = HXY1010 UZ00010

AZ00124    TYPE             = APAR
           STATUS           = REC APP
           FMID             = GXY1000
           DATE/TIME REC    = 77.318 14:28:54
                     APP    = 77.318 14:30:54
           SREL  VER(001)   = Z038
           PRE   VER(001)   = UZ00010
           SRCUPD           = MOD001
           SUPBY(IN SYSMOD)= HXY1010 UZ00012

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 12:22:45
           JCLIN            = YES
           SREL VER(001)    = Z038
           MAC              = MACRO1 MACRO2
           MOD              = MOD001 MOD002
           SRC              = MOD001 MOD002

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)  = MOD001  MOD002

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
           IFREQBY  (XREF)  = XY10101
```

Figure 24 - LIST CDS SYSMOD XREF

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCDS     SYSTEM ENTRY
  NAME

SYSTEM OPTIONS = CDSID=CDS1 SREL=2038 NUCID=8 PEMAX=9999 SAVEMTS=YES
```

Figure 25 - LIST CDS SYS

LIST ACDS/CDS Exception Reports

There are two possible exception reports from ACDS/CDS list processing; the "LIST MASS SUMMARY REPORT" and the "LIST SELECT SUMMARY REPORT". The reports are produced at the end of your LIST output for the following exceptions:

- For listings of the entire CDS or ACDS:

  Those entry types expected to be in the data set but not found are reported. For example, if there are no MOD entries are found in the CDS, this situation will be reported as "MOD ENTRIES NOT FOUND" in the "LIST MASS SUMMARY REPORT FOR SMPCDS".

- For listings qualified by a particular entry type:

  If no entries of the particular type are found, the situation will be reported in the "LIST MASS SUMMARY REPORT". For example, if there are no MOD entries are found in the CDS when "LIST CDS MOD" is specified, "MOD ENTRIES NOT FOUND" will be reported in the "LIST MASS SUMMARY REPORT FOR SMPCDS".

- For listings of a specific entry type an· entry name:

  If the specified entry type and name are not found, the situation will be reported in the "LIST SELECT SUMMARY REPORT". For example, if there is not MOD entry for "MOD0001" and "LIST CDS MOD(MOD0001)" is specified, "MOD MOD0001" is reported in the "LIST SELECT SUMMARY REPORT" as not found.

- For SYSMOD listings qualified by SYSMOD status:

  SYSMOD entry listings may be qualified by one of the following status qualifiers: APAR, DELETE, ERROR, FUNCTION, NOACCEPT, NOAPPLY, NOSUP, PTF, SUP, RESTORE and USERMOD.

  If no SYSMOD entries are found for the specified status, this situation will be reported as "SYSMOD ENTRIES NOT FOUND" in the "LIST MASS SUMMARY REPORT".

- <u>For specific SYSMOD listings qualified by SYSMOD status or type:</u>

  If you attempt to list specific CDS SYSMOD entries qualified by APAR, DELETE, ERROR, FUNCTION, NOACCEPT, NOSUP, PTF, SUP, RESTORE or USERMOD, those entries not found <u>as qualified,</u> will be reported as "not found".

  For example, if SYSMOD AR00001 is an APAR-type SYSMOD, "LIST CDS SYSMOD(AR00001) PTF" will fail.

LIST ACRQ/CRQ SYNTAX

```
LIST { ACRQ | CRQ }

     [SYSMOD[(sysmodid[,sysmodid]...)]]

     [FMIDC(fmid[,fmid]...)]]

     .
```

LIST ACRQ/CRQ Operands

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that all, or selected, SYSMOD entries are to be listed. SYSMOD entries contain the conditional requisite data supplied by the SYSMOD, sysmodid.

FMID[(fmid[,fmid]...)]
specifies that all, or selected, FMID entries are to be listed. FMID entries contain the names of the SYSMODs which have provided ++IF statements which reference the FMID, fmid.

List ACRQ/CRQ Output

Complete ACRQ/CRQ listing ... neither SYSMOD nor FMID specified.

This listing shows the names of all functions ("NAME" field) in the ACRQ/CRQ for which conditional requisites are present. For each of these functions, the list indicates the SYSMODs which have supplied conditional requisites ("CAUSER" field) and the requisites these "causers" have established ("IFREQ" field).

SYSMOD listing:

This listing shows conditional requisite data supplied by the SYSMOD. The "NAME" field is the SYSMOD's sysmod-id, the "Environment" field is the functional environment specified on the ++IF statement supplied and the "IFREQ" field is the requisite established for the associated functional environment.

FMID listing:

This listing shown the names of the SYSMODs ("CAUSER" field) which have provided conditional requisites for each FMID ("NAME").

The figures which follow illustrate the CRQ entries which would be created by the following PTFs:

```
         ++PTF(UZ00014) .
         ++VER(Z038) FMID(HXY0000) .
         ++IF FMID(HXY1010) THEN REQ(UZ00024) .
         ++IF FMID(HXY1011) THEN REQ(UZ00054) .

         ++PTF(UZ00015) .
         ++VER(Z038) FMID(HXY0000) .
         ++IF FMID(HXY1010) THEN REQ(UZ00025) .

         ++PTF(UZ00045) .
         ++VER(Z038) FMID(HXY0000) .
         ++IF FMID(HXY1011) THEN REQ(UZ00055) .
```

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT-

SMPCRQ   FMID/SYSMOD ENTRIES

NAME      CAUSER

HXY1010   UZ00014 IFREQ = UZ00024
          UZ00015 IFREQ = UZ00025

HXY1011   UZ00014 IFREQ = UZ00054
          UZ00045 IFREQ = UZ00055
```

Figure 26 - LIST CRQ

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCRQ     SYSMOD ENTRIES

NAME      ENV

UZ00014   HXY1010   IFREQ = UZ00024
          HXY1011   IFREQ = UZ00054

U200015   HXY1010   IFREQ = UZ00025

UZ00045   HXY1011   IFREQ = UZ00055
```

Figure 27 - LIST CRQ SYSMOD

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

SMPCRQ     FMID ENTRIES

NAME

HXY1010   SYSMOD CAUSERS = UZ00014 UZ00015

HXY1011   SYSMOD CAUSERS = UZ00014 U200045
```

Figure 28 - LIST CRQ FMID

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 a range of dates (inclusive) within the data set to be listed.

   The dates are specified as mm dd yy , where mm is the month (01-12), dd is the day (01-31), yy is the year (00-99) and blanks delimit the month, day and year as illustrated.

   If no date range is specified, the contents of the entire LOG data set are listed.

Example

```
      LIST LOG(08 08'80,08 11 80) .
```

will list the data in the log for August 8 through August 11, 1980.

```
      LIST LOG(08 09 80,08 09 80) .
```

will list the data in the log for August 9, 1980 only.

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 30 on page 163 is an example of output from LIST PTS MCS.

SYSMOD[(sysmodid[,sysmodid]...)]
specifies that information for all or the specified SYSMOD entries are to be listed.

Figure 31 on page 164 is an example of output from 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 | FUNC
specifies that function SYSMODs are to be listed.

NOACCEPT | NOACC
specifies that SYSMODs that appear in the PTS but do not appear in the ACDS are to be listed. These are SYSMODs which have not been successfully accepted. The ACDS dataset is required for this function.

NOAPPLY | NOAPP
specifies that SYSMODs that appear in the PTS but do not appear in the CDS are to be listed. These are SYSMODs which have not been successfully applied. The CDS dataset is required for this function.

PTF
specifies that PTF SYSMODs are to be listed.

USERMOD | USER
specifies that USERMOD SYSMODs are to be listed.

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.

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 29 on page 163 is an example of output from LIST PTS SYS.

LIST PTS Output

The following figures illustrate the output produced for three variations of PTS listing commands.

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

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      = IEWLG
                SYSPRINT  = LKDPRING
                PARM      = 'DECK,XREF,LET'
          UPDAT SYSPRINT  = UPDPRINT
          SREL            = Z038
          FMID            = GXY1000   HXY1010
```

Figure 29 - LIST PTS SYS

```
     DATE 80.092  TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

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)  .
                            ++ VER(Z038) .
                            ++ JCLIN RELFILE(1)  .
                            ++ SRC(MOD001) RELFILE(3) DISTLIB(AOS00).
                            ++ MOD(MOD001) RELFILE(2) DISTLIB(AOSXX).
                            ++ MOD(MOD002) RELFILE(2) DISTLIB(AOSXX).

UZ00010   M.C.S.  ENTRIES = ++ PTF(UZ00010)  .
                            ++ VER(X070) FMID(GXY2000)
                            ++ VER(Z038) FMID(GXY1000)  .
                            ++ SRCUPD(MOD001) DISTLIB(AOS00).
                            ++ MOD(MOD001) DISTLIB(AOSXX).
```

Figure 30 - LIST PTS MCS

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

  SMPPTS     SYSMOD ENTRIES
    NAME

  AZ00124    TYPE              = APAR
             STATUS            = APP
             DATE/TIME REC     = 77.318 14:28:54
             APPLY CDSID       = CDS1
             FMID  VER(001)    = GXY1000
             SREL  VER(001)    = Z038
             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

  UZ00010    TYPE              = PTF
             STATUS            = APP
             DATE/TIME REC     = 77.312 09:43:12
             APPLY CDSID       = CDS1
             SREL  VER(001)    = X070
             FMID  VER(001)    = GXY2000
             SREL  VER(002)    = Z038
             FMID  VER(002)    = GXY1000
             MOD               = MOD001
             SRCUPD            = MOD001
```

Figure 31 - LIST PTS SYSMOD

```
 LIST MASS SUMMARY REPORT FOR SMPPTS

 SYSMOD ENTRIES NOT FOUND
 MCS    ENTRIES NOT FOUND
```

Figure 32 - LIST MASS SUMMARY REPORT FOR SMPPTS

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:

- For listinas of the entire PTS:

  The "LIST MASS SUMMARY REPORT FOR SMPPTS" will show "MCS ENTRIES NOT FOUND" or "SYSMOD ENTRIES NOT FOUND" if there are either no MCS or SYSMOD entries respectively. Figure 32 on page 164 shows an example of the "LIST MASS SUMMARY REPORT FOR SMPPTS".

- For listings qualified by MCS or SYSMOD entry type:

  The "LIST MASS SUMMARY REPORT FOR SMPPTS" will show "MCS ENTRIES NOT FOUND" or "SYSMOD ENTRIES NOT FOUND" if there are either no MCS or no SYSMOD entries

- For listings of a specific MCS or SYSMOD entry:

  The "LIST SELECT SUMMARY REPORT FOR SMPPTS" will indicate those specific entries which are not found.

- For SYSMOD listings qualified only by SYSMOD type:

  The "LIST MASS SUMMARY REPORT FOR SMPPTS" will indicate those types which are not found.

- For specific SYSMOD listings qualified by SYSMOD status or type:

  If you attempt to list specific PTS SYSMOD entries qualified by APAR, FUNCTION, NOACCEPT, NOAPPLY, PTF or USERMOD, those entries not found as qualified, will be reported as "not found".

  For example, if SYSMOD AR00001 has been applied, "LIST PTS SYSMOD(AR00001) NOAPPLY" will fail.

LIST SCDS SYNTAX

```
LIST SCDS
     [SYSMOD[(sysmodid[,sysmodid]...)]]
     .
```

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.

LIST SCDS Operands

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.

LIST SCDS Output

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.

Entries ADDed by 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.

Entries updated by the SYSMOD:

ASSEM (UPDATE) - a list of any ASSEM entries updated by inline JCLIN.

LMOD (UPDATE) - a list of any LMOD entries updated 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.

Entries deleted by the SYSMOD:

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.

Backed-Up Entry:

The BACKUP entry for each updated or deleted entry is listed and formatted as it would appear in a CDS listing.

```
     DATE 80.092 TIME 01:23:46    HMASMP LVL 04.18 SMPLIST OUTPUT

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 = ... assembler input on CDS prior
                          ... to input supplied by HXY1010
```

Figure 33 - LIST SCDS SYSMOD(HXY1010)

The SMP LOG control statement is used to write 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 PROGRAMING CONSIDERATIONS

The LOG data set must be a sequential dataset. The dataset should be defined in the DD statement as a dataset that can be modified; for example:

```
//SMPLOG DD DSN=SYS1.SMPLOG,DISP=MOD
```

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

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. A SYSMOD which has already been successfully received cannot be re-received by specifically selecting it; a SYSMOD which is successfully received must be removed from the PTS using the REJECT function. 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 (which have not already been received) and are eligible, will be 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

- 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 block size.

- 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.

- 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.

- 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.

- SYSMODs are received regardless of the status of any requisite, negative prerequisite or prerequisite SYSMODs.

- 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.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

The SMP REJECT control statement invokes REJECT processing to remove SYSMODs from the SMPPTS and delete temporary libraries loaded during RECEIVE. 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 used 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 affect on the current REJECT processing.

REJECT DDNAMES

SMPACDS	(required for REJECT PURGE)
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

- A data set can be specified as a COMPRESS operand value even if it is not affected by the REJECT process.
- The compress function does not process keyed or unmovable data sets.
- The compress operand should not select or make eligible for compress any target system libraries of a running operating system.
- Processing time can increase significantly when the COMPRESS operand is specified.
- If a function SYSMOD that has been neither applied nor accepted (as determined from PTS SYSMOD APPID and ACCID subentries) is rejected, the FMID subentry for its SYSMOD-ID is deleted from the PTS SYSTEM entry. Note: when PURGE is specified, the FMID subentries for rejected SYSMODs are not affected.

- The RETRY facility (RETRYDDN in CDS) can be preferred to the COMPRESS for performance. In this case only datasets that need to be, are compressed.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

The RESETRC control statement resets the return coda 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

- 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.

- 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.

- 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.

- The RESETRC control statement does not have any return codes.

The SMP RESTORE control statement invokes RESTORE processing which removes SYSMODs processed by APPLY from the operating system libraries. Any number of RESTORE control statements can be coded in an SMP job step. SELECT or GROUP must be specified.

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.

<u>Note:</u> 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'.

<u>Note:</u> 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(<u>READ</u> | 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.

<u>Note:</u> If DIS(NO) is specified with the CHECK operand, it is ignored and DISCREAD), 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)
tgtlib	(one for each target system library to be restored)

RESTORE PROGRAMMING CONSIDERATIONS

- If the REJECT indicator is set off in the PTS SYSTEM entry, then a successfully restored SYSMOD is not deleted from the PTS.

- 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.

- 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 on the SMPRPT data set, if supplied on SMPOUT.

- 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 cannot 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.

- RESTORE processing does not replace the nucleus with the saved copy, but re-links it using the last version of modules accepted on the DLIB's. The saved nucleus is available only to provide an alternate nucleus for IPL should an applied SYSMOD damage IEANUC01.

- 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.

- 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.

- 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.

- 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.

- If you do not use SMP to recover after a failure, and choose the option of restoring your system and the distribution libraries via system and DLIB restore tapes, you must ensure that the CDS, MTS, ACDS, PTS, and STS are also restored to their previous levels.

RETURN CODES

See Chapter 1 of the OS/VS SMP Messages And Codes, (GC38-1047) manual for information about Return codes and Error recovery procedures.

The UCLIN control statement invokes UCLIN processing to update entries on the SMP data sets.

The UCLIN facility 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.

The UCLIN statement itself specifies the data set whose entries are to be updated; the UCLIN statement is followed by a set of UCL statements which specify the updates to be made to the entries. UCLIN processing is terminated by the ENDUCL statement.

The operation of a UCLIN request is as follows:

- All requested changes are made to an in-storage copy of the entry as keywords are encountered.
- When end of statement '.' (period) is found, SMP will check the remaining data in the entry to insure that it is valid. Any changes that result in an incorrect entry will either be fixed by SMP at this time or an error message will be generated. For example:
  - Deleting the APP indicator without deleting the APPDATE will result in SMP turning on the APP indicator again.
  - Deleting the ERR indicator in a SYSMOD marked RESTORE will result in the ERR indicator being turned on as RESTORE without ERR is an invalid status.
  - Adding ACCDATE will also add the ACC indicator.
  - Adding a module without a DISTLIB will result in an error message.
- If any "error" message is generated no requested changes are made to the actual entry on the specified dataset.

UCLIN SYNTAX

```
UCLIN dataset
       [DIS( READ | NO | WRITE ) ]
       [RC(function=code[,function=code]...)]
       .
```

UCLIN OPERANDS

dataset - Specifies the SMP dataset to be updated.
The allowable values are: ACDS, ACRQ, CDS. CRQ, MTS, PTS, SCDS and STS.
Note, If FMID= is coded on the EXEC statement, the default data set will be the CDS.

DIS(READ | 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 sat 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 MIS 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

- Each UCLIN control statement must be followed by at least one UCL statement.

- The ENDUCL control statement must terminate the UCL statements.

- Use the DIS(NO) option only when the number of updates to entries is small or when the trade off between storage utilization and performance has to be made in favor of storage.

- 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.

- 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.

UCL statements are used to add, delete, and modify entries in the ACDS, ACRQ, CDS, CRQ, MIS, 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 } entry_type(entry_name) [sub_entry[,sub_entry]...] .
```

UCL OPERANDS

ADD
: specifies that new data is to be added to an existing entry or that a new entry is to be created.

DEL
: specifies that an entry is to be deleted or, within an entry, subentries are to be deleted and indicators placed in reset state.

REP
: specifies that subentries are to be replaced and indicators placed in set state in an existing entry. If the entry does not exist, it will be created using the criteria for ADD operations.

entry_type
: specifies the entry type of the entry to be updated. The allowable entry types are: ASSEM, DLIB, FMID, LMOD, MAC, MOD, PTF, SRC, SYS and SYSMOD. The following pages describe the allowable updates for each of these entry types.

entry_name
: specifies the name of the entry to be updated.

subentry
: specifies the sub-entries and indicators that are to be updated. The syntax and explanation of these options are described on the following pages.

The UCL syntax descriptions in this chapter are arranged by data set and entry type as follows:

DATA SET	Entry Type	Page
CDS	ASSEM	195
	DLIB	197
	LMOD	199
CDS & ACDS	MAC	203
	MOD	205
	SRC	209
	SYSTEM	211
	SYSMOD	213
CRQ & ACRQ	FMID	223
	SYSMOD	225
PTS	SYSMOD	227
	SYSTEM	229
SCDS	SYSMOD	239
STS	SRC	241
MTS	MAC	243

Appendix E contains a set of charts which cross reference the UCL operations to their appropriate SMP data sets and entries therein.

UCL ADD CONSIDERATIONS:

The ADD function is valid for entries on the ACDS, ACRQ, CDS, CRQ and 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, 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, 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:

```
UCLIN CDS .
ADD MOD(XYZ) DISTLIB(AOS99).
ENDUCL .
```

creates a MOD entry on the CDS for module XYZ.

Subentries are added to the existing entry using the specified values. For example:

```
UCLIN CDS .
ADD MOD(XYZ) UMID(UZ12345,UZ13579).
ENDUCL .
```

adds two UMID subentries to MOD entry XYZ.

Indicators are placed in set state in the existing entry. For example:

```
UCLIN CDS .
ADD SYSMOD(UZ12345) RESTORE.
ENDUCL .
```

sets the RESTORE indicator in the CDS SYSMOD entry, UZ12345.

UCL DEL CONSIDERATIONS:

The DEL function is valid for entries on the ACDS, ACRQ, CDS, CRQ, MIS, PTS, SCDS and STS.

For DEL operations, the specified entry must exist. When subentries are specified, they must exist and contain the same value as is specified in the operand or be unconditionally deleted, and indicators must be in set state. If any of these conditions are false, the invalid condition is corrected by SMP or 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:

```
UCLIN ACDS .
DEL SYSMOD(UZ12345).
ENDUCL .
```

deletes the ACDS 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:

```
UCLIN CDS .
DEL MAC(ABC) UMID( ).
ENDUCL .
```

deletes all UMID subentries in the CDS 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:

```
UCLIN ACDS .
DEL SYSMOD(UZ12345) ERROR.
ENDUCL .
```

resets the ERROR indicator in the ACDS SYSMOD entry for UZ12345.

UCL REP CONSIDERATIONS:

The REP function is valid for entries on the ACDS, ACRQ, CDS, CRQ and PTS.

For REP operations, if the entry did not exist or a subentry within an existing entry did not exist, 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, all subentries of the same type are replaced with the values specified in the operand. For example:

```
UCLIN ACDS .
REP SYSMOD(UZ12345) SUPING(AZ11111,AZ11122).
ENDUCL .
```

replaces all SUPING subentries in the ACDS SYSMOD entry for U212345.

Indicators within an existing entry are placed in set state. For example:

```
UCLIN ACDS .
REP SYSMOD(UZ12345) ERROR.
ENDUCL .
```

sets the ERROR indicator in the ACDS SYSMOD entry for UZ12345.

COMMON ERRORS

1. During the checking phase of UCL SMP checks to ensure that the modified entry contains sufficient and consistent data. If not, SMP issues the following message:

   ```
   HMA2574 SPECIFIED UPDATE RESULTS IN INSUFFICIENT DATA - xxx REQUIRED
   ```

   where xxx is the operand that is required.

   The xxx is not the operand that is required to be present on the UCL statement (although it could be). It is rather the operand that is required to be present in the entry after the modifications have been made.

   For example, after the following UCL statement for an existing SMPCDS SYSMOD entry

   ```
   UCLIN CDS.
   DEL SYSMOD(UR11111) MOD(HMASMDRV) APPDATE(79001) APPTIME(08:00:00) APP.
   ENDUCL.
   ```

   SMP will issue message HMA257 for the APPDATE and APPTIME operand because SMP will not allow an SMPCDS SYSMOD entry to be modified such that there is not and APPLY date and time. The message will not be issued for the APP indicator because SMP resets that indicator on automatically during the checking phase. SMP cannot do that for the date and time because there is no way of knowing the correct data value for these option.

2. Insure that all changes you make to one entry are consistent with other existing entries.

3. The DATE operand on SYSMOD entries in the SMPCDS and SMPACDS is provided to maintain compatibility with previous releases of SMP that did not support unique dates for each of the SMP status (REC, APP, ACC, etc). This keyword can only be used when the compatibility feature of SMP (EXEC FMID operand) is used. If it is specified SMP will use the date for all the required status date fields. For example if a SYSMOD entry is marked REC and APP then the date specifed by the DATE operand will be used for both the RECDATE and APPDATE.

Syntax

```
{ ADD | DEL | REP } ASSEM (name)
[ ++ASMIN
     assembler input CARD1
     assembler input CARD2
         .
         .
  ++ENDASMIN ]
[ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
[ LASTUPDTYPE({ADD | UPD}) ]
.
```

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 column 1. If specified then ENDASMIN 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
: Terminates assembler input. This control word must begin in column 1 and is only used when ++ASMIN is used.

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.

Syntax

```
{ ADD | DEL | REP } DLIB (name)
   [SYSLIB(ddname[,ddname])]
   [ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
   [ LASTUPDTYPE({ADD | UPD}) ]
   .
```

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.

Syntax

```
{ ADD | DEL | REP } LMOD(name)
   [AC=1]
   [ALIGN2]
   [COPY]
   [DC]
   [NE]
   [OVLY]
   [REFR]
   [RENT]
   [REUS]
   [SCTR]
   [STD]
   [SYSLIB(ddname[,ddname])]
[ ++LMODIN
     link edit control statement
     link edit control statement
        "  "
        "  "
  ++ENDLMODIN ]
   [ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
   [ LASTUPDTYPE({ADD | UPD}) ]
   .
```

Operands

LMOD(name)
specifies a LMOD entry to be deleted, or subentries and indicators within the LMOD entry to be added, deleted, or replaced on the CDS, where "name" is the one to eight character LMOD entry name.

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 or modify the CDS module entry. No other options may be specified on the same line as LMODIN.

++ENDLMODIN
: Terminates Linkage Editor input. This control word must begin in column 1 and is used only when ++LMODIN is used.

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.

Syntax

```
{ ADD | DEL | REP } MAC(name)
   [DISTLIB(ddname)]
   [FMID(sysmodid)]
   [GENASM(name[,name]...) | ASSEM(name[,name]..)]
   [MALIAS(alias[,alias]...)]
   [RMID(sysmodid)]
   [SYSLIB(ddname)]
   [UMID(sysmodid[,sysmodid]...)]
   [ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
   [ LASTUPDTYPE({ADD | UPD}) ]
   .
```

Operands

MAC(name)
specifies a MAC 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 macro name.

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]...) | ASSEM(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. Either GENASM or ASSEM can be specified.

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|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.

Syntax

```
{ ADD | DEL | REP } MOD(name)
   [ASSEMBLE]
   [DALIAS(alias[,alias]...)]
   [DISTLIB(ddname)]
   [FMID(sysmodid)]
   [LMOD(name[,name]...)]
   [RMID(sysmodid)]
   [RMIDASM]
   [TALIAS(alias[,alias]...)]
   [UMID(sysmodid[,sysmodid]...)]
   [ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
   [ LASTUPDTYPE({ADD | UPD}) ]
   [AC=1]
   [ALIGN2]
   [DC]
   [NE]
   [ONLY]
   [REFR]
   [RENT]
   [REUS]
   [SCTR]
   .
```

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.

ASSEMBLE
Specifies the 'must assemble' indicator for the module. When sat on (ADD), the indicator causes object text to be ignored and an assembly performed.

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.

RMIDASM
specifies the last replacement (RMID) to the module was done by a SYSMOD which caused (or could have caused) an assembly of the module as a result of a macro or source modification.

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|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.

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.

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 it 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 sat, the SCTR parameter is passed to the linkage editor program when the load module is link edited.

Syntax

```
{ ADD | DEL | REP } SRC(name)
   [DISTLIB(ddname)]
   [FMID(sysmodid)]
   [RMID(sysmodid)]
   [SYSLIB(ddname)]
   [UMID(sysmodid[,sysmodid]...)]
   [ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
   [ LASTUPDTYPE({ADD | UPD}) ]
   .
```

Operands

SRC(name)
specifies a SRC 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 source module name.

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|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.

Syntax

```
{ ADD | DEL | REP } SYS
   [CDSID(name)]
   [NUCID(n)]
   [PEMAX(nnnn)]
   [RETRYDDN({ALL | [ddname][,ddname]... })
   [SAVEMTS]
   [SAVESTS]
   [SREL(cnnn)]
   .
```

Operands

SYS
specifies a SYSTEM entry or subentries and indicators within an entry to be added, deleted, or replaced on the ACDS or CDS.

Note: Changes to a SYSTEM entry are effective immediately after processing of the UCL SYS statement.

CDSID(name)
specifies the CDSID subentry. "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.

NUCID(n)
specifies the NUCID subentry. "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.

Note: This alternate nucleus id may be overridden by the NUCID operand on the APPLY statement.

PEMAX(nnnn)
specifies the PEMAX subentry. "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.

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 sat 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, 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.

Note: Unlike the normal SMP compress (via COMPRESS keyword), A compress recovery done as the result of coding RETRYDDN(ALL) will attempt to compress any candidate including SYS1.LINKLIB.

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 indicator 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 indicator is shown as "YES" if the SAVESTS indicator is sat and as "NO" if the SAVESTS indicator is reset.

SREL(cnnn)
specifies the SREL subentry. "cnnn" is a system release identifier.

Note: When creating a SYSTEM entry, this operand must be specified. The SREL subentry cannot be deleted from the ACDS and CDS SYSTEM entries.

Syntax

```
{ ADD | DEL | REP } SYSMOD(name) [ sysmod_type ]
   [ACCDATE(yyddd)]
   [ACCEPT]
   [ACCTIME(hh:mm:ss)
   [APPDATE(yyddd)]
   [APPLY]
   [APPTIME(hh:mm:ss)
   [ASSEM(name[,name]...)]
   [BYPASS]
   [DELBY(sysmodid)]
   [DELETE(sysmodid[,sysmodid]...)]
   [ERROR]
   [FESN]
   [FMID(sysmodid)]
   [JCLIN]
   [LASTSUP(sysmodid)]
   [ LASTUPD({JCLIN | UCLIN | sysmodid}) ]
   [ LASTUPDTYPE({ADD | UPD}) ]
   [MAC(name[,name]...)]
   [MACUPD(name[,name]...)]
   [MOD(name[,nama)...)]
   [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(5ysmodid[,sysmodid]...)]
   [SZAP(name[,name]...)]
   [UPDTE(name[,name]...)]
   [UCLDATE(yyddd)]
   [UCLTIME(hh:mm:ss )]
   [VERNUM(value)]
   [VERSION(sysmodid[,sysmodid]...)]
   [XZAP(name[,name]...)]
   .
```

Operands

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_type
specifies the type of SYSMOD. This operand may be specified as either PTF, APAR, USERMOD or FUNCTION.

If an attempt is made to ADD a SYSMOD type to an existing entry which already has a sysmod_type, an error message is issued and the sysmod_type is not changed. The sysmod_type may be changed using REP. The DEL operation will leave the entry with no sysmod_type.

Note: The default sysmod_type for ADD operations is PTF.

ACCDATE(yyddd)
specifies the ACCDATE subentry. "yyddd" is the Julian date that the SYSMOD was accepted. If the ACCDATE subentry is present in a SYSMOD entry, 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.

Note: When deleting ACCURATE, the ACC indicator, ACCTIME and if necessary the ERR indicator must also be deleted.

ACCEPT
specifies the ACCEPT indicator. When this indicator is set, the SYSMOD has been 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. Deleting the ACCEPT indicator without deleting the APPDATE will result in SMP turning on the ACCEPT indicator again.

ACCTIME(hh:mm:ss)
specifies the ACCTIME subentry. "hh:mm:ss" is the hour, minute, and second that the SYSMOD was accepted. A colon must be specified between digits. If the ACCDATE is changed or added 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 ACCTIME is not specified then ACCTIME is set to 00:00:00.

APPDATE(yyddd)
specifies the APPDATE subentry. "yyddd" is the Julian date that the SYSMOD was applied. If the APPDATE subentry is present in a SYSMOD entry, the APPLY indicator is sat. 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.

Note: When deleting APPDATE, the APPLY indicator, APPTIME and if necessary the ERR indicator must also be deleted.

APPLY
specifies the APPLY indicator. When this indicator is sat, the SYSMOD has been applied. 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. Deleting the APPLY indicator without deleting the APPDATE will result in SMP turning on the APPLY indicator again.

APPTIME(hh:mm:ss)
specifies the APPTIME subentry. "hh:mm:ss" is the hour, minute, and second that the SYSMOD was applied. A colon must be specified between digits. If the APPDATE is changed or added 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 APPTIME is not specified then APPTIME is set to 00:00:00.

ASSEM(name[,name]...)
specifies one or more ASSEM subentries. "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. 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. "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. "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. This operand can also be specified as "ERR". When this indicator is set, the SYSMOD is considered to have been unsuccessfully processed.

Note: Deleting the ERROR indicator in a SYSMOD marked RESTORE will result in the ERR indicator being left on if the RESDATE, RESTORE indicator and RESTIME is not also deleted.

FESN(fe service number)
specifies a seven character FE service number.

FMID(sysmodid)
specifies the FMID subentry. "sysmodid" is the SYSMOD-ID of a function SYSMOD. A SYSMOD's FMID is the function which "owns" the SYSMOD.

Note: This operand is required when creating a SYSMOD entry that is not a superseded-only type.

JCLIN
indicates that the SYSMOD contains inline JCLIN.

LASTSUP(sysmodid)
specifies the LASTSUP subentry. "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. "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. "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. "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. "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. "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. "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. "hh:mm:ss" is the hour, minute, and second that the SYSMOD was received. A colon must be specified between digits. If the RECDATE is changed or added 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 RECTIME is not specified then RECTIME is set to 00:00:00.

REGEN
specifies the REGEN indicator. 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. "sysmodid" is the SYSMOD-ID of a SYSMOD that is a requisite of this SYSMOD.

Note: 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.

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, 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.

Note: When deleting the RESDATE, the RESTORE indicator, RESTIME and ERROR indicator also must be deleted.

Note: This subentry is valid only in a CDS SYSMOD entry.

RESTIME(hh:mm:ss)
specifies the RESTIME subentry. "hh:mm:ss" are the hour, minute, and second that the SYSMOD was restored. A colon must be specified between digits. If the RESDATE is changed or added 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 RESTIME is not specified then RESTIME is set to 00:00:00.

Note: This subentry is valid only in a CDS SYSMOD entry.

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. Deleting the RESTORE indicator without deleting the RESDATE will result in SMP turning on the RESTORE indicator again.

Note: This indicator is valid only in a CDS SYSMOD entry.

RMAC(name[,name]...)
specifies one or more RMAC subentries. "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. "name" is the name of a macro updated by this SYSMOD. Each RMACUPD 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.

Note: You can specify "RMUPD" in place of RMACUPD and get the same results.

RMOD(name[,name]...)
specifies one or more RMOD subentries. "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. "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. "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.

Note: You can specify "RSUPD" in place of RSRCUPD and get the same results.

RSZAP(name[,name]...)
specifies one or more RSZAP subentries. "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. "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. "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. "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. "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. "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. "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.

UCLDATE(yyddd)
specifies the UCLDATE subentry. 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. "hh:mm:ss" is the hour, .minute, and second that the SYSMOD was accepted. A colon must be specified between digits. If the

UCLDATE is changed or added 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 UCLTIME is not specified then UCLTIME is set to 00:00:00.

UPDTE(name[,name]...)
specifies one or more MACUPD subentries.

Note: This operand is equivalent to the MACUPD operand and is included for compatibility with UCL statements processable by previous versions of SMP.

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 than 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.

VERSION(sysmodid[,sysmodid]...)
specifies one or more VERSION subentries. "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. "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.

Syntax

```
{ ADD | DEL | REP } FMID(name)
   [SYSMOD(sysmodid(,sysmodid])]
   .
```

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.

<u>Note:</u> When creating a FMID entry, this operand must be specified. When deleting a SYSMOD subentry, at least one SYSMOD subentry must remain.

When adding a SYSMOD to an FMID entry, the corresponding ACRQ/CRQ SYSMOD entry must generally be updated.

For example, consider that SYSMOD UZ00000 should have supplied an IF conditional requisite statement indicating requisites for functions F111111 and F222222.

```
++IF FMID(F111111) THEN REQ(UZ11111) .
++IF FMID(F222222) THEN REQ(UZ22222) .
```

CRQ/ACRQ FMID entries must be created which indicate that SYSMOD UZ00000 has requisites for the two functions:

```
UCLIN CRQ .
 ADD FMID(F111111) SYSMOD(UZ00000) .
 ADD FMID(F222222) SYSMOD(UZ00000) .
ENDUCL.
```

A CRQ/ACRQ SYSMOD entry must be created for SYSMOD UZ00000 which indicates the actual requisites:

```
UCLIN CRQ .
 ADD SYSMOD(UZ00000) FMID(F111111) REQ(UZ11111) .
 ADD SYSMOD(UZ00000) FMID(F222222) REQ(UZ22222) .
ENDUCL.
```

Syntax

```
{ ADD | DEL | REP } SYSMOD(name)
   [FMID(sysmodid)]
   [REQ(sysmodid[,sysmodid]...)]
   .
```

Operands

SYSMOD(name)
specifies a SYSMOD entry or subentries 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.

Note: The associated FMID entry on the ACRQ or CRQ should be updated to reflect changes made to a SYSMOD entry.

FMID(sysmodid)
specifies the FMID subentry where "sysmodid" is the SYSMOD-ID of a function SYSMOD. 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, the REQ operand must also be specified and must physically follow the FMID operand on UCL statement. If the DEL operand is specified and 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, the SYSMOD-IDs specified in REQ operand replace the existing REQ subentries in the SYSMOD entry.

REQ(sysmodid[,sysmodid]...)
specifies one or more REQ subentries of the ACRQ or CRQ SYSMOD entry, where "sysmodid" is the SYSMOD-ID of a SYSMOD that is a requisite of this SYSMOD. 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.

Syntax

```
{ ADD | DEL | REP } SYSMOD(name)
   [ACCID(cdsid[,cdsid]...)]
   [APPID(cdsid[,cdsid]...)]
   .
```

Operands

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. 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. If the PTS SYSMOD entry is deleted, the associated MCS entry is also deleted.

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.

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.

Syntax

```
{ ADD | DEL | REP } SYS
   [ASMNAME(name)]
   [ASMPARM(parm)]
   [ASMPRINT(ddname)]
   [ASMRC(value)]
   [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)]
   [PAGELEN(nnnn)]
   [PEMAX(nnnn)]
   [PURGE]
   [REJECT]
   [RETRYNAME(name)]
   [RETRYPARM(parm)]
   [RETRYPRINT(ddname)]
   [RETRYRC(value)]
   [SREL(cnnnt,cnnn]...)]
   [UPDATNAME(name)]
   [UPDATPARM(parm)]
   [UPDATPRINT(ddname)]
   [UPDATRC(value)]
   [ZAPNAME(name)]
   [ZAPPARM(parm)]
   [ZAPPRINT(ddname)]
   [ZAPRC(value)]
   .
```

Operands

SYS
   specifies a SYSTEM entry or subentries and indicators within an entry to be added, deleted, or replaced on the PTS.

Note: Changes to a SYSTEM entry are effective immediately after processing of the UCL SYS statement.

ASMNAME(name)
specifies the ASMNAME subentry. "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, 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. "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, 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. "ddname" is the ddname for the output listing data set produced by the assembler program.

Note: If the ASMPRINT subentry is not present, the ddname SYSPRINT is used. 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. "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, 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, the value of 4 is compared with the assembler program return code.

COMPNAME(name)
specifies the COMPNAME subentry. "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, SMP invokes the program IEBCOPY to perform the PDS compress function.

COMPPARM(parm)
specifies the COMPPARM subentry. "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, 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. "ddname" is the ddname for the output listing data set produced by the PDS compress program.

  Note: If the COMPPRINT subentry is not present, the ddname SYSPRINT is used. 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. "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, 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, the value of 0 is compared with the PDS compress program return code.

COPYNAME(name)
: specifies the COPYNAME subentry. "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, SMP invokes the program IEBCOPY to perform the PDS copy and load functions.

COPYPARM(parm)
: specifies the COPYPARM subentry. "perm" 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, 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. "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, the ddname SYSPRINT is used. 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. "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, 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, the 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. "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, no high order qualifier is used during allocation and subsequent accessing.

DSSPACE(prim,sec,dirblks)
specifies the DSSPACE subentry space parameters for data sets that are allocated during RECEIVE processing for library loading. "prim" and "sac" 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. "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. "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, SMP invokes the program IEHIOSUP to perform the IEHIOSUP function.

IOSUPPARM(parm)
specifies the IOSUPPARM subentry. "perm" 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, 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. "ddname" is the ddname for the output listing data set produced by the IEHIOSUP program.

Note: If the IOSUPPRINT subentry is not present, the ddname SYSPRINT is used.

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. "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, 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, the value of 0 is compared with the IEHIOSUP program return code.

LKEDNAME(name)
: specifies the LKEDNAME subentry. "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, SMP invokes the program IEWL to perform the linkage editor function.

LKEDPARM(parm)
: specifies the LKEDPARM subentry. "perm" 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.

If no LKEDPARM subentry is present, the parameters "LET, LIST, XREF and NCAL" are used.

Note: These parameters are passed in addition to the link edit attributes determined during APPLY and ACCEPT processing.

Note: The parameters LET and NCAL are normally required for maintenance of IBM operating systems. If this subentry is updated, be sure to include LET and NCAL in the parameters chosen.

LKEDPRINT(ddname)
: specifies the LKEDPRINT subentry. "ddname" is the ddname for the output listing data set produced by the linkage editor program.

Note: If the LKEDPRINT subentry is not present, the ddname SYSPRINT is used. 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. "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. 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, the value of 8 is compared with the linkage editor program return code.

PAGELEN(nnnn)
specifies the PAGELEN subentry. "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 number of lines per page is 60.

PEMAX(nnnn)
specifies the PEMAX subentry. "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 indicator 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 indicator 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 attempting a RETRY following a UTILITY failure.

NOTE: 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 'perm' 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.

NOTE: 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.

NOTE: If the RETRYPRINT subentry is not present in the PTS system entry, then the ddname SYSPRINT is used. 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, 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.

NOTE: If the RETRYRC subentry is not present in the PTS system entry, then the value of 0 is compared with the recovery COMPRESS program return code.

SREL(cnnn[,cnnn]...)
specifies the SREL subentriers. "cnnn" is a system release identifier. Multiple "cnnn" values may appear in the PIS SYSTEM entry.

Note: At least one SREL subentry must be present.

UPDATNAME(name)
specifies the UPDATNAME subentry. "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, SMP invokes the program IEBUPDTE to perform the text update function.

UPDATPARM(parm)
specifies the UPDATPARM subentry. "perm" 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, 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, 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. "ddname" is the ddname for the output listing data set produced by the text update program.

Note: If the UPDATPRINT subentry is not present, the ddname SYSPRINT is used. 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. "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, 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, the value of 0 is compared with the text update program return code.

ZAPNAME(name)

specifies the ZAPNAME subentry. "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, SMP invokes the program IMASPZAP to perform the IMASPZAP function.

ZAPPARM(parm)

specifies the ZAPPARM subentry. "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, 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. "ddname" is the ddname for the output listing data sat produced by the IMASPZAP program.

Note: If the ZAPPRINT subentry is not present, the ddname SYSPRINT is used. 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. "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, 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.

<u>Note:</u> **If the ZAPRC subentry is not present, the value of 4 is compared with the IMASPZAP program return code.**

Syntax

```
 DEL SYSMOD(name)
   .
```

Operands

SYSMOD(name)
   specifies a SYSMOD entry which is to be deleted. For the SCDS, the only valid operation is DEL.

Syntax

```
 DEL SRC(name) .
```

Operands

SRC(name)
specifies a SRC entry which is to be deleted. For the STS, the only valid operation is DEL.

Syntax

```
 DEL MAC(name) .
```

Operands

MAC(name)
specifies a MAC entry which is to be deleted. For the MIS, the only valid operation is DEL.

The UNLOAD control statement will unload entries from the CDS or ACDS to the SMPPUNCH dataset. The output produced is in the form of UCL statements which, if processed by UCLIN, will recreate the unloaded ACDS/CDS entries. This function enables the user to unload all or selected parts of a CDS or ACDS for backup or initialization of entries on other control datasets.

UNLOAD SYNTAX

```
UNLOAD dataset [entry[,entry]...] [option[,option]...]
[RC(function=code[,function=code]...)]
.
```

UNLOAD OPERANDS

dataset
specifies either the ACDS or CDS data set.

entry
specifies the entry or entry types to be unloaded. (See below)

option
specifies the options which control the data unloaded. (See below)

If no entries or options are specified, all of the data in the specified data set is unloaded.

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 UNLOAD 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, UNLOAD processing is bypassed and UNLOAD 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, UNLOAD is executed.

- Previous processing by any SMP function not specified on the RC operand has no effect on the current UNLOAD processing.

UNLOAD ENTRY TYPES AND OPTIONS

ADDINPUT
: directs SMP to read the set of ADDIN 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 ADDIN control statements override the data present in the dataset being UNLOADED. See ADDIN statement description on page 249 for further information.

ASSEM[(assemname[,assemname]...)]
: specifies that information for all ASSEM entries or the specified ASSEM entries is to be unloaded. Cannot be specified for the ACDS.

DLIB[(dlibname[,dlibname]...)]
: specifies that information for all DLIB entries or the specified DLIB entries is to be unloaded. Cannot be specified for the ACDS.

FORFMID(fmid)
: specifies that only those entries whose FMID subentry field matches the value specified will be unloaded. Note: Since ASSEM, LMOD, DLIB and SYSTEM entries have no FMID associated with them, they are unloaded when FORFMID is specified without MOD, MAC, SRC or SYSMOD entry qualifiers.

MAC[(macname[,macname]...)]
: specifies that information for all MAC entries or the specified MAC entries is to be unloaded.

MOD[(modname[,modname]...)]
: specifies that information for all MOD entries or the specified MOD entries is to be unloaded.

SRC[(srcname[,srcname]...)]
: specifies that information for all SRC entries or the specified SRC entries is to be unloaded.

SYSMOD[(sysmodid[,sysmodid]...)] [sysmod_options]
: specifies that information for all SYSMOD entries or the specified SYSMOD entries is to be unloaded.

  The following sysmod_options may be specified to control the types of SYSMODs unloaded:

  APAR - unload APAR-type SYSMODs

  DELETE - unload deleted SYSMODs

  ERROR - unload SYSMODs whose ERROR indicator is set

  FUNCTION - unload FUNCTION-type SYSMODs

NOAPPLY - unload SYSMODs that have been received and accepted, but not applied. 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'.

<u>NOAPPLY cannot be specified for the CDS data set.</u>

NOSUP - unload SYSMODs which are not superseded This operand is mutually exclusive with the SUP operand. Specification of both causes a syntax error.

PTF - unload PTF-type SYSMODs

SUP - unload superseded SYSMODs. This operand is mutually exclusive with the NOSUP operand. Specification of both causes a syntax error.

USERMOD - unload USERMOD-type SYSMODs.

SYS
specifies that system entry is to be unloaded.

UCLINDIS(READ|WRITE|NO)
controls the DIS option to be generated on the UCLIN statement produced. For example, if "UNLOAD CDS UCLINDIS(NO)" is used to unload a CDS, the UCLIN statement produced in SMPPUNCH is "UCLIN CDS DIS(NO)."

UNLOAD DDNAMES

SMPADDIN	(required if ADDINPUT is specified)
SMPACDS	(required if the ACDS operand is specified)
SMPCDS	(required if the CDS operand is specified)
SMPCNTL	(required)
SMPLOG	(required)
SMPOUT	(required)
SMPPUNCH	(required)

PROGRAMMING 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 have to perform.

ADDIN control statements allow the user to change the UCLIN statements generated by UNLOAD. The data provided in the ADDIN statements override the corresponding data in the CDS or ACDS entry. The format of the ADDINPUT statements is similiar to that of the UCLIN statements, however, only a limited number of fields are supported.

The ADDIN facility is included to be compatible with the facility in SMP Release 3. In SMP Release 3 this facility was used for creation of SMP Release 4 UCLIN statements; in SMP Release 4 there is little use for this facility.

ADDIN control statements are placed in the SMPADDIN dataset.

ADDIN SYNTAX AND OPERANDS

ADDIN - MACRO Entries

```
REP MAC(macname) FMID(sysmodid) .
```

Changes the FMID value generated for the specified macro.

ADDIN - MODULE Entries

```
REP MOD(modname) FMID(sysmodid) .
```

Changes the FMID value generated for the specified module.

ADDIN - SOURCE Entries

```
REP SRC(srcname) FMID(sysmodid) .
```

Changes the FMID value generated for the specified source.

ADDIN - SYSTEM Entries

```
REP SYS [CDSID(name)] (SREL(cnnn)] .
```

Changes the CDSID and/or SREL value generated for the SYSTEM entry.

<u>ADDIN - SYSMOD Entries</u>

```
   REP SYSMOD(sysmodid) [NEWNAME(sysmodid)]

      [FMID(sysmodid)] [FUNCTION|PTF|APAR|USERMOD] .
```

Changes the specified SYSMOD's name and/or FMID and/or type.

ADDIN DDNAMES

(see "UNLOAD Control Statement")

UNLOAD/ADDIN EXAMPLE

The entries in the CDS may be unloaded and the CDSID generated for subsequent UCLIN processing may be changed using the following set of control and ADDIN statements:

```
    //SMPCNTL DD
     UNLOAD CDS ADDINPUT /* Unload the Entire CDS */ .
    /*

    //SMPADDIN DD *
     REP SYS CDSID(BKUPCDS) /* Change CDSID */ .
    /*
```

The output produced (to SMPPUNCH) will be of the form

```
     UCLIN CDS .
     REP SYS SREL(xxxx) NUCID(n)
             CDSID(BKUPCDS) /* As specified by ADDIN data */ .
     REP ASSEM(....
     REP DLIB(....
     REP LMOD(....
     REP MAC(....
     REP MOD(....
     REP SRC(....
     REP SYSMOD(....
```

CHAPTER 6: MODIFICATION CONTROL STATEMENTS

Modification Control Statements are the input definitions of elements to be added to, modified in, or 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)

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

- 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.

- An APAR SYSMOD is accepted into the distribution libraries only when the APARS keyword is specified on the ACCEPT control statement.

- 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 IFBMOD01 is needed to answer APAR OZ12345 on an MVS system. The module must be at the service level provided by PTF U200004 for function FXY1040.

```
++APAR(AZ12345).
++VER(Z038) FMID(FXY1040) PRE(UZ00004).
++ZAP(IFBMOD01) DISTLIB(AOSFB).
  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)
    [FESN(fe service number)]
    [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.

FESN(fe service number)
   specifies a seven character FE service number.

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

- 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.

- ++MACUPD, ++UPDTE, ++SRCUPD, and ++ZAP modification control statements are not allowed in function SYSMOD packages.

  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 FXY1040 that is dependent on function GXY1000. The elements and JCL input data are members of three unloaded partitioned data sets on a tape created using the relative file technique.

```
++FUNCTION(FXY1040) FILES(3).
++VER(Z038) FMID(GXY1000).
++JCLIN RELFILE(1).
++MOD(IFBMOD01) RELFILE(2) DISTLIB(AOSFB).
++MOD(IFBMOD02) RELFILE(2) DISTLIB(AOSFB).
++MAC(IFBMAC01) RELFILE(3) DISTLIB(IFBMACS).
```

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(fmid)
      [THEN]
      REQ(req[,req]...)
      .
```

IF OPERANDS

++ must be in columns 1 and 2

FMID(fmid)
: specifies, as a condition, the SYSMOD-ID of a function SYSMOD, fmid. 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(req[,req]...)
specifies, as an action, one or more SYSMODs that are requisites, req, 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 identified by the FMID operand is processed.

IF PROGRAMMING CONSIDERATIONS

- The operands must be specified in the order shown in the syntax.
- 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 FXY1040. If function FXY1050 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 FXY1050 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 FXY1050 is processed at a future time. When function FXY1050 is applied, PTF UZ00005 is considered to be an unsatisfied conditional requisite that must be applied concurrently, if it has not already been applied.

```
++PTF(UZ00004).
++VER(Z038) FMID(FXY1040).
++IF FMID(FXY1050) THEN REQ(UZ00005).
++MOD(IFBMOD01) DISTLIB(AOSFB).
++MACUPD(IFBMAC01) DISTLIB(IFBMACS).
```

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 RELFILE tape 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 must match the SYSMOD's sysmod-id; for example, the JCLIN for "++FUNCTION(ESY1400) ...." would be supplied in member, "ESY1400", of the file identified by the ++JCLIN RELFILE keyword.

<u>Note:</u> This keyword is optional and mutually exclusive with TXLIB.

TXLIB(ddname)
: specifies the ddname of a partitioned data set which contains the JCL input data for the SYSMOD. The member of the TXLIB data set that contains the JCL input data must match the SYSMOD's sysmod-id; for example, the JCLIN for "++PTF(UZ11111) ...." would be supplied in member, "UZ11111", of the data set identified by the ++JCLIN TXLIB keyword.

<u>Note:</u> 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

- 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.
- Processing the JCL data can be avoided by specifying the NOJCLIN operand on the APPLY control statement.
- See the Programming Considerations of the JCLIN Control Statement in Chapter 5 for examples of JCL input data.

JCLIN EXAMPLE

For function FXY1050, the JCL input data, an object module, IFBMOD01, and a macro, IFBMAC01, are located in a separate text library named LIB1501. The JCL data contains an assembler program named ALTASM.

```
   ++FUNCTION (FXY1050).
   ++VER(Z038) FMID(GXY1000) VERSION(FXY1040) .
   ++JCLIN ASM(PGM=ALTASM) TXLIB(LIB1501).
   ++MOD(IFBMOD01) DISTLIB(AOSFB) TXLIB(LIB1501).
   ++MAC(IFBMAC01) DISTLIB(IFBMACS) TXLIB(LIB1501).
```

The following DD statement is needed at APPLY/ACCEPT time to define the TXLIB data set:

```
   //LIB1501 DD DSN=...
```

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]...)]
      [PREFIX(prefix[,prefix]...)]
      [RELFILE(number) | TXLIB(ddname)]
      [RMID(sysmodid)]
      [SSI(code)]
      [SYSLIB(ddname)]
      [UMID(sysmodid[,sysmodid]...)]
      [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 modules to be assembled in addition to those modules named as GENASM subentries in the CDS MAC entry. The source for the assemblies is found as a matching CDS ASSEM entry, (A)CDS SOURCE entry or member in the DISTSRC/ASMLIB dataset. The first match (in the order indicated above) is used.

   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.

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.

MALIAS(alias[alias]...)
specifies the alias names for the macro in both the target system and distribution libraries.

PREFIX(prefix[,prefix]...)]
specifies the first characters (prefix) of the names of modules to be assembled in addition to those modules named as GENASM subentries in the CDS MAC entry. The prefix values must be seven (7) characters or less.

The full module names are determined by comparing the prefix with the (A)CDS MOD entry names.

The sources for the assemblies are found as a CDS ASSEM entry, (A)CDS SOURCE entry or member in the DISTSRC/ASMLIB dataset matching the module names determined above. The first matching source (in the order indicated) is used.

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 RELFILE tape 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 RMID of the macro. This keyword may be used only in function SYSMODs and indicates the PTF service level of the MACRO (the last PTF to replace the macro.)

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.

Note: 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.

Note: This keyword is mutually exclusive with the RELFILE keyword.

UMID(sysmodid[,sysmodid]...)
specifies the UMID(s) of the macro. This keyword may be used only in function SYSMODs and indicates the PTF service level of the MACRO (the set of PTFs which have updated the macro since it was last replaced).

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

- 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.

- 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.

- If the macro resides in a target system library (rather than the SMPMTS), the target system library should be included in the SYSLIB DD concatenation for assemblies at APPLY. See "SMP Cataloged Procedure" in Chapter 3 and "SYSLIB Data Set" in Chapter 7 for a discussion of the SYSLIB concatenation.

- The operating system library for the macro is determined either from the SYSLIB or DISTLIB keywords (see "Processing Source and Macro Modifications" in Chapter 2.)

MAC EXAMPLE

The macro replacement for IFBMAC01 does not follow in the input stream. The replacement resides in the text library SYS1.REPLACE.

```
   ++MAC(IFBMAC01) TXLIB(REPLACE).
```

In this example, the following DD statement is needed at APPLY and ACCEPT time to define the TXLIB data set=

```
   //REPLACE DD DSN=...
```

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]...)]
      [PREFIX(prefix[,prefix]...)]
      [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 modules to be assembled in addition to those modules named as GENASM subentries in the CDS MAC entry. The source for the assemblies is found as a matching CDS ASSEM entry, (A)CDS SOURCE entry or member in the DISTSRC/ASMLIB dataset. The first match (in the order indicated above) is used.

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.

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.

PREFIX(prefix[,prefix]...)
specifies the first characters (prefix) of the names of modules to be assembled in addition to those modules named as GENASM subentries in the CDS MAC entry. The prefix values must be seven characters or less.

The full module names are determined by comparing the prefix with (A)CDS MOD entry names.

The source for the assemblies is found as a CDS ASSEM entry, (A)CDS SOURCE entry or member in the DISTSRC/ASMLIB dataset matching the above full module names. The first match (in the order indicated above) is used.

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

- 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.

- The operating system library for the macro is determined either from the SYSLIB or DISTLIB keywords (see "Processing Source and Macro Modifications" in Chapter 2.)

- If the macro resides in a operating system library (rather than the SMPMTS), the operating system library should be included in the SYSLIB DD concatenation for assemblies at APPLY. See "SMP Cataloged Procedure" in Chapter 3 and "SYSLIB Data Set" in Chapter 7 for a discussion of the SYSLIB concatenation.

MACUPD EXAMPLE

The macro IFBMAC02 is in the IFBMACS distribution library and is to be updated. The module IFBSRC01 must be reassembled when IFBMAC02 is modified. IFBSRC01 is a source module in the distribution library SYS1.IFBSRC and a module in the distribution library SYS1.AOS23.

```
++MACUPD(IFBMAC02) DISTLIB(IFBMACS) ASSEM(IFBSRC01)
            DISTSRC(IFBSRC) DISTMOD(AOS23).
```

In this example, DD statements are required at APPLY time to define the operating system libraries for the macro and the load module to be updated as a result of the assembly. For example, if the modules in SYS1.AOS23 (the assembled module's distribution library) were copied to SYS1.LINKLIB and and the source modules in SYS1.IFBSRC (the source element's distribution library) were copied to SYS1.CHGLIB, then the following DD cards are needed:

```
//LINKLIB DD DSN=SYS1.LINKLIB....
//CHGLIB DD DSN=SYS1.CHGLIB....
```

In this example, the following DD statements are needed at ACCEPT time to define the distribution library data sets:

```
//IFBMACS DD DSN=SYS1.IFBMACS.... (macro's DLIB)
//IFBSRC DD DSN=SYS1.IFBSRC.... (source DLIB for assembly)
//AOS23 DD DSN=SYS1.AOS23.... (DLIB for module assembled)
```

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)]
      [UMID(sysmodid[,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(alias)
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 attributes can be specified:

AC=1	REFR
ALIGN2	RENT
DC	REUS
NE	SCTR
OVLY	STD

See "Load Module Attributes and Link Edit Parameters" in the "APPLY Processing" and "ACCEPT Processina" sections of Chapter 2 for a discussion of the use of the LEPARM keyword.

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 creates 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 RELFILE tape 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 RMID of the module. This keyword may be used only in function SYSMODs and indicates the PTF service level of the MOD (the last PTF to replace the module.)

UMID(sysmodid[,sysmodid]...)
specifies the UMID(s) of the module. This keyword may be used only in function SYSMODs and indicates the PTF service level of the MOD (the set of PTFs which have updated the module since it was last replaced).

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

- 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.
- 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.
- 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 IFBMOD05 is a new module that is to be placed in the distribution library SYS1.AOSFB and is to be link edited with the existing load module IEEFRQ in the target system library SYS1.LINKLIB.

```
   ++MOD(IFBMOD05) DISTLIB(AOSFB) LMOD(IEEFRQ).
```

The following DO statement is needed at APPLY time to define the operating system load module library:

```
   //LINKLIB DD DSN=SYS1.LINKLIB....
```

The following DD statement is needed at ACCEPT time to define the module's distribution library:

```
   //AOSFB DD DSN=SYS1.AOSFB....
```

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

- 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.

- You should use the ++USERMOD modification control. statement to create modifications to IBM components rather than the ++PTF modification control statement.

- 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 which replaces module IFBMOD01 for function FXY1040. The prerequisite service SYSMOD for the module is PTF U200004. The APAR incident fixed by this PTF is OZ12345.

```
++PTF(UZ00006).
++VER(Z038) FMID(FXY1040) PRE(U200004)
    SUP(AZ12345).
++MOD(IFBMOD01) DISTLIB(AOSFB).
```

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)]
      [DISTMODtddname) | DISTOBJ(ddname)]
      [RELFILE(number)]
      [RMID(sysmodid)]
      [SSI(code)]
      [SYSLIB(ddname)]
      [TXLIB(ddname)]
      [UMID(sysmodid[,sysmodid]...)]
      [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 RELFILE-tape 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 RMID of the source. This keyword may be used only in function SYSMODs and indicates the PTF service level of the SRC (the last PTF to replace the source.)

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.

UMID(sysmodid[,sysmodid]...)
: specifies the UMID(s) of the source. This keyword may be used only in function SYSMODs and indicates the PTF service level of the SRC (the sat of PTFs which have updated the source since it was last replaced).

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

The operating system library for the source is determined either from the SYSLIB or DISTLIB keywords (see "Processing Source and Macro Modifications" in Chapter 2.)

SRC EXAMPLE

A replacement for the source module IFBSRC01 is in a partitioned data set referenced by the ddname REPLACE. The distribution library for the source module is SYS1.IFBSRC; SYS1.AOS23 is the distribution library for the module, IFBSRC01, resulting from the assembly of the source, IFBSRC01.

```
++SRC(IFBSRC01) TXLIB(REPLACE) DISTLIB(IFBSRC)
    DISTMOD(AOS23).
```

The following DD statement is needed at APPLY and ACCEPT time to define the TXLIB data set:

```
//REPLACE DD DSN=...
```

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

The operating system library for the source is determined either from the SYSLIB or DISTLIB keywords (see "Processing Source and Macro Modifications" in Chapter 2.)

SRCUPD EXAMPLE

The source module IFBSRC02 to be updated resides on the target system library SYS1.SRCLIB and distribution library SYS1.IFBSRC.

```
++SRCUPD(IFBSRC02) SYSLIB(SRCLIB) DISTLIB(IFBSRC).
```

The following DD statement is needed at APPLY time

```
//SRCLIB DD DSN=SYS1.SRCLIB....
```

The following DD statement is needed at ACCEPT time:

```
//IFBSRC DD DSN=SYS1.IFBSRC....
```

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

- You must define the seven 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; letters 'L' through 'T' and numbers '0' through '9' are available for user modifications. 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.

- 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.

- Subsequent replacements to elements modified by your modification from non-function SYSMODs cannot occur unless you explicitly allow them by bypassing MODID checking. Installation of new function-type SYSMODs may overlay your user modifications; a warning message will be issued when SMP detects that a user modification is being overlaid by the installation of a function.

- A user modification is only accepted into the distribution libraries if the USERMODS keyword is specified on the ACCEPT control statement.

- 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, IFBSRC02, which is owned by function SYSMOD FXY1040, is to be modified. Your modification requires a service level provided PTF UZ00007; 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(FXY1040) PRE(UZ00007).
++SRCUPD(IFBSRC02) DISTLIB(IFBSRC).
```

THE VERIFY (++VER) MODIFICATION CONTROL STATEMENT

The ++VER modification control statement describes the environment required receive, apply and accept a SYSMOD. 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[,sysmodid3...)]
    [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 the functional prerequisite for the SYSMOD. FMID must be specified for dependent-level functions and all non-function SYSMODs.

When specified on the ++VER statement for a ++FUNCTION SYSMOD, FMID defines the function as a dependent-level function. In this case, FMID indicates that the elements supplied by the dependent-level function SYSMOD are functionally superior to the base-level function named.

When specified on the ++VER statement for a non-function SYSMOD, FMID indicates the functional level of all elements in the SYSMOD.

Non-function SYSMODs are not received unless the PTS SYSTEM entry contains an FMID subentry which matches the names FMID.

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 values 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.

VER PROGRAMMING CONSIDERATIONS

- 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.
- 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.
- With one exception, 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. The exception being the SYSMOD-IDs that are specified in the VERSION operand list, these SYSMOD-IDs may be specified in any one of the other operand lists with the exception of FMID.

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(alias)
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

- An EXPAND control statement in linkage editor format can be placed within IMASPZAP input to allow lengthening of control sections. The EXPAND statement must follow the NAME statement. Refer to OS/VS Linkage Editor and Loader for the syntax and description of the EXPAND statement.

- 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.

- Expand-type IMASPZAP processing cannot be performed against a non-editable (NE) module.

- 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.

- 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.

- 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.

  It is not recommended that you use IMASPZAP to modify assembled modules. An assembled module modified by IMASPZAP does not cause updating of the distribution library during ACCEPT processing, therefore, a subsequently SYSGEN'd system will not contain the IMASPZAP modification.

  A more satisfactory method of updating assembled modules is to update the macros which generate the modules.

- 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.

- 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 IFBMOD05 is to be changed via IMASPZAP. The module is owned by function SYSMOD FXY1050 and the last SYSMOD which replaced the module was UZ00008 (found as the current RMID subentry value in CDS MOD entry). The module is in load module IEEFRQ. You are creating the modification and assigning a SYSMOD-ID of MY00006.

```
++USERMOD(MY00006).
++VER(Z038) FMID(FXY1050) PRE(UZ00008).
++ZAP(IFBMOD05) DISTLIB(AOSFB).
  NAME IEEFRQ IFBMOD05
  VER 13F6 47E0A138
  REP 13F6 4770A14C
```

CHAPTER 7: SMP DATASETS

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)
- SMPADDIN (contains ADDIN control statements)
- 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)
- SMPPUNCH (contains output from UNLOAD function)
- SMPRPT (Report Output Data Set)
- SMPSCDS (Saved Control Data Set)
- SMPSTS (Source Temporary Store Data Set)
- SMPTLIB (RELFILE tape libraries)
- 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)

- SYSUT4, (retry only)

- 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.

Note: Any dataset restrictions that apply to Utility programs invoked by SMP must be adhered to.

DATA SETS REQUIRED

Figures 34 and 35 provides a summary of the data sets required by each SMP control statement. The following list explains the meaning of each number used in Figures 34 and 35:

- 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 sat 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 MIS 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.
- 16 - Required when ADDINPUT keyword specified on UNLOAD control statement.
- 17 - Required when RETRY is specified on the ACCEPT, APPLY, and RESTORE control statements.
- 18 - Required for REJECT with the PURGE option.

DATA SET NAMES	ACCEPT	APPLY	JCLIN	LIST	LOG	RECEIVE	REJECT	RESETRC	RESTORE	UCLIN	UNLOAD
lklib	2	2									
txlib	3	3									
SMPACDS	1			15			18		1	14	14
SMPACRQ	1			15						14	
SMPADDIN											16
SMPCDS	6	1	1	15					1	14	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					
SMPPUNCH											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		

Figure 34 Data Set Requirements

DATA SET NAMES	ACCEPT	APPLY	JCLIN	LIST	LOG	RECEIVE	REJECT	RESETRC	RESTORE	UCLIN	UNLOAD
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		
SYSUT4	17	17							17		
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 6 for information. A DD statement specifying either SYSPRINT of the substitute ddname must be supplied as described for the SYSPRINT data set above.

Figure 35 Data Set Requirements - Continued

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: Partitioned

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: Partitioned; 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 nucleus backup identifier (NUCID), system release (SREL) and data set identifier (CDSID).

The ACDS directories are maintained in-storage to enhance the performance of ACCEPT, UCLIN and LIST functions. A DIS(NO) option is provided for each of these functions to disable the in-storage maintenance of the directory for those systems which are storage constrained or when the function to be performed is so trivial that the overhead of bringing the directories into storage exceeds the performance benefits of in-storage maintenance.

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.

SMPACRQ DATA SET

Ddname: SMPACRQ

Acronym: ACRQ

Description: Alternate Conditional Requisite Queue Data Set

Attributes: Partitioned; LRECL=80,BLKSIZE=multiple of 80,DISP=OLD

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.

SMPADDIN DATA SET

Ddname: SMPADDIN

Acronym: None

Description: Contains SMP UNLOAD control statements

Attributes: Sequential; LRECL=80,BLKSIZE=multiple of 80

Device: Card, tape, direct access or terminal device

Information: The SMPADDIN dataset is used to contain control statements that are used during the UNLOAD processing. If the ADDIN option is specified on the UNLOAD function control statement, then the SMPADDIN DD statement must be present.

SMPCDS DATA SET

Ddname: SMPCDS

Acronym: CDS

Data Set: Control Data Set

Attributes: Partitioned; LRECL=80,BLKSIZE=multiple of 80,DISP=OLD

Device: Direct access only

Information: This data set contains information regarding the structure of the operating system in terms of macros, modules, assemblies, load modules, libraries copied at system generation time, source modules, and SYSMODs. 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 nucleus backup identifier (NUCID), system release (SREL) and data set identifier (CDSID).

The CDS is initialized with the names of the elements making up the operating system by copying the ACDS to the CDS.

The CDS is initialized with information regarding the structure of the operating system by JCLIN processing of the job stream which built the system.

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 Updating of and examination of the CDS should be done only through the use of SMP.

The CDS directories are maintained in-storage to enhance the performance of APPLY, RESTORE, UCLIN, JCLIN and LIST functions. A DIS(NO) option is provided for each of these functions to disable the in-storage maintenance of the directory for those systems which are storage constrained or when the function to be performed is so trivial that the overhead of bringing the directories into storage exceeds the performance benefits of in-storage maintenance.

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,DISP=OLD

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 referance the SYSMOD entries that contained the ++IF modification control statements in which they wore 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 E0

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

Descriptions LIST Output Data Set

Attributes: Sequential; BLKSIZE=multiple of 121,LRECL=121,RECFM=FB

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: MIS

Description Macro Temporary Store Data Set

Attributes: Partitioned; LRECL=80,BLKSIZE=multiple of 80,DISP=OLD

Device: Direct access only

Information: For APPLY processing, the SMPMTS data set must be allocated with sufficient space to hold the entire set of SYSGEN macros as well as any other macros that do not reside in a system library of the operating system. This is due to the restructuring of the system into functional packages where the complete set of SYSGEN macros are contained in the set of functional packages.

The SMPMTS data set must be included as the first library in the SYSLIB DD concatenation for APPLY and RESTORE processing. The SMPMTS data set must not be included in the SYSLIB DD concatenation for ACCEPT processing. See "SMP Cataloged Procedure" in Chapter 3 for a further discussing of SYSLIB concatenation requirements.

SMPOUT DATA SET

Ddname: SMPOUT

Acronym: None

Description: Message Output Data Sat

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 Sat

Attributes: Partitioned; LRECL=80,BLKSIZE=multiple of 80,DISP=OLD

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 facilities and must contain at least one system release (SREL) subentry, any number of function modification identifier (FMID) subentries and and a DSSPACE subentry. The PTS entries can be modified by UCLIN facilities.

SMPPUNCH DATA SET

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 the output of the UNLOAD function which consists of UCL control statements representing the contents of the CDS or ACDS data sets.

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,DISP=OLD

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,DISP=OLD

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: RELFILE 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=CNEW,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

Device: Direct Access only

Information: This data set is used as temporary storage for object module text supplied by a SYSMOD, object text generated by assemblies and AMASPZAP control cards following ++ZAP modification control statements.

The object modules assembled during SMP processing are kept on this dataset until SMP has successfully APPLIED or ACCEPTED the SYSMOD(s) which caused the assemblies. Therefore, if the dataset is allocated as permanent, these objects will be available for the REUSE facility following a failure during APPLY or ACCEPT.

It is recommended that this dataset be scratched and allocated new before each "normal" SMP run. If this "normal" SMP fails after SMP has assembled a large number of modules, and you wish to rerun without redoing the assemblies, do not scratch WRK3 and code REUSE on the APPLY or ACCEPT statement.

If this dataset is kept, SMP will scratch objects when the SYSMOD causing an assembly is successfully processed; you must, however, reclaim space by compressing the dataset.

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 IMASPZAP and the linkage editor for ZAPS which contain EXPAND control statements. 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 block size of the data set must be compatible with the block size 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.

For APPLY and RESTORE the libraries consist of data sets concatenated in the following sequence:

- 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 MTS and target system libraries should _not_ be included.

The block size 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 5 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,BLKSIZE=multiple of 80,LRECL=80

Device: Direct access only

Information: Required only if RETRY is requested or defaulted on APPLY, ACCEPT, or RESTORE control statements.

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.

CONTROL DATA SET ENTRIES

This chapter describes the entries in the primary SMP data sets. This information can be used in the analysis of SMP listings and UCLIN functions.

DATASET ORGANIZATION

SMP uses partitioned (PDS) datasets; the entries in these datasets are actually PDS members. The member names are generally encoded and cannot easily be accessed by utilities which process PDS datasets. The data in the actual member portion of the datasets contains hexadecimal (non-printable) codes. LIST and UCLIN facilities are provided to display and update these entries.

In addition to the data stored in the actual member portion of the dataset, SMP utilizes the user data portion of the directory entries to record certain information. The use of SPF or other dataset utilities to examine or modify the SMP datasets may result in invalid directory user data which can cause unpredictable SMP processing.

DEFINITION OF TERMS:

entry	The term entry refers to a member of an SMP dataset. With the exception of the MCS entry in the SMPPTS dataset, these member names are encoded and cannot be easily accessed by utilities other than SMP. SYSMOD and MACRO entries are examples of the types of entries maintained in the SMPCDS dataset.
sub-entry	A sub-entry is a field within an entry. Each sub-entry has associated with it a type and a value. Multiple occurrences of the same sub-entry type may exist in an entry each with a different value. For example, the modules supplied by a PTF are saved as "MOD" type sub-entries within the PTF's SYSMOD entry. Multiple occurrences of the same sub-entry type and value pair do not exist. For example, there are not two "MOD" type sub-entries with the same module name.
Indicator	An indicator is a field in an SMP dataset entry that does not have a data value associated with it. An example of an indicator is the APP indicator in the SMPCDS SYSMOD entry. An indicator is either "on" or "off".

The descriptions in this chapter are arranged by data set and entry type as follows:

DATA SET	Entry Type	Page
CDS & ACDS	ASSEM *	315
	DLIB *	315
	LMOD *	315
	MAC	316
	MOD	317
	SRC	318
	SYSTEM	321
	SYSMOD	319
CRQ & ACRQ	FMID	323
	SYSMOD	323
PTS	SYSMOD	328
	SYSTEM	325
SCDS	SYSMOD	331
STS	SRC	333
MIS	MAC	333

* CDS Only

SMPCDS entries basically control the SMP apply function and contain information concerning SYSMODs and elements which have been applied.

SMPACDS entries basically control the SMP accept function and contain information concerning SYSMODs and elements which have been accepted.

SMPCDS ASSEMBLER ENTRIES

These entries contain assembler source text found in JCLIN processing. This source is assembled during apply processing of macro modifications. ASSEMBLER entries exist only on the CDS.

There are no indicators or sub-entries within an ASSEMBLER entry.

SMPCDS DLIB ENTRIES

DLIB entries are created during JCLIN processing for distribution libraries which are totally copied to a target system library. DLIB entries exist only on the CDS.

DLIB entries contain one sub-entry type:

- SYSLIB sub-entries: Specify the DDNAMEs of the target system libraries to which the distribution library was totally copied. There can be at most two SYSLIB sub-entries.

SMPCDS LMOD ENTRIES

LMOD entries are created during APPLY and JCLIN processing and contain the data required to properly link edit modules into target system load modules. LMOD entries exist only on the CDS.

In addition to the sub-entries and indicators listed below, LMOD entries contain the linkage editor control statements (with the exception of INCLUDES) used to link edit the load modules.

LMOD entries contain the following sub-entry types and indicators:

- LASTUPD sub-entry: Identifies the cause of the last change made to this entry.

- LASTUPDTYPE sub-entry: Identifies the last type of update made to this entry.

- SYSLIB sub-entry: Specifies the DDNAME of the target system library that contains the load module. There can be at most two SYSLIB sub-entries.

- AC=1 indicator: When this indicator is set, the AC=1 parameter is passed to the linkage editor program when the load module is link edited.

- ALIGN2 indicator: When this indicator is set, the ALIGN2 parameter is passed to the linkage editor program when the load module is link edited.

- COPY indicator: Indicates the load module was copied at SYSGEN time and implies that the load module is made up on only one CSECT. When this indicator is set, SMP will not include the earlier version of the load module when link edits are performed.

- DC indicator: When this indicator is set, the DC parameter is passed to the linkage editor program when the load module is link edited.

- NE indicator: When this indicator is set, the NE parameter is passed to the linkage editor program when the load module is link edited.

- OVLY indicator: When this indicator is set, the OVLY parameter is passed to the linkage editor program when the load module is link edited.

- REFR indicator: When this indicator is set, the REFR parameter is passed to the linkage editor program when the load module is link edited.

- RENT indicator: When this indicator is set, the RENT parameter is passed to the linkage editor program when the load module is link edited.

- REUS indicator: When this indicator is set, the REUS parameter is passed to the linkage editor program when the load module is link edited.

- SCTR indicator: When this indicator is set, the SCTR parameter is passed to the linkage editor program when the load module is link edited.

- STD indicator: Indicates that the load module is to be link edited with no particular attributes. When this indicator is set the parameters passed are those determined from LKEDPARMs in the PTS SYSTEM entry.

SMPACDS/SMPCDS MACRO ENTRIES

MACRO entries are created by APPLY and ACCEPT processing of ++MAC and ++MACUPD elements. They describe the macro elements installed on the distribution and target system libraries.

ACDS and CDS MACRO entries contain the following sub-entry types and indicators:

- DISTLIB sub-entry: Distribution library DDNAME.

- FMID sub-entry: Names the function-SYSMOD which owns the macro.

- GENASM sub-entries: Specifies the assemblies to be done when the macro is updated at APPLY. The source for these assemblies may be either SMPCDS ASSEM or SOURCE entries.

- MALIAS sub-entries: Specifies the alias name(s) of the macro in the distribution and target system libraries.

- RMID sub-entry: Names the last SYSMOD to replace the macro.

- SYSLIB sub-entries: DDNAMEs of the target system libraries for the macro.

- UMID sub-entries: Names the SYSMOD(s) which have updated the macro since the last replacement.

- LASTUPD sub-entry: Identifies the cause of the last change made to this entry.

- LASTUPDTYPE sub-entry: Identifies the last type of update made to this entry.

SMPACDS/SMPCDS MOD ENTRIES

MOD entries are created by APPLY and ACCEPT processing of ++MOD and ++ZAP elements. SMPCDS MOD entries may also be created by JCLIN processing. They describe the modules installed on the distribution and target system libraries.

The ACDS and CDS MOD entries contain the following indicators and sub-entry types:

- ASSEMBLE indicator: Indicates that this module should be assembled from ASSEM or source text.

- DALIAS sub-entries: Specify the alias names of the module in the distribution library and, for a copied module, in the target system library.

- DISTLIB sub-entry: DDNAME of module's distribution library.

- FMID sub-entry: Names the Function SYSMOD which owns the module.

- LMOD sub-entries: Specify the load modules into which the module is link edited.

- RMID sub-entry: Names the last SYSMOD to replace the module.

- RMIDASM indicator: Indicates that the last replacement to the module was done by a SYSMOD supplying a MACRO or SOURCE element which required the assembly of the module.

  Note: this indicator will be set in a MOD entry for modules affected by SYSMODs which supply MACRO or SOURCE elements _and_ a corresponding assembled MOD (even though SMP may actually choose the MOD and not do the assembly). As such, this is used to allow installation of subsequent source/macro modifications which neither supply the MOD nor prereq the previous modification to the assembled MOD.

- UMID sub-entries: Name the SYSMODs which have updated (ZAPed) the module since it was last replaced.
- LASTUPD sub-entry: Identifies the cause of the last change made to this entry.
- LASTUPDTYPE sub-entry: Identifies the last type of update made to this entry.

The following linkage editor indicators may be set in the SMPACDS. Their meaning is exactly the same as the corresponding indicators in the CDS LMOD entries and are used during ACCEPT processing when link editing the module to the distribution libraries:

- AC=1 indicator
- ALIGN2 indicator
- DC indicator
- NE indicator
- OVLY indicator
- REFR indicator
- RENT indicator
- REUS indicator
- SCTR indicator

SMPACDS/SMPCDS SRC ENTRIES

SRC entries are created by APPLY and ACCEPT processing of ++SRC and ++SRCUPD elements. They describe the source elements installed on the distribution and target system libraries.

ACDS and CDS SRC entries contain the following indicators and sub-entry types:

- DISTLIB sub-entry: DDNAME of source's distribution library.
- FMID sub-entry: Names the function-SYSMOD which owns the source.
- RMID sub-entry: Names the last SYSMOD to replace the source.
- SYSLIB sub-entry: DDNAME of the target system library for the source.
- UMID sub-entries: Name the SYSMODs which have updated the source since the last replacement.
- LASTUPD sub-entry: Identifies the cause of the last change made to this entry.
- LASTUPDTYPE sub-entry: Identifies the last type of update made to this entry.

SMPACDS/SMPCDS SYSMOD ENTRIES

SYSMOD entries describe SYSMODs which have been installed in the distribution and target system libraries. These entries are used to track the status of a SYSMOD and maintain a historical record.

Date fields in these entries are Julian dates in the form, "yyddd".

Time fields in these entries are in the form, "hh:mm:ss".

ACDS and CDS SYSMOD entries have one of the following four TYPE indicators set:

- FUNCTION
- PTF
- APAR
- USERMOD

The following indicators and sub-entry types appear in the SYSMOD entry;

- ACCDATE sub-entry: Date that the SYSMOD was accepted.
- ACCEPT indicator: Indicates that the SYSMOD has been accepted.
- ACCTIME sub-entry: Specifies the time that the SYSMOD was accepted.
- APPDATE sub-entry: Julian data that the SYSMOD was applied.
- APPLY indicator: Indicates that the SYSMOD has been applied.
- APPTIME sub-entry: Specifies the time at which the SYSMOD was applied.
- ASSEM sub-entries: Names of ASSEM or SRC entries that were assembled in processing the SYSMOD or were specified in the ASSEM keyword of a ++MAC or ++MACUPD modification control statement.
- BYPASS indicator: Indicates that requisite or MODID checking error conditions were bypassed in order to process this SYSMOD.
- DELBY sub-entries: Names the SYSMOD(s) that deleted this SYSMOD. These sub-entry types are only present for deleted function SYSMODs.
- DELETE sub-entries: Name of SYSMOD(s) that were deleted by this SYSMOD.
- ERROR indicator: When this indicator is set, the SYSMOD is considered to have been unsuccessfully processed by apply, accept or restore.
- FMID sub-entry: Names the function upon which the SYSMOD is dependent. For base level functions (which are not dependent upon any other function), this value will match the SYSMOD's Id.

- JCLIN indicator: Indicates that the SYSMOD contains inline JCLIN.

- LASTSUP sub-entry: Names the last SYSMOD which superseded this SYSMOD.

- LASTUPD sub-entry: Identifies the cause of the last change to this entry.

- LASTUPDTYPE sub-entry: Identifies the last type of update made to this entry

- MAC sub-entries: Names of macro replacememnts (++MACs) supplied by this SYSMOD.

- MACUPD sub-entries: Names of the macro updates (++MACUPDs) supplied by this SYSMOD.

- MOD sub-entries: Names of the modules (++MODs) supplied by this SYSMOD.

- NPRE sub-entries: Names of SYSMODs which are negative prerequisites (NPREs) of this SYSMOD.

- PRE sub-entries: Names of SYSMODs which are prerequisites (PREs) of this SYSMOD.

- RECDATE sub-entry: Date that the SYSMOD was received.

- RECTIME sub-entry: Time at which the SYSMOD was received.

- REGEN indicator: 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.

- REQ sub-entries: Names of SYSMODs which are requisites (REQs) of this SYSMOD.

- RESDATE sub-entry: Date this SYSMOD was restored.

- RESTIME sub-entry: Time this SYSMOD was restored.

- RESTORE indicator: If this indicator is set, the SYSMOD is considered to have had a RESTORE operation attempted.

- RMAC sub-entries: Name macros in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- RMACUPD sub-entries: Name macro updates in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- RMOD sub-entries: Name modules in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- RSRC sub-entries: Name source elements in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- RSRCUPD sub-entries: Name source updates in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- RSZAP sub-entries: Name the module update elements, ZAPs, in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- RXZAP sub-entries: Name the expand-ZAP updates in this SYSMOD which have been potentially regressed (see APPLY Processing Chapter 2) by another SYSMOD.

- SRC sub-entries: Name the source replacements (++SRC) supplied by this SYSMOD.

- SRCUPD sub-entries: Name the source updates (++SRCUPD) supplied by this SYSMOD.

- SUPBY sub-entries: Name the SYSMODs which have superseded this SYSMOD.

- SUPING sub-entries: Name those SYSMODs superseded (SUP) by this SYSMOD.

- SZAP sub-entries: Name the module update elements (++ZAPs) supplied by this SYSMOD.

  Note: ZAPs supplied with the EXPAND control statement are considered "expand-ZAPS" and show up as XZAP subentries.

- UCLDATE sub-entry: Date that the SYSMOD was updated by UCLIN.

- UCLTIME sub-entry: Time when UCL updated the SYSMOD.

- VERNUM sub-entry: The 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.

- VERSION sub-entries: Name the function SYSMODs that are functionally inferior to this SYSMOD.

- XZAP sub-entries: Name the modules updated and expanded by this SYSMOD.

SMPACDS/SMPCDS SYSTEM ENTRIES

The SYSTEM entry is used by APPLY, RESTORE and ACCEPT processing for system verification and contains some indicators which control SMP's processing.

- CDSID sub-entry: A character string which identifies the control dataset.

  The CDSID value from the CDS is put in the PTS SYSMOD entry when a SYSMOD is applied to indicate those systems on which a SYSMOD has been applied.

  The CDSID value from the ACDS is put in the PTS SYSMOD entry when a SYSMOD is accepted to indicate those systems on which a SYSMOD has been accepted.

- NUCID sub-entry (CDS): A 1-digit number appended to the nucleus program name IEANUCO to form the name of the nucleus load module saved during APPLY processing. This sub-entry may appear in the ACDS but is not used.

- PEMAX sub-entry: A number from 1 to 9999 that defines the maximum number of subentries that can be present in an entry on the ACDS or CDS respectively. If this subentry is not present, a default value of 500 is used.

- SAVEMTS indicator (CDS only): When this indicator is set, the macros in the MIS data set are not deleted by ACCEPT processing. The default for this indicator is "reset". 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 indicator (CDS only): When this indicator is set, the modules in the STS data set are not deleted by ACCEPT processing. The default for this indicator is "reset". 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.

- RETRYDDN sub-entry: names the DDNAMEs of the datasets for which out-of-space retry is to be attempted. The value, 'ALL', causes RETRY to be attempted for utility failures on any 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.

  Unlike the normal SMP compress (via COMPRESS keyword), compress recovery done as the result of coding RETRYDDN(ALL) will attempt to compress any candidate including SYS1.LINKLIB.

- SREL sub-entry: System release identifier. Only one system release may appear in a CDS or ACDS SYSTEM entry.

ACRQ and CRQ entries contain data saved from ++IF conditional requisite statements during accept and apply processing respectively. They are used to determine functionally-conditional requisites when function SYSMODs are installed.

SMPACRQ/SMPCRQ FMID ENTRIES

There is an FMID entry for every function named as an FMID in a ++IF conditional requisite statement.

FMID entries contain only one subentry type, which is referred to as a "SYSMOD" sub-entry for UCL updates and is shown as a "CAUSER" in ACRQ/CRQ listings. These subentries name the SYSMODs which supplied ++IF conditional requisites for the function. The actual data from these ++IF statements is found in the ACRQ/CRQ SYSMOD entry for the SYSMOD which supplied the conditional requisites.

SMPACRQ/SMPCRQ SYSMOD ENTRIES

These entries contain the ++IF conditional requisite information supplied by a SYSMOD. There is a SYSMOD entry for every SYSMOD that supplied ++IF conditional requisite statements.

SYSMOD entries contain FMID and associated REQ subentries. The FMID subentry names a function-type SYSMOD and the associated REQ subentry(s) name the requisite SYSMODs for the particular functional environment. The FMID subentries are shown as "ENV" (environment), and the REQ subentries are shown as "IFREQ"s in ACRQ/CRQ listings.

The following example illustrates the SMPCRQ entries created and/or updated for the PTF shown

```
++PTF(UR00000) .
++VER(Z038) FMID(ESY1400) .
++IF FMID(ESY1401) THEN REQ(UR00001,UR00002) .
++IF FMID(ESY1402) THEN REQ(UR00003,UR00004) .
```

- Two CRQ FMID entries, ESY1401 and ESY1402, will be created (or added to). These entries will contain the name of the PIP, UR00000, as a subentry.
- A CRQ SYSMOD entry, UR00000, will be created with:
  - FMID subentry ESY1401 containing requisites UR00001 and UR00002.
  - FMID subentry ESY1402 containing requisites UR00003 and UR00004.

SMPPTS SYSTEM ENTRIES

The PTS SYSTEM entry is created by the user using UCLIN. This entry contains data which controls not only RECEIVE but also APPLY, ACCEPT and RESTORE processing.

The PTS SYSTEM entry contains the following indicators and sub-entries:

- ASMNAME sub-entry: The name of the program to be invoked by SMP to perform assemblies. The default program is "ASMBLR".

- ASMPARM sub-entry: The character string to be passed as parameters to the program invoked by SMP to perform assemblies. The default character string passed is "XREF,NOLOAD,DECK".

- ASMPRINT sub-entry: The DDNAME for the output listing data set produced by the assembler program. The default DDNAME is "SYSPRINT".

- ASMRC sub-entry: 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 default value is "4".

- COMPNAME sub-entry: The name of the program to be invoked by SMP to perform the PDS compress function. The default program name is "IEBCOPY".

- COMPPARM sub-entry: The character string to be passed as parameters to the program invoked by SMP to perform the PDS compress function. There is no default set of parameters; if COMPPARM is not present, no parameters are passed.

- COMPPRINT sub-entry:

  The ddname for the output listing data set produced by the PDS compress program. The default DDNAME is "SYSPRINT".

- COMPRC sub-entry: 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 default value is "0".

- COPYNAME sub-entry: The name of the program to be invoked by SMP to perform the PDS copy and load functions. The default program name is "IEBCOPY".

- COPYPARM sub-entry: The character string to be passed as parameters to the program invoked by SMP to perform the PDS copy and load functions. There is no default set of parameters.

- COPYPRINT sub-entry: The ddname for the output listing data set produced by the PDS copy and load program. The default DDNAME is "SYSPRINT".

- COPYRC sub-entry: 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 default value is "0".

  Note: IEBCOPY returns a code of 4 when it encounters I/O errors during the copying of members.

- DSPREFIX sub-entry: 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. There is no default DSPREFIX.

- DSSPACE sub-entry: The space parameters for data sets that are allocated during RECEIVE processing for library loading in terms of primary and secondary track allocation and number of directory blocks. There are no default space parameters; these must be set by the user when creating the PTS SYSTEM entry.

- FMID sub-entries: The functions which have been received (PTS).

- IOSUPNAME sub-entry: The name of the program to be invoked by SMP to perform the IEHIOSUP function. The default program is "IEHIOSUP".

- IOSUPPARM sub-entry: The character string to be passed as parameters to the program invoked by SMP to perform the IEHIOSUP function. There is no default set of parameters.

- IOSUPPRINT sub-entry: The ddname for the output listing data set produced by the IEHIOSUP program. The default DDNAME is "SYSPRINT".

- IOSUPRC sub-entry: 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 default value is "0".

- LKEDNAME sub-entry: The name of the program to be invoked by SMP to perform the linkage editor function. The default program is "IEWL".

- LKEDPARM sub-entry: The character string to be passed as parameters to the program invoked by SMP to perform the linkage editor functions. These parameters are always passed in addition to the link edit attributes determined during APPLY, ACCEPT or RESTORE processing.

  If no LKEDPARM sub-entry is present, "LET, LIST, XREF, NCAL" is used.

- LKEDPRINT sub-entry: The ddname for the output listing data set produced by the linkage editor program. The default DDNAME is "SYSPRINT".

- LKEDRC sub-entry: 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 SYSMODs for which the linkage editor program was invoked are terminated. The default value is "8".

- PAGELEN sub-entry: 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. The default value is 60.

- PEMAX sub-entry: A number from 1 to 9999 that defines the maximum number of subentries that can be present in an entry. The default value is 500.

  PURGE indicator: 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. When the PTS SYSTEM entry is created, the PURGE indicator is set. 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 indicator: When this indicator is set, any SYSMOD that is successfully processed by RESTORE is deleted from the PTS. When the PTS SYSTEM entry is created, the REJECT indicator is set. 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 sub-entry: The name of the program to be invoked by SMP4 to perform the RETRY recovery compress function for an out-of-space ABEND. If the RETRYNAME subentry is not present in the PTS system entry SMP4 invokes the program IEBCOPY to compress the out-of-space datasets.

- RETRYPARM sub-entry: specifies the character string to be passed as parameters to the program invoked by SMP for RETRY recovery compress functions. If the RETRYPARM subentry is not present in the PTS system entry, no parameters are passed.

- RETRYPRINT sub-entry: The DDNAME for the output listing data set produced by the RETRY recovery compress program. If the RETRYPRINT subentry is not present in the PTS system entry, then the ddname SYSPRINT is used.

- RETRYRC sub-entry: The return code value to be compared with the code returned from the RETRY recovery compress program. If the compress program return code is higher than the RETRYRC subentry value, then the result of the compress function is considered unsuccessful and RETRY is considered to have failed. In this case SMP is terminated. If the RETRYRC subentry is not present in the PTS system entry, then the value. 0, is used.

- SREL sub-entries: System release identifier. Multiple system release identifiers may appear in the PTS SYSTEM entry.

- UPDATNAME sub-entry: The name of the program to be invoked by SMP to perform the text update function. If the UPDATNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IEBUPDTE to perform the text update function.

- UPDATPARM sub-entry: The character string to be passed as parameters to the program invoked by SMP to perform the text update function. If the UPDATPARM subentry is not present in the PTS SYSTEM entry, SMP passes "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 it is appended to "MOD" or "REP" and passed to the text update program.

- UPDATPRINT sub-entry: The ddname for the output listing data set produced by the text update program. If the UPDATPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used.

- UPDATRC sub-entry: 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. 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 sub-entry: The name of the program to be invoked by SMP to perform the IMASPZAP service aid function. If the ZAPNAME subentry is not present in the PTS SYSTEM entry, SMP invokes the program IMASPZAP to perform the IMASPZAP function.

- ZAPPARM sub-entry: The character string to be passed as a parameter to the program invoked by SMP to perform the IMASPZAP function. If the ZAPPARM subentry is not present in the PTS SYSTEM entry, SMP does not pass any parameters to the IMASPZAP program.

- ZAPPRINT sub-entry: The ddname for the output listing data set produced by the IMASPZAP program. If the ZAPPRINT subentry is not present in the PTS SYSTEM entry, then the ddname SYSPRINT is used.

- ZAPRC sub-entry: 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. If the ZAPRC subentry is not present in the PTS SYSTEM entry, then the value of 4 is compared with the IMASPZAP program return code.

SMPPTS SYSMOD ENTRIES

PTS SYSMOD entries are created by RECEIVE processing and describe the SYSMODs which are potentially eligible for APPLY and ACCEPT processing. Associated with each SYSMOD entry is a MCS entry which contains the actual SMP modification control statements which compose the SYSMOD.

SYSMOD entries have one of the following four TYPE indicators set:

- FUNCTION - indicates ++FUNCTION SYSMOD

- PTF - indicates ++PTF SYSMOD
- APAR - indicates ++APAR SYSMOD
- USERMOD - indicates ++USERMOD SYSMOD

The following indicators and sub-entry types appear in the SYSMOD entry:

- ACC indicator: indicates that the SYSMOD has been accepted on one or more systems.
- ACCID sub-entries: CDSID values from the ACDS SYSTEM entries for systems on which the SYSMOD has been accepted. Thus, these sub-entries identify where the SYSMOD has been accepted.
- APP indicator: indicates that the SYSMOD has been applied to one or more systems.
- APPID sub-entries: CDSID values from the CDS SYSTEM entries for systems on which the SYSMOD has been applied. Thus, these sub-entries identify where the SYSMOD has been applied.
- BYP indicator: indicates that BYPASS(FMID) was used to receive this SYSMOD.
- DSPREFIX sub-entry: The user-specified high-level data set name qualifier for the files on the SMPTLIB volume for this SYSMOD.
- ERROR indicator: When this indicator is set, the SYSMOD is considered to have been unsuccessfully processed by RECEIVE.
- FMID VER(vernum) sub-entries: Names the functions to which the SYSMOD is applicable.
- JCLIN indicator: Indicates that the SYSMOD contains inline JCLIN.
- MAC sub-entries: Names of macro elements supplied by this SYSMOD.
- MACUPD sub-entries: Names of the macro updates supplied by this SYSMOD.
- MOD sub-entries: Names of the modules supplied by this SYSMOD.
- NPRE VER(vernum) sub-entries: Names of SYSMODs which are negative prerequisites of this SYSMOD (NPREs).
- PRE VER(vernum) sub-entries: Names of SYSMODs which are prerequisites (PREs) of this SYSMOD.
- RECDATE sub-entry: Julian date that the SYSMOD was received.
- RECTIME sub-entry: Time at which the SYSMOD was received.
- REQ VER(vernum) sub-entries: Names of SYSMODs which are requisites (REQs) of this SYSMOD.

- SRC sub-entries: Names of source replacement elements supplied by this SYSMOD.

- SRCUPD sub-entries: Names of source update elements supplied by this SYSMOD.

- SREL VER(vernum) sub-entries: Names the system releases to which the SYSMOD is applicable.

- SUP VER(vernum) sub-entries: Names those SYSMODs superseded by this SYSMOD.

- SZAP sub-entries: Names the module update elements (ZAPS)

- VERSION VER(vernum) sub-entries: Name the function SYSMODs that are functionally inferior to this SYSMOD.

Note: vernum is the 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 (SREL, FMID, PRE, REQ, NPRE, SUP and VERSION).

SMPSCDS entries contain data saved from the SMPCDS during APPLY processing (generally for inline JCLIN). A COPY is saved of any MOD, MAC, SRC, DLIB, ASSEM, or LMOD entry which was updated or deleted by applying the SYSMOD. There is an entry for each SYSMOD which modified element processing data in the CDS. This data is used by the RESTORE function to remove updates to the CDS caused by SYSMODs which are being restored.

SMPMTS

MTS entries contain macros from ++MAC and ++MACUPD elements processed during APPLY. The macro elements maintained on the MTS are those elements which have no target system library for APPLY.

- **These macro elements are deleted when the SYSMOD(s) supplying the MAC or MACUPDs are ACCEPTED.**

SMPSTS

STS entries contain source from ++SRC and ++SRCUPD elements processed during APPLY. The source elements maintained on the STS are those elements which have no target system library for APPLY.

- **These source elements are deleted when the SYSMOD(s) supplying the SRC or SRCUPDs are ACCEPTED.**

A SYSMOD consists of the following basic components:

- Header Modification Control Statement

  A header modification control statement (++APAR, ++FUNCTION, ++PTF, ++USERMOD) 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. The header statement supplies SMP with a unique seven character sysmod-id used to describe the SYSMOD.

- VERIFY Modification Control Statement

  The VERIFY statements (++VER) describe system, function and service dependencies in terms of a system release, FMIDs and requisite SYSMODs. At least one ++VER modification control statement must be present for a SYSMOD, and a maximum of 255 ++VER modification control statements are permitted.

  - FMID - is required for dependent-level functions and all non-function SYSMODs. The SYSMOD named in the FMID parameter is treated as a prerequisite for the installation of the SYSMOD which contains the FMID parameter.

    When specified for dependent-level functions, FMID implicitly indicates that elements in the dependent-level function are superior to those elements in the function named.

    When specified for non-function SYSMODs, FMID indicates the function-level of the elements in the SYSMOD. Unless functional "superiority" is expressed using the VERSION parameter, SMP will <u>exclude</u> elements from a SYSMOD if the corresponding elements in the target system belong to another function (that is, if the target system elements have a different FMID).

  - PRE and REQ - indicate that the SYSMOD requires the installation of the named SYSMOD(s). If the required SYSMOD(s) must be installed <u>before</u> the SYSMOD being built, it should be specified as a PRE (prerequisite); if the order in which SMP installs the SYSMODs is irrelevant, the required SYSMODs may be specified as REQs (requisites).

    For non-function SYSMODs, PRE is generally used to specify the presence of previously defined SYSMODs containing some, but not all, of the elements in the SYSMOD. Thus, PRE ensures that the complete set of elements (including those not in the present SYSMOD) is installed.

    Further, PRE provides a positive indication that the SYSMOD "knows" about earlier SYSMODs containing elements in common; this "knowledge" implies, to SMP, that the elements in the present SYSMOD are at a higher service-level than those in the prerequisite SYSMOD. Thus, PRE provides service-level relationships between SYSMODs in much the same manner as VERSION provides function-level relationships.

- NPRE - indicates that the SYSMOD being built is mutually exclusive with another SYSMOD, and that it should not be installed if the other SYSMOD is present on the target system. SMP will not install the SYSMOD with the NPRE parameter if the SYSMOD specified in the NPRE parameter is installed on the system.

- SUP (supersede) - indicates that this SYSMOD includes the modifications made by those SYSMODs named as superseded (SUP). SMP maintains a record of the superseded SYSMODs and allows installation of other SYSMODs which specify these SYSMODs as requisites.

  In contrast with the use of PRE, SUP is generally used when the present SYSMOD contains _all_ of the elements supplied by previously defined SYSMODs. When a SYSMOD does contain all the elements supplied by previously defined SYSMODs, the SYSMOD will SUP these previous SYSMODs indicating that they need not be installed.

  When both the superseded and superseding SYSMOD are processed in the same APPLY step, none of the elements (including in-line JCLIN) from the superseded SYSMOD are processed.

  As with the use of PRE, SUP provides a positive indication that the present SYSMOD "knows" about earlier SYSMODs containing elements in common and is used to derive service-level relationships between SYSMODs.

- DELETE - indicates that the SYSMOD is replacing the function SYSMODs named as operands of the DELETE parameter. DELETE is used to clean up data in the CDS and ACDS pertaining to the deleted functions: the CDS and ACDS SYSMOD entries for all PTFs, APARs, USERMODs and functions dependent upon the deleted functions are removed; the element entries (MOD, MAC and SRC) belonging to the deleted functions and dependent SYSMODs are removed; load module entries (LMODs) are removed if all the modules which make up the load module are removed. Further, CRQ/ACRQ FMID entries for explicitly deleted SYSMODs are themselves deleted, and CRQ/ACRQ SYSMOD entries are deleted if all of the FMID environments in the SYSMOD entry are explicitly deleted.

  Since DELETE removes CDS and ACDS entries containing element processing data (such as element distribution libraries and load module system libraries), the function deleting another function must supply sufficient data on the element statements (and possibly JCLIN) to allow SMP processing.

  SMP keeps a record for each SYSMOD which is explicitly deleted (that is, named as an operand on the DELETE parameter). SMP will not install any subsequent SYSMODs that require the presence of the explicitly deleted SYSMODs.

  DELETE may only be specified in a function SYSMOD.

- VERSION - indicates that the elements in the SYSMOD being built are functionally superior to elements belonging to the functions named as operands of the VERSION parameter.

In order to ensure the function level of a system, SMP will not select elements from a SYSMOD whose FMID does not match the FMID of the corresponding elements on the target system unless VERSION is coded to indicate that the SYSMOD's elements are functionally superior. VERSION is used to change the functional ownership of elements.

Since a dependent-level function's specification of its dependency to a base-level function (using FMID) also implies functional superiority to the base, a dependent-level function does not "version" the base-level function. If, however, a dependent-level function is supplying elements which are functionally superior to elements in functions other than its base, the dependent-level function must indicate this "superiority" relationship by coding the FMIDs of the "inferior" functions as VERSION operands.

- Conditional Requisite Modification Control Statements

  Conditional Requisite statements (++IFs) provide SMP with the data necessary to resolve requisites which are dependent upon particular functional (FMID) environments.

  ++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.

- Installation Data (JCLIN)

  The ++JCLIN statement specifies that a job stream of assembly, update, copy and link edit steps is available in order to provide SMP with information required to properly install the SYSMOD's elements on the operating system. There can be only one ++JCLIN modification control statement for each SYSMOD. It appears anywhere after the ++VER and ++IF modification control statements.

- Element Modification Control Statements

  The ++MAC, ++MOD, ++SRC, ++MACUPD, ++SRCUPD, ++UPDTE and ++ZAP modification control statements describe the elements being modified by the SYSMOD. They are referred to as element modification control statements.

  A SYSMOD may not be constructed which has more than one version of the same element type (such as two ++MOD(A)'s) or which has a replacement and update for the same element (such as a ++SRC(X) and a ++SRCUPD(X)).

  Although a source and a module element with the same names are related to one another, a SYSMOD may contain both a source modification and a ++MOD object deck for the same element. Such construction requires that the object deck match the object that would be produced if the corresponding source were assembled.

  Figure 36 illustrates the valid (VALID) and invalid (INV) combinations of modifications to the same element within one SYSMOD.

  <u>Note:</u> since JCLIN cannot determine distribution and operating system libraries for macros and source elements, ++MACs and ++SRC generally require

DISTLIB and SYSLIB information when they are first introduced to SMP.

	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 36: - Valid Modifications to the Same Element.

MISCELLANEOUS CONSTRUCTION RULES:

- 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) /* BASE LEVEL FUNCTION */ .
++VER(Z038) FMID(HVT1303) /K DEPENDENT FUNCTION */ .
```

- SMP must be able to determine the function to which a service SYSMOD (PTF, APAR or USERMOD) applies. Therefore, service SYSMODs that can be processed by SMP Release 4 must have an FMID coded on their Release 4 applicable ++VER statements.

  Since SMP will only consider ++VER statements whose SREL matches an SREL in a control dataset SYSTEM entry, it is possible to construct a service SYSMOD without an FMID on the ++VER as follows,

```
++PTF(UZ00001).
++VER(Z037) /* FOR MVS 3.7 / SMP R3 SYSTEMS */ .
++VER(Z038) FMID(ESY1400) /* FOR MVS 3.8 / SMP R4 SYSTEMS */ .
```

- The functional "owner" (FMID) of a SYSMOD on any given system release (SREL) must not be ambiguous. Therefore, a SYSMOD cannot be constructed which specifies different FMIDs for the same system release (SREL). The following example shows a SYSMOD construction error:

```
      ++PTF(UZ00005).
      ++VER(Z038) FMID(ESY1400).
      ++VER(Z038) FMID(ESY1101).
```

- Service SYSMODs cannot specify the DELETE or NPRE keywords on their SMP Release 4 ++VER statements.

- The same SYSMOD-ID may not be specified more than once in the same ++VER keyword operand list. The following illustrates this error by coding the same SYSMOD-ID twice in the PRE list.

```
      ++PTF(UZ00079).
      ++VER(Z038) FMID(ESY1400) PRE(UZ00010,UZ00010).
```

- The same SYSMOD-ID may not be specified in more than one ++VER keyword operand list. The following illustrates this error by coding the same SYSMOD-ID in the PRE and REQ lists.

```
      ++PTF(UZ00080).
      ++VER(Z038) FMID(GVT1202) PRE(UZ00010) REQ(UZ00010).
```

  Exception: 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.

- The FMID operand on a ++IF statement specifies the <u>conditonal</u> presence of a function whereas the FMID operand on a ++VER specifies the <u>absolute</u> presence of a function.

  Thus, FMID operand on a ++IF statement cannot be the same as the FMID specified in the associated ++VER 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).
```

- IEBUPDTE Control Cards: The only IEBUPDTE control statements allowed in the SYSMOD are the ./ CHANGE and ./ ENDUP. The member name specified on the ./ CHANGE statement must match the name in the SMP modification control statement.

  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.

  When processing multiple updates to the same source or macro, SMP will use the "./ CHANGE" statement from the last update to the element.

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

the indirect library technique, SYSMODs are packaged with element and JCLIN data in files separate from the file containing the modification control statements. Each indirect library is a partitioned dataset containing one or more members:

- MAC and SRC elements specify the DDNAME of the library containing the respective elements using the TXLIB keyword.
- The MOD elements specify the DDNAME of libraries containing object decks using TXLIB and link-edited load modules using LKLIB.
- JCLIN specifies the DDNAME of the library containing JCL using the TXLIB keyword. The library specified must be a partitioned dataset. The JCL must be in a member whose name matches the SYSMOD name. In the following example, SMP expects to find the JCL for SYSMOD FAA1000 in a member named FAA1000 in the library identified by the AAJCLIN DD statement.
- Update elements (ZAPS. MACUPDs and SRCUPDs) cannot be supplied in indirect TXLIB or LKLIB 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. 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. These files will be accessed during APPLY and ACCEPT processing.

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 37 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.

See 'RECEIVE Processing' on page 15 in Chapter 2 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	DSN = GBB3100.F1 Unloaded PDS containing member GBB3100, which is JCLIN data for function GBB3100
3	DSN = GBB3100.F2 Unloaded PDS containing modules A and B for function GBB3100
4	DSN = GBB3100.F3 Unloaded PDS containing macro X for function GBB3100
5	DSN = EBB3101.F1 Unloaded PDS containing member EBB3101, which is JCLIN data for function EBB3101
6	DSN = EBB3101.F2 Unloaded PDS containing modules A and C for function EBB3101
7	DSN = EBB3101.F3 Unloaded PDS containing macro Y for function EBB3101

Figure 37 - Physical Organization of Relative File Tape

RELFILE Packaging Notes:

- Tapes containing SYSMODs packaged with this technique must have standard labels.

- The files containing the unloaded partitioned datasets 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.

- The SMPPTFIN DD statement must point to the file on the tape containing the SMP modification control statements. This file is expected to be a sequential, card-image dataset.

- All RELATIVE files on the tape containing elements accessed by SMP are expected to be IEBCOPY-unloaded partitioned datasets. These files must follow the sequential file containing the SMP modification control statements.

  - The ++JCLIN RELFILE is expected to be a partitioned dataset containing a member matching the SYSMOD name. SMP accesses this member to obtain the SYSMOD's JCL data.

  - ++MOD RELFILES are expected to contain link-edited load modules.

  - Alias members must exist in the RELFILE datasets for each DALIAS, MALIAS and TALIAS specified on the SMP element modification control statements.

- In-line elements may be supplied in the sequential modification control statement file.

- Update elements (ZAPs, MACUPDs and SRCUPDs) cannot be supplied in RELATIVE files. They may, however, be supplied in-line for a SYSMOD which supplies other elements in RELATIVE files.

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. Figure 38 shows functions and module relationships:

MODULES

FUNCTIONS	A	B	C	D
ESY1400	x	x	x	x
ESY1500	x	x	x	
ESY1500		x	x	x

Assume:
ESY1500 is superior
to ESY1500;
ESY1500 is superior
to ESY1400

"x" means that the module
is present in the function

Figure 38 - Function and Module Relationships

The example that follows assumes that function ESY1400 must be present for either function ESY1500 or ESY1500 to be applicable. Either ESY1500 or ESY1500 can be present without the other but, if both are present, function ESY1500 is superior to function ESY1500.

```
++FUNCTION(ESY1400).
++VER(Z038).
++MOD(A)              /* FOR ESY1400 */.
++MOD(B)              /* FOR ESY1400 */.
++MOD(C)              /* FOR ESY1400 */.
++MOD(D)              /* FOR ESY1400 */.

++FUNCTION(ESY1500).
++VER(Z038) FMID(ESY1400).
++MOD(A)              /* FOR ESY1500 */.
++MOD(B)              /* FOR ESY1500 */.
++MOD(C)              /* FOR ESY1500 */.

++FUNCTION(ESY1500).
++VER(Z038) FMID(ESY1400) VERSION(ESY1500).
++MOD(B)              /* FOR ESY1500 */.
++MOD(C)              /* FOR ESY1500 */.
++MOD(D)              /* FOR ESY1500 */.
```

The following example shows how you would construct the PTFs required to fix APARs spanning all four modules in three functions:

```
++PTF(UZ13001).
++VER(Z038) FMID(ESY1400).
++IF FMID(ESY1500) THEN REQ(UZ13002).
++IF FMID(ESY1500) THEN REQ(UZ13003).
++MOD(A)             /* FOR ESY1400 */.
++MOD(B)             /* FOR ESY1400 */.
++MOD(C)             /* FOR ESY1400 */.
++MOD(D)             /* FOR ESY1400 */.
```

```
++PTF(UZ13002).
++VER(Z038) FMID(ESY1500) REQ(UZ13001).
++IF FMID(ESY1500) THEN REQ(UZ13003).
++MOD(A)            /* FOR ESY1500 */.
++MOD(B)            /* FOR ESY1500 */.
++MOD(C)            /* FOR ESY1500 */.

++PTF(UZ13003).
++VER(Z038) FMID(ESY1500) REQ(UZ13001).
++IF FMID(ESY1500) THEN REQ(UZ13002).
++MOD(B)            /* FOR ESY1500 */.
++MOD(C)            /* FOR ESY1500 */.
++MOD(D)            /* FOR ESY1500 */.
```

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.

```
++PTF(UZ13001).
++VER(Z037) PRE(E5Y1400) NPRE(ESY1500,ESY1500)
    REQ(UZ13002,UZ13003).
++VER(Z037) PRE(ESY1400,ESY1500) NPRECESY1500)
    REQ(UZ13006).
++VER(Z038) FMID(ESY1400) REQ(UZ13002,UZ13003).
++IF FMID(ESY1500) THEN REQ(U213004,UZ13005).
++IF FMIDCESY1500) THEN REQ(U213006).
++MOD(A)            /* FOR ESY1400 */.

++PTF(UZ13002).
++VER(Z037) PRE(ESY1400) NPRE(ESY1500,ESY1500)
    REQ(UZ13001,UZ13003).
++VER(Z038) FMID(ESY1400) REQ(UZ13001,UZ13003).
++IF FMID(ESY1500) THEN REQ(UZ13004,UZ13005).
++IF FMID(ESY1500) THEN REQ(UZ13006).
++MOD(B)            /* FOR ESY1400 */.
++MOD(C)            /* FOR ESY1400 */.

++PTF(UZ13003).
++VER(Z037) PRE(ESY1400) NPRE(ESY1500,ESY1500)
    REQ(UZ130014213002).
++VER(Z037) PRE(ESY1400,ESY1500) NPRE(ESY1500)
    REQ(UZ13004,UZ13005).
++VER(Z038) FMID(ESY1400) REQ(UZ13001,UZ13002).
++IF FMID(ESY1500) THEN REQ(UZ13004,UZ13005).
++IF FMID(ESY1500) THEN REQ(UZ13006).
++MOD(D)            /* FOR ESY1400 */.
```

```
++PTF(UZ13004).
++VER(Z037) PRE(ESY1400,ESY1500) NPRECESY1500)
    REQ(UZ13003,UZ13005).
++VER(Z037) PRE(ESY1400,ESY1500,ESY1500)
    REQ(UZ13006).
++VER(Z038) FMID(ESY1500) REQ(UZ13005,UZ13003).
++IF FMID(ESY1500) THEN REQ(UZ13006).
++MOD(A)             /* FOR ESY1500 */.

++PTF(UZ13005).
++VER(Z037) PRE(ESY1400,ESY1500) NPRE(ESY1500)
    REQ(UZ13003,UZ13004).
++VER(Z038) FMID(ESY1500) REQ(UZ13004,UZ13003).
++IF FMID(ESY1500) THEN REQ(UZ13006).
++MOD(B)             /* FOR ESY1500 */.
++MOD(C)             /* FOR ESY1500 */.

++PTF(UZ13006).
++VER(Z037) PRE(ESY1400,ESY1500) NPRE(ESY1500)
    REQ(UZ13001).
++VER(Z037) PRE(ESY1400,ESY1500,ESY1500)
    REQ(UZ13004).
++VER(Z038) FMID(ESY1500) REQ(UZ13001).
++IF FMID(ESY1500) THEN REQ(UZ13004).
++MOD(8)             /* FOR ESY1500 */.
++MOD(C)             /* FOR ESY1500 */.
++MOD(D)             /* FOR ESY1500 */.
```

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 "2037" will be ignored.

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

When IBM-supplied service is installed, the MODID verification checks described in Chapter 2 will ensure that you are notified by a message when the service intersects with your modification. The following situations might be encountered when service is installed on a system with user modifications:

- If a service SYSMOD attempts to replace an element you have replaced or updated, the service SYSMOD will be terminated.
- If a service SYSMOD attempts to update an element you have replaced, the service SYSMOD will be terminated.
- If a service SYSMOD attempts to update an element you have updated, the service SYSMOD will be processed and a message will inform you that there is a potential loss of your modification.

When function is installed, element selection is based upon functional relationships and the MODID verification checks will not detect overlaid user modifications. In general, the processing of the function is not affected by the presence of non-superseded PTFs, APARs and USERMODs. To provide an indication of overlaid user modifications, SMP will issue a message whenever a function SYSMOD element overlays an element which was modified by a USERMOD.

USERMOD CONSIDERATIONS

- The seven character sysmod-id should be choosen so as not to conflict with the conventions used by IBM. The sysmod-ids for IBM function and service SYSMODs begin with the letters A through K and U through Z; sysmod-ids beginning with the letters l through T and the numbers 0 through 9 are available for user modifications.
- When an APAR or USERMOD is built, it must name the functional owner of the element in the ++VER FMID keyword; if it does not, the element will not be updated. If the APAR/USERMOD affects elements from more than one function (FMID), more than one APAR/USERMOD package must be constructed. One APAR/USERMOD package is required for each function (FMID) whose elements are affected.
- An APAR/USERMOD should never use VERSION to control the selection of elements from a number of different FMIDs as this will change the functional ownership of the elements; subsequent IBM-supplied service will not be properly applied.

- User modifications which replace target system elements should PRE the last SYSMOD to replace the element (the element's RMID) and all SYSMODs which have updated the element since the last replacement (the element's UMIDs).

- User modifications which update target system elements should PRE the last SYSMOD to replace the element (the element's RMID); they need not PRE all previous updates.

- A user modification (or APAR) to a macro which causes an assembly will cause the ASSEMBLY indicator in the assembled module's MOD entry to be set. This indicator tells subsequent SMP processing that the module is in some way affected by a user modification to a macro. Since service modifications to the module will not know the relationship between the user macro change and the module, SMP will re-assemble the module whenever a subsequent SYSMOD attempts to modify the module. These re-assemblies may be suppressed by resetting the ASSEMBLY indicator in the module entry.

SUPERZAP MODIFICATIONS

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
```

Figure 39 - Load Module Structure for Zap Examples

The examples assume the use of the cataloged procedure described in "SMP Cataloged Procedure" from Chapter 3. The appropriate DD statements for defining the target system and distribution libraries have been added to the procedure.

ZAP 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
  /*
```

ZAP 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
  /*
```

ZAP 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
  /*
```

ZAP 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).
  NAME CSECT3
  VER 000D FF
  REP 0000 FE
  EXPAND CSECT3(4)
/*
```

ADDING NEW LOAD MODULES

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.

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='$$'
++USERMOD(MYMOD05).
++VER(Z038) FMID(FXY1000).
++JCLIN.
//MYJOB     JOB 1,'MYNAME',MSGLEVEL=1,CLASS=A
//STEP1     EXEC PGM=ASMBLR
//SYSPUNCH DD    DSN=USER.OBJPDS(IEFUSERA),DISP=OLD
//SYSIN     DD   *
  PRINT ON,NODATA
  USERMACA PARM1,PARM2
  COPY USERMACB
  END
/*
//STEP2     EXEC PGM=IEWL,PARM='RENT'
//SYSLMOD DD     DSN=SYS1.LINKLIB,DISP=OLD
//USERLIB DD     DSN=SYS1.USERLIB,DISP=OLD
//SYSPUNCH DD    DSN=USER.OBJPDS,DISP=OLD
//SYSIN     DD   *
  INCLUDE SYSPUNCH(IEFUSERA)
  INCLUDE USERLIB(IEFUSERB)
  ENTRY USERONE
  NAME USERTWO(R)
/*
//STEP3     EXEC PGM=IEBCOPY
//USERLIB DD     DSN=SYS1.USERLIB,DISP=OLD
//SVCLIB    DD   DSN=SYS1.SVCLIB,DISP=OLD
//SYSIN     DD
  COPY INDD=USERLIB,OUTDD=SVCLIB
  SELECT MEMBER=(USERSVC1,USERSVC2)
/*
++MAC(USERMACA) TXLIB(MACTXLIB) SYSLIB(MACLIB) DISTLIB(AMACLIB).
++MAC(USERMACB) TXLIB(MACTXLIB) SYSLIB(MACLIB) DISTLIB(AMACLIB).
++MOD(IEFUSERB) TXLIB(MODTXLIB) DISTLIB(USERLIB) LEPARM(RENT).
++MOD(USERSVC1) TXLIB(MODTXLIB) DISTLIB(USERLIB) LEPARM(RENT).
++MOD(USERSVC2) TXLIB(MODTXLIB) DISTLIB(USERLIB) LEPARM(RENT).
$$
```

Figure 40 - JCLIN For "Adding New Load Modules"

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. SMP assigns FYX1000 as the value of the FMID subentries for macros and modules created by this SYSMOD.

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. An FMID subentry of FXY1000 will be assigned to macros and modules created by this SYSMOD.

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; duplicotes 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 <u>replaced</u> the element is placed in the element modification control statement as the value of the RMID operand.

- For Each element that has been modified by integrated service SYSMODs, the SYSMOD-ID of the service SYSMODs that have <u>updated</u> the element since its last replacement are placed in the element modification control statement as the values of the UMID 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 2 for a description of how a service updated function SYSMOD is applied to a target system.

SAMPLE SERVICE UPDATED FUNCTION SYSMOD

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

```
++FUNCTION(FXX4101) FILES(S).
++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(IXXJWYDS) DISTLIB(AOS98) RELFILE(3).


++PTF(UZ13579).
++VER(Z038) FMID(FXX4101) SUP(AZ11335).
++MACUPD(IXXKLTD) DIST.LIB(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(IXXJWYDI) 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(U213644).
++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(IXXJWYDS) 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)
    UMID(UZ13579,UZ13601).
++MAC(IXXLQIQ) DISTLIB(AXXMACLB) RELFILE(2)
    UMID(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(IXXJWYDS) DISTLIB(AOS98) RELFILE(3)
    RMID(UZ13644).
```

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.

   a. The symbol '++' in the modification control statement must appear in bytes 1 and 2.

   b. The KEYWORD for the '++' must appear on the same card as the '++'

      Note: Except for these two restrictions, the control statements and modification control statements can begin and end anywhere up to and including byte 72.

2. The statement function must be specified first, followed by any keywords.

3. The optional keywords can be coded in any sequence, except where noted in the syntax and operand descriptions.

4. At least one blank must occur between each keyword.

5. Blanks or a comma, as specified in the syntax, must separate the keywords and their options.

6. 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.

7. Each statement must be terminated with a period(.).

8. Bytes 73-80 are ignored by SMP.

9. 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.

10. SMP completes processing one statement before the next statement is processed.

11. All input to SMP must be in UPPER CASE letters

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 | C

   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 compatibility 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 that 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. SMPCDS is the default data set for UCLIN operations.

APPENDIX D: SMP RELEASE 3/SMP RELEASE 4 AMMENDMENTS

This appendix describes the major changes between SMP Release 3 and Release 4. The information that you will read was formally the "Summary of Amendments" section of OS/VS SMP System Programmer's Guide GC28-0673-5, and is here to assist SMP Release 3 users to understand the basic differences between the two Releases.

INCOMPATIBILITIES

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

SUPPORT OF FUNCTION INSTALLATION

- SMP Release 4 recognizes when a function is being installed. Facilities are provided for the support of a hierarchy of function.
- SMP Release 4 provides facilities for the management of function. Specifically, SMP Release 4 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 Release 4 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 Release 4 to ensure that, upon function installation, the system is automatically brought up to the correct service level with respect to the functions installed.
- SMP Release 4 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

The user is 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 Release 4 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 Release 4 allows a user to specify a permanent parameter list in the SMP data sets to override some default operations.

- SMP Release 4 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 Release 4 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 Release 4. 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 Release 4 allows the user to receive all potentially applicable service into the PTS. SMP Release 4 then automatically groups together related service whenever the environment of the target system or distribution libraries changes.

- SMP Release 4 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 Release 4 allows multiple replacements and updates to elements to be processed concurrently during APPLY and ACCEPT processing.

- SMP Release 4 distinguishes between APAR fixes, PTFs, user modifications, and function modifications.

- SMP Release 4 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 `unction 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 Release 4 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 Release 4 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

The following is a list of new data set ddnames that have been added to SMP.

- SMPACRQ
- SMPADDIN
- SMPCRQ
- SMPLIST
- SMPTLIB
- SMPWRK1
- SMPWRK2
- SMPWRK3

- SMPPUNCH (new use)
- SMPRPT
- SMPSCDS
- SMPWRK4
- SMPWRK5
- SYSUT4

The following is a list of data sets that have been deleted from SMP.

- SMPREPIN
- SMPUCS

SMP CONTROL STATEMENTS

The following SMP control statements have been added:

- DEBUG MSGMODID - Provides debug facitlites.
- RESETRC - Sets SMP return code to zero.
- UNLOAD -CDS or ACDS data is punched in UCLIN format.

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.

SMP CONTROL STATEMENT KEYWORDS

The following new SMP control statement keywords have been added:

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

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

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

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

- REUSE causes SMP to use assemblies done during a previous (failing) SMP execution.

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

  - ASSEM, BYPASS, RETRY and REUSE are same as for ACCEPT above.

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

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

  - 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

  - BYPASS specifies MODID and requisite checking to be bypassed.

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

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

- UCLIN - ACRQ, CRQ, DIS, SCDS

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

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

  - 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

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

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

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

  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.

  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.

  NOREQ is replaced by the BYPASS(REQ) option.

  REPLACE is unnecessary because of support for user modifications.

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

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

  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

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

  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).

  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.

  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

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

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

EXEC CARD PARAMETERS

The following parameters can be specified in the FARM 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 data maintained by the operating system. 'yyddd' is the Julian date.

NORECOVERY

Specifies that the SMP ESTAE recovery environment ngi be established when running on non-VS1 and non-MVS systems.

ASM, COMPRESS, COPY, LKED, UPDTE and ZAP are no longer specifiable. See 'The UCL SYS Statement' in Chapter 6 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.

APPENDIX E: UCL OPERAND/DATA SET REFERENCE

This appendix consists of a set of charts which indicate the allowable UCL modifications to entries in the various SMP datasets.

The general syntax for UCL updates to SMP datasets is:

```
UCLIN dataset /* Begin UCL Updates */ .

{ ADD | DEL | REP } entry_type(entry_name) [sub_entry[,sub_entry]...] .

ENDUCL        /* End UCL Updates */ .
```

where,

<u>dataset</u> specifies the SMP data set (ACDS, CDS, ACRQ, CRQ, MIS, PTS, SCDS or STS);

<u>ADD</u> specifies that new data is to be added to an existing entry or that a new entry is to be created;

<u>DEL</u> specifies that an entry is to be deleted or, within an entry, subentries are to be deleted and indicators placed in reset state;

<u>REP</u> specifies that subentries are to be replaced and indicators placed in set state in an existing entry;

<u>entry type</u> specifies the type of entry to be updated (such as, ASSEM, DLIB, FMID, LMOD, MAC, MOD, SRC, SYSMOD and SYSTEM);

<u>entry name</u> specifies the name of the entry to be updated,

and

<u>sub entry</u> specifies the sub-entries and indicators that are to be updated.

<u>Notation:</u>

- All characters entered in upper case must be entered as shown.
- All characters entered in lower case represent data that must be specified as appropriate to the entry.
- When an item is followed by three periods (...) that indicates that the data value can be repeated multiple times. The limit is established by the value of PEMAX in either the SMPCDS, SMPACDS, or SMPPTS SYSTEM entry. An example of this is the GENASM sub-entry of the SMPCDS MAC entry:

  ```
  GENASM(name...)
  ```

- When an operand has a specified number of values then that indicates an upper limit. An example of this is the SYSLIB sub-entry for a SMPCDS LMOD entry:

  ```
  SYSLIB(ddname(,ddname))
  ```

  This indicates that at most two ddnames may be specified.

- The syntax of SMP allows for either blanks or commas to be used to separate items of a list. Thus the above example could have been specified as

  ```
  SYSLIB(ddname1,ddname2)
  or
  SYSLIB(ddname1 ddname2)
  ```

- "nn" indicate a numeric value must be entered with a length of at most the number of characters as the number of n's in the syntax.
- "ddname" names must be at most 8 characters long and must follow standard naming conventions.
- '|' indicates alternative forms of an operand.
- 'r' in the table indicates that the information specified by the operand must be present in the entry after all the UCL statement operands are processed. It does not mean that the operand must be specified on the UCL statement.
- 'c' in the table indicates that the operand is present in the current release of SMP only to maintain compatibility with previous releases. No processing, other than syntax checking, will be done for these operands.
- 'n' where n is a numeric value indicates that a note is present for this operand describing special considerations.
- 'x' in the table indicates that the operand is valid for the dataset.

UCL Examples:

- To DELETE an entire ENTRY, no sub-entries are specified as illustrated below:

  ```
  DEL entry_type(entry_name) .
  ```

  For example,

  ```
  DEL MAC(macro1) .
  ```

  will delete the specified macro entry.

- To Delete all occurrences of a particular multiple-occurrence sub-entry type,

  ```
  DEL entry_type(entry_name) sub_entry() .
  ```

  For example,

  ```
  DEL MAC(macro1) GENASM () .
  ```

  will delete all GENASM sub-entries for the specified macro.

- To change a particular single-occurrence sub-entry type,

  ```
  REP entry_type(entry_name) sub_entry(new_value) .
  ```

  For example,

  ```
  REP MAC(macro1) DLIB(new_dlib) .
  ```

  will change the distribution library for the specified macro.

- To "set" a particular indicator,

  ```
  ADD entry_type(entry_name) sub_entry .
  ```

  For example,

  ```
  ADD SYS PURGE .
  ```

  will set the PURGE indicator in the SYSTEM entry (note that the SYSTEM entry has no "entry_name" since there is only one SYSTEM entry in any particular SMP data set).

- To "reset" a particular indicator,

  ```
  DEL entry_type(entry_name) sub_entry .
  ```

  For example,

  ```
  DEL SYS PURGE .
  ```

  will reset the PURGE indicator in the SYSTEM entry.

The following chart illustrates the UCL functions which may be performed for the entries on each SMP data set.

UCL Functions / Data Set	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
ADD	x	x	x	x	x	x		
REP	x	x	x	x	x	x		
DEL	x	x	x	x	x	x	x	x

UCL Functions For Each SMP Data Set

The following chart illustrates the entries which exist in each SMP data set.

Entry / Data Set	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
ASSEM(name)	x							
DLIB(ddname)	x							
FMID(name)				x	x			
LMOD(name)	x							
MAC(name)	x	x					x	
MOD(name)	x	x						
PTF(sysmodid)	x	x						
SRC(name)	x	x						x
SYS	x	x				x		
SYSMOD(sysmodid)	x	x	x	x	x	x		

Entries to Data Set Chart

The charts on the following pages illustrate the sub-entries of each SMP data set ENTRY in terms of the UCL keywords used to modify them.

ASSEM ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
++ASMIN	r							
++ENDASMIN	r							
LASTUPD(JCLIN|UCLIN|sysmodid)	x							
LASTUPDTYPE(ADD|UPD)	x							

DLIB ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
SYSLIB(ddname(,ddname))	r							
LASTUPD(JCLIN|UCLIN|sysmodid)	x							
LASTUPDTYPE(ADD|UPD)	x							

FMID ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
SYSMOD(sysmodid...)				r	r			

LMOD ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
AC=1	x							
ALIGN2|ALN2	x							
COPY	x							
DC	x							
LASTUPD(JCLIN|UCLIN|sysmodid)	x							
LASTUPDTYPE(ADD|UPD)	x							
NE	x							
OVLY	x							
REFR	x							
RENT	x							
REUS	x							
SCTR	x							
STD	x							
SYSLIB(ddname(,ddname))	r							
++LMODIN	x							
++ENDLMODIN	x							

MAC ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
ASMLIB(ddname)	c							
BASE(FIXED)	c							
DISTLIB|DLIB(ddname)	x	x						
FMID(sysmodid)	r	r						
GENASM|ASSEM(name...)	x							
LASTUPD(JCLIN|UCLIN|sysmodid)	x	x						
LASTUPDTYPE(ADD|UPD)	x	x						
MALIAS(name...)	x							
RMID(sysmodid)	x	x						
SYSLIB(ddname)	x							
UMID(sysmodid...)	x	x						

<u>Note:</u> no sub-entries exist in the MTS MAC entry. The only UCL function which may be Performed against the MTS MAC entry is delete.

MOD ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
AC=1	x	x						
ALIGN2|ALN2	x	x						
DC	x	x						
DALIAS(name...)	x							
DISTLIB|DLIB(ddname)	r	r						
FMID(sysmodid)	r	r						
LASTUPD(JCLIN|UCLIN|sysmodid)	x	x						
LASTUPDTYPE(ADD|UPD)	x	x						
LMOD(name...)	x							
NE	x	x						
OVLY	x	x						
REFR	x	x						
RENT	x	x						
REUS	x	x						
RMID(sysmodid)	x	x						
SCTR	x	x						
TALIAS(name...)	x							
UMID(sysmodid...)	x	x						

SRC ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
BASE(FIXED)	c	r						
DISTLIB|DLIB(ddname)	r	r						
FMID(sysmodid)	r	r						
LASTUPD(JCLIN|UCLIN|sysmodid)	x	x						
LASTUPDTYPE(ADD|UPD)	x	x						
RMID(sysmodid)	x	x						
SYSLIB(ddname)	x							
UMID(sysmodid...)	x	x						

Note: no sub-entries exist in the STS SRC entry. The only UCL function which may be performed against the STS SRC entry is delete.

SYS (System) ENTRY Sub-ENtries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
ASMNAME(name)						x		
ASMPARM(parm)						x		
ASMPRINT(ddname)						x		
ASMRC(nn)						x		
CDSID(name)	r	r						
COMPNAME(name)						x		
COMPPARM(parm)						x		
COMPPRINT(ddname)						x		
COMPRC(nn)						x		
COPYNAME(name)						x		
COPYPARM(parm)						x		
COPYPRINT(ddname)						x		
COPYRC(nn)						x		
DSPREFIX(value)						x		
DSSPACE(prim,sec,dirblks)						r		
FMID(sysmodid...)						x		
IOSUPNAME(name)						x		
IOSUPPARM(parm)						x		
IOSUPPRINT(ddname)						x		
IOSUPRC(nn)						x		
LKEDNAME(name)						x		
LKEDPARM(parm)						x		
LKEDPRINT(ddname)						x		
LKEDRC(nn)						x		
NUCID(n)	r	r						
PAGELEN(nnnn)						x		
PEMAX(nnnn)	x	x				x		
PURGE						x		
REJECT						x		
RETRYNAME(name)						x		
RETRYPARM(parm)						x		
RETRYPRINT(ddname)						x		
RETRYRC(nn)						x		
RETRYDDN(ddname...)						x		
SAVEMTS	x	x						
SAVESTS	x	x						
SREL(cnnn)	r	r						
SREL(cnnn...)						r		
UPDATNAME(name)						x		
UPDATPARM(parm)						x		
UPDATPRINT(ddname)						x		
UPDATRC(nn)						x		
ZAPNAME(name)						x		
ZAPPARM(parm)						x		
ZAPPRINT(ddname)						x		
ZAPRC(nn)						x		

SYSMOD ENTRY Sub-Entries	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
APAR|PTF|FUNCTION|USERMOD	x	x						
ACCDATE(yyddd)	x	x						
ACCEPT|ACPT|ACC	x	x						
ACCID(cdsid...)						x		
ACCTIME(hh:mm:ss)	x	x						
APPDATE(yyddd)	x	x						
APPID(cdsid...)						x		
APPLY|APPL|APP	x	x						
APPTIME(hh:mm:ss)	x	x						
ASSEM(name...)	x	x						
BYPASS|BYP	x	x						
DATE(yyddd)	c	c						
DELBY(sysmodid)	x	x						
DELETE(sysmodid...)	x	x						
ERROR|ERR	x	x						
FESN(sysmodid)	x	x						
FMID(sysmodid)	x	x						
FMID(sysmodid...)				r	r			
IFREQ(sysmodid...)	x	x						
JCLIN	x	x						
LASTSUP(sysmodid)	x	x						
LASTUPD(JCLIN|UCLIN)	x	x						
LASTUPDTYPE(ADD|UPD)	x	x						
MAC(name...)	x	x						
MACUPD(name...)	x	x						
MOD(name...)	x	x						
NPRE(sysmodid...)	x	x						
PRE(sysmodid...)	x	x						
RECDATE(yyddd)	x	x						
RECTIME(hh:mm:ss)	x	x						
REGEN|RGN	x	r						
REQ(sysmodid...)	x	x						
RESDATE(yyddd)	x	x						
RESTIME(hh:mm:ss)	x	x						
RESTORE|REST|RES	x							

(continued on next page)

SYSMOD ENTRY Sub-Entries (continued)	CDS	ACDS	SCDS	CRQ	ACRQ	PTS	MTS	STS
RMAC(name...)	x	x						
RMACUPD|RMUPD(name)	x	x						
RMOD(name...)	x	x						
RSRC(name...)	x	x						
RSRCUPD|RSUPD(name)	x	x						
RSZAP(name...)	x	x						
RXZAP(name...)	x	x						
SRC(name...)	x	x						
SRCUPD(name...)	x	x						
SUPBY|SUP(sysmodid...)	x	x						
SUPING(sysmodid...)	x	x						
SZAP(name...)	x	x						
UCLDATE(yyddd)	x	x						
UCLTIME(hh:mm:ss)	x	x						
UPDTE(name...)	c	c						
VERNUM(nnn)	x	x						
VERSION(sysmodid...)	x	x						
XZAP(name...)	x	x						

Notes - CDS/ACDS SYSMOD Entries:

1. The keywords APAR, PTF, FUNCTION and USERMOD specify the SYSMOD type. Only one type is valid for any given SYSMOD entry or sysmod-id.

2. An element sub-entry (module, macro or source) cannot be added or replaced if there is an update or regressed element sub-entry for the same element (and vice versa). For example, a MAC sub-entry cannot be added it the SYSMOD already has a MACUPD sub-entry for the same macro.

3. NPRE sub-entries can only be added to FUNCTION-type SYSMOD entries.

4. The APPLY, APPDATE and APPTIME sub-entries cannot be removed from a CDS SYSMOD entry.

5. The ACCEPT, ACCDATE and ACCTIME sub-entries cannot be removed from an ACDS SYSMOD entry.

6. In order to remove the apply indications from an ACDS SYSMOD entry, the APPLY, APPDATE and APPTIME sub-entries must all be deleted.

7. In order to remove the accept indications from a CDS SYSMOD entry, the ACCEPT, ACCDATE and APPTIME sub-entries must all be deleted.

8. A CDS SYSMOD entry with the RESTORE indicator set is always "in-error" (the ERROR indicator is set). The ERROR indicator can only be deleted in conjunction with the deletion of the RESTORE, RESDATE and RESTIME sub-entries.

GLOSSARY

This glossary defines terms used in SMP publications. 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 -

accent

In SMP, the process initiated by the ACCEPT control statement that places SYSMODs into the distribution libraries or permanent user libraries.

accented SYSMOD

A SYSMOD which has been successfully processed by the SMP ACCEPT function. Accepted SYSMODs are those found as SYSMOD entries on the ACDS with the ERROR flag off.

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.

applied SYSMOD

A SYSMOD which has been successfully processed by the SMP APPLY function. Applied SYSMODs are those found as SYSMOD entries on the CDS with the ERROR flag off.

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 -

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.

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.

__entry__

The term entry refers to a member of an SMP dataset. With the exception of the MCS entry in the SMPPTS dataset, these member names are encoded and cannot be easily accessed by utilities other than SMP. SYSMOD and MACRO entries are examples of the types of entries maintained in the SMPCDS dataset.

__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 -

<u>header modification control statement</u>

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.

<u>hierarchy</u>

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 -

<u>IMASPZAP</u>

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.

<u>indicator</u>

Is a field in an SMP dataset entry that does not have a data value associated with it. An example of an indicator is the APP indicator in the SMPCDS SYSMOD entry. An indicator is either "on" or "off".

<u>inline JCLIN</u>

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.

<u>inner macro</u>

Is a macro invoked by another macro. In particular, inner macros are those which SMP does not detect during JCLIN processing of assembler job steps.

<u>install</u>

In SMP, to apply a SYSMOD into the target system libraries or to accept a SYSMOD into the distribution libraries.

- J -

<u>JCLIN</u>

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 i5 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 successfully 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. operating system

The system updated by APPLY and RESTORE processing. Also referenced to as the target system.

- 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 of removing SYSMODs from the PTS data set and temporary libraries from the SMPTLIB volumes. The reject process may invoked by the REJECT control statement or as part of ACCEPT and RESTORE processing. For ACCEPT and RESTORE processing, the PURGE and REJECT indicators in the PTS SYSTEM entry determine whether reject will be invoked.

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 operating 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.

sub-entry

Is a field within an entry. Each sub-entry has associated with it a type and a value. Multiple occurrences of the same sub-entry type may exist in an entry each with a different value. For example, the modules supplied by a PTF are saved as "MOD" type sub-entries within the PTF's SYSMOD entry. Some sub-entries may occur only once within an entry; for example, the CDSID sub-entry in a CDS SYSTEM 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.

super zap

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" 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 system updated during APPLY and RESTORE processing. Also referred to as the operating system.

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 -

UCL statement

Is the command to SMP that results in a change to one of the SMP dataset entries. UCL statements come between the UCLIN and ENDUCL commands. The UCL statement specifies the action to be taken (ADD, REP or DEL), the entry to be modified and possibly the indicators and sub-entries to be affected.

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 -

<u>VERSION</u>

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.

Modification Control Statements

OS/VS System Modification Program
(SMP) System Programmer's Guide
GC28-0673-6

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.

Customer	**(Q)** Install Mgr.	**(U)** System Consult.	**(X)** System Analyst	**(Y)** System Prog.	**(Z)** Applica. Prog.	**(F)** System Oper.	**(I)** I/O Oper.	**(L)** Term. Oper.

(O) **Other**

IBM	**(S)** System Eng.	**(P)** Prog. Sys. Rep.	**(A)** System Analyst	**(B)** System Prog.	(C) Applica. Prog.	**(D)** Dev. Prog.	**(R)** Comp. Prog.	**(G)** System Oper.	**(J)** I/O *Oper.*	**(E)** Ed. Dev. Rep.	**(N)** Cust. Eng.	**(T)** 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.)

GC28-0673-6

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Cut or Fold Along Line

OS/VS System Modificatio rogram (SMP) System Programmer's Guide (S370-37) Printed in U.S.A. GC28-0673-6