z/OS
2.5

z/OS SMP/E Commands

SA23-2275-50

Note

Before using this information and the product it supports, read the information in “Notices” on page 551.

This edition applies to Version 2 Release 5 of z/OS® (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2023-05-12

Contents

Figures

Tables

About this document

Use this publication when you need to code SMP/E commands and read SMP/E reports.

Table 1 on page xxv shows where to find various types of information contained in this publication.

Table 1. Types of information in SMP/E Commands

Topic	Description
Syntax notation and rules	Describes the syntax notations and the general syntax rules for SMP/E commands and input. See Chapter 1, "Syntax notation and rules," on page 1
Commands	ACCEPT command through Chapter 33, "The ZONERENAME command," on page 449 contain detailed information about the SMP/E commands, including syntax, operands, required data sets, usage notes, examples, and processing.
SMP/E reports	Contains descriptions of all the reports produced by SMP/E. See Chapter 34, "SMP/E reports," on page 457

This publication also includes these appendixes:

- Appendix A, "Processing the SMP/E RC operand," on page 541 explains the RC operand.
- Appendix B, "Sharing SMP/E data sets," on page 543 describes how to control access to SMP/E zones.

SMP/E publications

The IBM SMP/E for z/OS, V3R6 publications are available as PDF files in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Table 2 on page xxv lists the IBM SMP/E for z/OS, V3R6 publications and briefly describes each one.

For information about z/OS publications and more information about the IBM SMP/E for z/OS, V3R6 books, see *z/OS Information Roadmap*.

Table 2. Publications for IBM SMP/E for z/OS, V3R6

Title	Description
z/OS SMP/E Messages, Codes, and Diagnosis, GA32-0883	Explains SMP/E messages and return codes and the actions to take for each; and how to handle suspected SMP/E problems.
z/OS SMP/E Commands, SA23-2275	Explains SMP/E commands and processing in detail.
z/OS SMP/E Reference, SA23-2276	Explains SMP/E modification control statements, data sets, exit routines, and programming interfaces in detail and provides additional SMP/E reference material.
z/OS SMP/E User's Guide, SA23-2277	Describes how to use SMP/E to install programs and service.

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xxvii.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Documentation function
If your comment or question is about the IBM Documentation functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Documentation Support at ibmdocs@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS SMP/E Commands, SA23-2275-50
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under IBM z/OS Product Documentation Update Policy (www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/ibm-zos-doc-update-policy?OpenDocument).

Summary of changes for z/OS SMP/E Commands Version 3 Release 7 in z/OS Version 2 Release 5 (V2R5)

New

The following content is new.

February 2023

For APAR IO28360, the RECEIVE Command is updated to support verifying the digital signatures for signed GIMZIP packages. Refer to "Content of CLIENT data set" on page 248 for more information.

Prior to February 2023

A note was added to the EXCLUDE operand of the APPLY and ACCEPT commands, refer to APPLY "Operands" on page 53 and ACCEPT "Operands" on page 7 for more information.

Changed

The following content is changed.

Prior to February 2023

The Note in topic "Zones for SET BOUNDARY" on page 107 was updated.

Deleted

The following content is deleted.

- None.

Summary of changes for z/OS SMP/E Commands Version 3 Release 7 in z/OS Version 2 Release 4 (V2R4)

New

The following content is new.

November 2020 refresh

Additional information regarding the SHELLSCR element was added, refer to "Hierarchical file system element and Java archive file replacements" on page 101.

Prior to September 2020 refresh

- SMP/E V3R7 overview in *z/OS SMP/E User's Guide*

Changed

The following content is changed.

September 2020 refresh

The syntax diagram for the LIST command was corrected, refer to "Global zone syntax" on page 208.

Prior to September 2020 refresh

- The notes were updated in the distribution and target zone syntax section, see "Distribution zone and target zone syntax" on page 206.

Deleted

The following content is deleted.

- None.

Changes made in SMP/E Version 3 Release 6

This document contains information that was previously presented in *SMP/E Commands*, SA23-2275-05, which supports z/OS Version 2 Release 2.

Changed information

- For APAR IO25506, the subdirectory name format was updated; see "Network transfer" on page 274.
- For APAR IO25475, updates were made to the link edit attributes; see Chapter 9, "The JCLIN command," on page 153.
- The exception SYSMOD Report generated by the REPORT ERRSYSMODS command now includes superceded SYSMODS; see Chapter 17, "The REPORT ERRSYSMODS command," on page 303.
- Updates were made to the ZoneMerge command; see Chapter 32, "The ZONEMERGE command," on page 437.
- Attribute *ftpccc*, which specifies whether SMP/E should use the z/OS FTP client CCC subcommand, was added to the <CLIENT> tag in the CLIENT/SMPCLNT data set. See "Content of CLIENT data set" on page 248 for more information.
- Chapter 14, "The RECEIVE command," on page 233 was updated with information about HTTPS support.
- To reflect the support for multitasking using the SYSPRINT definition in the GIMDDALC dataset, the section about the multi-tasking of link-edit utility invocations of the following commands has been updated:
  - ACCEPT command ("Multitasking of link-edit utility invocations" on page 44)
  - Chapter 3, "The APPLY command," on page 51 ("Multitasking of link-edit utility invocations" on page 97)
- The following commands were enhanced to allow users to specify target or dlib zones that are defined in different global zones
  - REPORT CROSSZONE command
  - REPORT SYSMODS command
- The REPORT SYSMODS command was changed to generate a new report called the SYSMOD Comparison HOLDDATA Report. It identifies the SYSTEM and USER HOLDS that must be resolved before the SYSMODS identified in the SYSMOD Comparison Report can be installed in the comparison zone. For more information, see "SYSMOD comparison HOLDDATA report" on page 523.
- Because the REPORT SYSMODS command was changed to identify SYSTEM and USER HOLDS that must be resolved before the SYSMODS identified in the SYSMOD Comparison Report can be installed in the comparison zone, the following commands were modified to retain HOLDDATA information:
  - REJECT command
  - REPORT SYSMODS command
  - RESTORE command

- Chapter 9, “The JCLIN command,” on page 153 was updated as a result of the added support for the RMODE(31) Binder option.
- A new comment for the STATUS field was added to “RECEIVE summary report” on page 505.
- These figures were updated because the REDO operand was replaced by the CHECK and BYPASS(HOLDSYSTEM) operands:
  - Figure 10 on page 296
  - Figure 12 on page 299
  - Figure 15 on page 306
  - Figure 17 on page 308
  - Figure 18 on page 313
  - Figure 21 on page 315
  - Example of SMPPUNCH output for REPORT SYSMODS

Changes made in SMP/E Version 1 Release 5

This section lists the changes that were made in SMP/E Version 1 Release 5.

SA22-7771-14

Changed information:

- The syntax diagram showing the SYSOUT DDDEF UCLIN operands was corrected. For details, see “SYSOUT data set” on page 368.

SA22-7771-13

Changed information:

- Chapter 9, “The JCLIN command,” on page 153 was updated for the following two changes:
  - A blank space is no longer required between the closing parenthesis and TYPE comment in the JCLIN INCLUDE statement.
  - X'C0' is a valid character in a module name that is specified on the INCLUDE statement.
  - The notes in Chapter 24, “The UCLIN command,” on page 361 were updated to explain that the SOURCEID value cannot span lines and must be explicitly specified.

SA22-7771-12

New information:

- You can use the REPORT MISSINGFIX command (Chapter 18, “The REPORT MISSINGFIX command,” on page 311) to determine whether any FIXCAT APARs exist that are applicable but are not installed yet, and whether any SYSMODs are available to satisfy the missing FIXCAT APARs. The Missing FIXCAT SYSMOD report was also been added; see “Missing FIXCAT SYSMOD report” on page 496.

Changed information:

- You can now use the ZONEEDIT command to add certain subentries to selected SMP/E entries in the same zone. Chapter 29, “The ZONEEDIT command,” on page 417 has been updated to describe this change.
- The description of the SOURCEID operand, the EXSRCID operand, or both have been updated for the following commands:
  - Chapter 3, “The APPLY command,” on page 51
  - ACCEPT command
  - Reject command

  - Chapter 12, “The LIST command,” on page 205
  - Chapter 25, “The UNLOAD command,” on page 393
- The notes in Chapter 24, “The UCLIN command,” on page 361 have been updated to explain that the SOURCEID value cannot span lines and must be explicitly specified.
- The Data sets used sections in ACCEPT command, Chapter 3, “The APPLY command,” on page 51, and Chapter 14, “The RECEIVE command,” on page 233 have been updated to include the optional data set SMPHRPT.
- The following chapters are updated for the FIXCAT operand:
  - ACCEPT command
  - Chapter 3, “The APPLY command,” on page 51
  - Chapter 12, “The LIST command,” on page 205
  - Chapter 14, “The RECEIVE command,” on page 233
  - Chapter 24, “The UCLIN command,” on page 361
  - Chapter 34, “SMP/E reports,” on page 457

Chapter 1. Syntax notation and rules

This chapter explains the syntax notation and rules for SMP/E commands. It describes:

- How to read the notation used to show how commands should be coded
- The rules to follow when coding commands

How to read the syntax diagrams

Throughout this publication, the following structures are used in describing syntax:

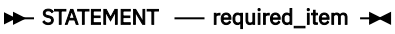
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The →→– symbol indicates the beginning of a command.

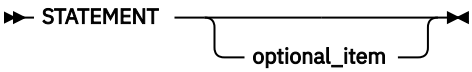
  The –→ symbol indicates that the command syntax is continued on the next line.

  The →– symbol indicates that a command is continued from the preceding line.

  The –→← symbol indicates the end of a command.
- Required items appear on the horizontal line (main path).

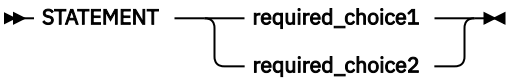


- Optional items appear below the main path.

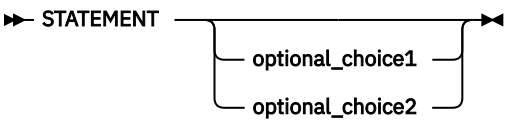


- If you can choose from two or more items, they appear in a vertical stack.

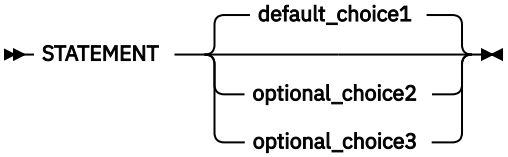
  If you **must** choose one of the items, one item of the stack appears on the main path.



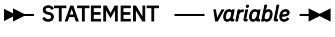
  If choosing one of the items is optional, the entire stack appears below the main path.



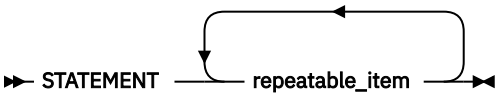
  If one of the optional items is the default, it appears above the main path and the remaining choices will be shown.



- Keywords appear in uppercase (for example, PARM1). **They must be spelled exactly as shown.**
- Variables appear in lowercase italics (for example, *parmx*). They represent user-supplied names or values.



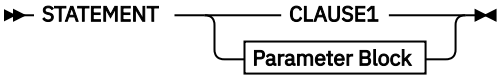
- An arrow returning to the left above the main line indicates an item that can be repeated.



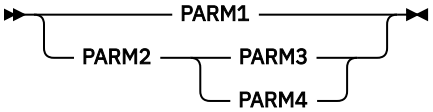
  A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

  – A repeat arrow above a stack of keywords means that you can enter one or more of the keywords. However, **each keyword can be entered only once**.
  – A repeat arrow above a variable means that you can enter one or more values for the variable. However, **each value can be entered only once**.

- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.
- Sometimes a single substitution represents a set of several parameters. For example, in the following diagram, the callout `Parameter Block` can be replaced by any of the interpretations of the subdiagram that is labeled `Parameter Block:`.



Parameter Block



Syntax rules

Follow these rules when you code SMP/E commands:

- SMP/E input is case-sensitive. Use uppercase letters to enter all SMP/E keywords. Enter operands in the same case as the intended operand values. Enter the text within a comment in any case you prefer.
- Start each command on a new logical 80-byte record.

  For SMP/E commands, enter the command name first, followed by any operands.

  Note: Except for these restrictions, SMP/E commands can begin and end anywhere up to and including column 72.
- You can code optional information in any order, except where noted in the syntax and operand descriptions.
- Separate operands and their values with a blank or comma.

  Note: Although the syntax diagrams show only commas when indicating the allowable separator characters for repeating values, one or more blank characters may be used instead to separate repeating values.
- You can continue a command on more than one line. SMP/E assumes a command is continued if it did not find a period (.) before column 73.

  Note: If an operand's value must span multiple lines and that value is delimited by quotation marks, the value should extend up to and including column 72 and restart on column 1 of the next line. Put a quotation mark before the value and another after the value, but do **not** add extra quotation marks where the value spans lines. Blanks within the quoted value are considered to be part of the value, including any blanks at the beginning of a continuation line.
- Start comments with "/*" and end them with "*/". The first "*/" encountered after the initial "/*" will end the comment. A comment can appear anywhere within or after a command, but should not start before a command, nor begin in column 1. (When "/*" starts in column 1, it indicates the end of an input

data set.) A comment after the ending period **must** start on the same line as the period. You cannot specify any additional operands or comments after that final comment. For example, you can code a comment like this:

```
SET      BDY(MVSTST1)      . /* Comment after period
                               continued on subsequent
                               records is okay.        */
```

However, you should **not** code a comment like this:

```
SET      BDY(MVSTST1)  .    /* Comment after period okay */
                            /* but this comment will give a
                               syntax error */
```

This causes a syntax error at the start of the second comment after the period.

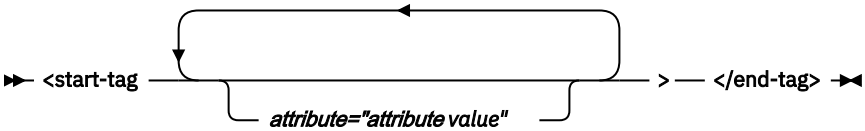
- Comments can be in single-byte characters (such as English alphanumeric characters) or in double-byte characters (such as Kanji).
- For a parameter that allows or requires the use of quotation marks as part of the parameter's value, the parameter value should extend up to and including column 72 and restart on column 1 of the next line. No intervening quotation marks are needed. Intervening blanks will be incorporated into the value.
- End each command with a period.
- SMP/E completes processing for one command before it starts processing the next one.
- SMP/E ignores columns 73 through 80. If data, such as a period, is specified beyond column 72, SMP/E ignores it and indicates an error in the command after the one containing that data.

Syntax rules for XML statements

XML statements may be coded in the CLIENT, SERVER, SYSIN, file attribute, and package attribute files for use with the GIMZIP and GIMUNZIP service routines.

The following syntax rules apply to XML statements:

- SMP/E ignores columns 73 through 80.
- All tags have a starting and ending delimiter specified as ***<keyword>*** and ***</keyword>***, respectively.
- Any tag that does not contain another tag (that is, nested tags) may have an ending delimiter of either ***</keyword>*** or just ***/>***.



- Comments must begin with <!-- and end with"-->. All data between the <!-- and the --> is ignored. Comments cannott be placed inside a tag.
- Any text not contained within comment delimiters is syntax checked.
- Tags are case sensitive; attribute values may be mixed case.
- A tag is not required to start on a new line.
- The XML markup characters, <, >, and &, cannot appear within a tag name or an attribute value.

Chapter 2. The ACCEPT command

The ACCEPT command is used to cause SMP/E to install the elements supplied by a SYSMOD into the distribution libraries (or DLIBs). The ACCEPT process:

- Selects SYSMODs present in the global zone that are applicable to the specified distribution libraries
- Makes sure all other required SYSMODs have been accepted or are being accepted concurrently
- Selects the elements from the accepted SYSMODs based on the functional and service level of those elements in the distribution libraries and the relationship between the SYSMODs being installed, ensuring that no current service is regressed by the installation of another SYSMOD
- Calls system utilities to install the elements into the distribution libraries
- Records the functional and service levels of the new elements in the distribution zone
- Records the installation of the SYSMOD in the distribution zone
- Deletes the global zone SYSMOD and PTS modification control statement entries for those SYSMODs that were successfully processed

The ACCEPT process is controlled by:

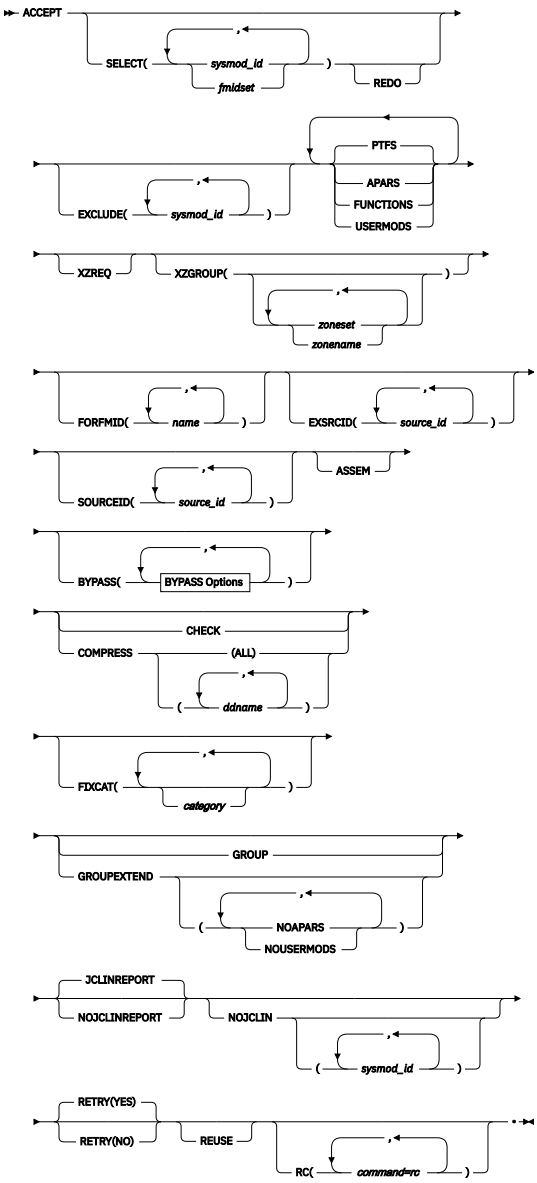
- The information in the distribution zone reflecting the status and structure of the distribution libraries
- Information on the SYSMODs indicating their applicability
- Information in the OPTIONS and UTILITY entries
- Operands on the ACCEPT command

Zones for SET BOUNDARY

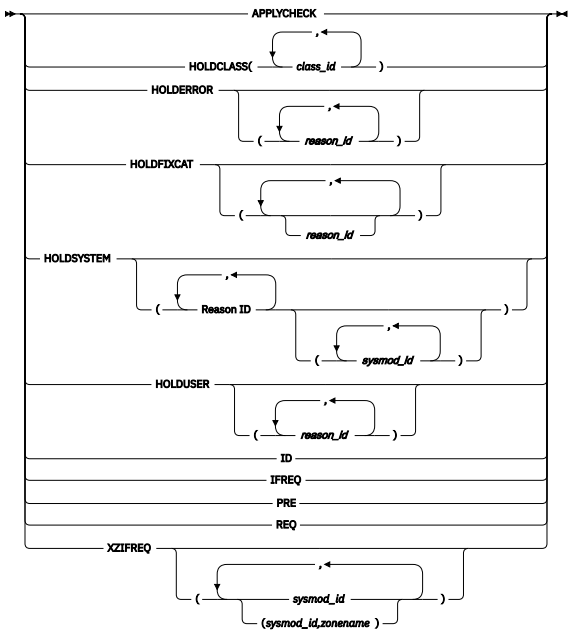
For the ACCEPT command, the SET BOUNDARY command must specify the distribution zone associated with the distribution libraries where the SYSMODs will be installed.

Syntax

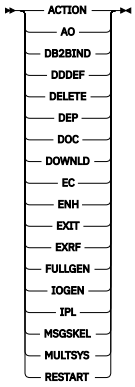
ACCEPT Command



BYPASS Options



HOLDSYSTEM Reason IDs Used by IBM



Operands

APARS
indicates that all eligible APARs should be accepted.

Note:

1. APARS can also be specified as APAR.
2. If APARS is specified along with SELECT, all eligible APARs are included in addition to the SYSMODs specified on SELECT.
3. If APARS is specified along with SOURCEID, all APARs associated with the specified source IDs are included.

ASSEM
indicates that if any SYSMODs contain both source code and object code for the same module, the source code should be assembled and should replace the object code.

BYPASS
You can specify any of these options:

APPLYCHECK
HOLDCLASS
HOLDERROR
HOLDFIXCAT

HOLDSYSTEM
HOLDUSER
ID
IFREQ
PRE
REQ
XZIFREQ
XZIFREQ(*list*)

Note: If you specify both BYPASS and GROUPEXTEND, SMP/E does not include superseding SYSMODs needed to take the place of requisites or error reason IDs that have been bypassed.

During CHECK processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify GROUPEXTEND without BYPASS.

BYPASS(APPLYCHECK)
indicates that SYSMODs should be accepted even if they have not been applied. For example, if you are preparing the distribution libraries before doing a system generation, you want to accept SYSMODs that have not been applied.

Note: APPLYCHECK can also be specified as APPCHK.

BYPASS(HOLDCLASS(*value*,...))
indicates that exception SYSMODs associated with the specified class names should not be held. The list of class names is required.

These are the hold classes you can specify:

Class
Explanation

ERREL
The SYSMOD is held for an error reason ID but should be installed anyway. IBM has determined that the problem the SYSMOD resolves is significantly more critical than the error reflected by the holding APAR.

HIPER
The SYSMOD is held with a hold class of HIPER (High Impact)

PE
The SYSMOD is held with a hold class of "PTF in Error".

UCLREL
UCLIN needed for the SYSMOD has been handled by IBM and no longer requires your attention.

YR2000
Identifies PTFs that provide Year 2000 function, or fix a Year 2000-related problem.

BYPASS(HOLDERROR)
indicates that exception SYSMODs associated with the specified error reason IDs should not be held. The list of reason IDs is optional. If you include one, only the reason IDs specified on it are bypassed. If you do not include a list, all error reason IDs are bypassed.

Note: HOLDERROR can also be specified as HOLDERR.

BYPASS(HOLDFIXCAT)
indicates that the held SYSMODs associated with the specified fix category reason IDs should not be held. The list of reason IDs is optional. If a list of reason IDs is included, only the ones specified are bypassed. If a list is not included, all fix category reason IDs are bypassed.

BYPASS(HOLDSYSTEM)
indicates that exception SYSMODs associated with the specified system reason IDs should not be held. The list of reason IDs is optional, as is the list of SYSMOD IDs for a particular reason ID. Generally, you should specify BYPASS(HOLDSYSTEM) on all ACCEPT CHECK commands, and

BYPASS(HOLDSYSTEM(*reason_id*,...)) on all ACCEPT commands for all system reason IDs for which appropriate action has been (or will be) taken.

How you specify the reason IDs determines which system reason IDs are bypassed. Make sure the appropriate action has been taken for all SYSMODs whose reason IDs are to be bypassed.

- If you do not include a list of reason IDs, all system reason IDs are bypassed.
- If you include a list of reason IDs without a list of SYSMOD IDs, all the SYSMODs with the specified reason IDs are bypassed.

  If you include a list of SYSMOD IDs for a particular reason ID, that reason ID is bypassed only for the specified SYSMODs. Other SYSMODs held for that reason remain held, unless the hold is released by some other BYPASS operand (such as CLASS).

Note: HOLDSYSTEM can also be specified as HOLDSYS.

These are the system reason IDs currently used by IBM:

ID
Explanation

ACTION
The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.

AO
The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.

DB2BIND
A DB2® application REBIND is required for the SYSMOD to become effective.

DDDEF
Data set changes or additions as required.

DELETE
The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.

DEP
The SYSMOD has a software dependency.

DOC
The SYSMOD has a documentation change that should be read before the SYSMOD is installed.

DOWNLD
Code that is shipped with maintenance that needs to be downloaded.

DYNACT
The changes supplied by the SYSMOD may be activated dynamically without requiring an IPL. The HOLD statement describes the instructions required for dynamic activation. If those instructions are not followed, then an IPL is required for the SYSMOD to take effect.

EC
The SYSMOD needs a related engineering change.

ENH
The SYSMOD contains an enhancement, new option or function. The HOLD statement provides information to the user regarding the implementation and use of the enhancement.

EXIT
The SYSMOD contains changes that may affect a user exit. For example, the interface for an exit may be changed, an exit may need to be reassembled, or a sample exit may be changed.

EXRF
The SYSMOD must be installed in both the active and the alternative Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)

FULLGEN
The SYSMOD needs a complete system or subsystem generation to take effect.

IOGEN
The SYSMOD needs a system or subsystem I/O generation to take effect.

IPL
The SYSMOD requires an IPL to become effective. For example, the SYSMOD may contain changes to LPA or NUCLEUS, the changes may require a CLPA, or a failure to perform an IPL might lead to catastrophic results, such as could be caused by activation of a partial fix.

Note: If you plan to perform an IPL with CLPA after the SYSMOD has been applied, then no further investigation of the HOLD is required; simply bypass the IPL reason ID. However, if you are not planning to perform an IPL with CLPA, then the details of the HOLD statement must be investigated to determine what kind of actions are required to activate the SYSMOD.

MSGSKEL
This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.

If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see *z/OS MVS Planning: Operations*.

If you want to use **only** the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.

MULTSYS
Identifies fixes that need to be applied to multiple systems, in one of three cases: preconditioning, coexistence, or exploitation.

RESTART
To become effective, the SYSMOD requires a special subsystem restart operation. The HOLD statement contains information regarding the required restart actions.

BYPASS(HOLDUSER)
indicates that exception SYSMODs associated with the specified user reason IDs should not be held. The list of reason IDs is optional. If you include one, only the reason IDs specified on it are bypassed. If you do not include a list, all user reason IDs are bypassed.

BYPASS(ID)
indicates that SMP/E should ignore any errors it detects when checking the SYSMOD's RMID and UMIDs. BYPASS(ID) should be used with caution and only after careful consideration because ignoring RMID and UMID errors may regress modifications included in previously accepted SYSMODs.

BYPASS(IFREQ)
indicates that SMP/E should ignore any conditional requisites that are missing.

BYPASS(PRE)
indicates that SMP/E should ignore any prerequisites that are missing.

BYPASS(REQ)
indicates that SMP/E should ignore any requisites that are missing.

BYPASS(XZIFREQ)
indicates that SMP/E is to continue ACCEPT processing for a SYSMOD, even if SMP/E detects a missing cross-zone requisite. SMP/E will identify such missing cross-zone requisites with a warning message, instead of terminating the ACCEPT processing.

BYPASS(XZIFREQ(*list*))
indicates that SMP/E is to continue ACCEPT processing for a SYSMOD, even if SMP/E detects a missing cross-zone requisite, provided that the missing requisite SYSMOD is included in the list provided with the XZIFREQ option.

Each entry in the list must be in one of the following formats:

- *sysmod_id*
- (*sysmod_id*,*zonename*)

sysmod_id
Indicates that SMP/E is to continue ACCEPT processing, even if the requisite *sysmod_id* is missing in any zone.

(sysmod_id,zonename)
Indicates that SMP/E is to continue ACCEPT processing, even if the requisite *sysmod_id* is missing from either:

- the set-to zone. In this case, SYSMODs in zone *zonename* require that *sysmod_id* be installed in the set-to zone.
- the zone identified by the *zonename* parameter. In this case, SYSMODs in the set-to zone require that *sysmod_id* be installed in zone *zonename*.

Note: A cross-zone requisite relationship necessarily involves two zones (the set-to zone and another zone) and two SYSMODs (the SYSMOD making the requirement and the requisite SYSMOD), with the requiring SYSMOD being in one zone and the requisite SYSMOD in the other zone. The zone specified in the (sysmod_id,zonename) pair must never be the set-to zone. It must always be a zone that has a requisite relationship with the set-to zone.

Each entry in the list must be unique. Also, a SYSMOD ID must not appear both by itself and as part of a SYSMOD/zone pair. However, a SYSMOD ID may appear in multiple SYSMOD/zone pairs, provided each of the pairs is unique.

The list provided must not be a null list; that is, BYPASS(XZIFREQ()) is not allowed.

CHECK
indicates that SMP/E should not actually update any libraries. Rather, it should just take these actions:

- Test for errors other than those that could occur when the libraries are actually updated.
- Report on which libraries are affected.
- Report on any SYSMOD that would be regressed.

COMPRESS
indicates which distribution libraries should be compressed.

- If you specify ALL, any libraries in which elements will be installed by this ACCEPT command are compressed.
- If you specify particular ddnames, those libraries are compressed regardless of whether they will be updated.

Note:

1. COMPRESS can also be specified as C.
2. If you specify both COMPRESS and CHECK, COMPRESS is ignored. This is because SMP/E does not update any data sets for CHECK.

EXCLUDE
Specifies one or more SYSMODs that should not be accepted.

Note:

1. EXCLUDE can also be specified as E.
2. If a SYSMOD is specified on the EXCLUDE operand, SMP/E does **not** include this SYSMOD, even though it might be specified on the GROUP or GROUPEXTEND operand.
3. SYSMODs that are superseded by another SYSMOD being accepted are not excluded even if they are specified on the EXCLUDE operand.

EXSRCID
indicates that SYSMODs associated with the specified source IDs should **not** be accepted.

Note:

1. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are excluded.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU* are excluded.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are excluded.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are excluded.
     - A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for example, SYSMODs that contain any of these source IDs are excluded: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not excluded.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source ID specifications:

   ```
   RSU0709
   RSU*
   IBM.Device.20%4
   IBM.Device.*.zAAP
   ```

2. A given source ID can be explicitly specified **only once** on the EXSRCID operand.
3. The same source ID **cannot** be explicitly specified on both the EXSRCID and SOURCEID operands.
4. If a source ID is specified implicitly or explicitly on the EXSRCID operand and is also specified either implicitly or explicitly on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs and at least one of those source IDs is specified either implicitly or explicitly on the SOURCEID operand, the SYSMOD is excluded from processing if another one of its source IDs is specified either explicitly or implicitly on the EXSRCID operand.

   For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.
6. If a SYSMOD that would have been included by the GROUP or GROUPEXTEND operand is excluded by the EXSRCID operand, SMP/E does not include it.
7. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

FIXCAT
identifies the list of fix categories of interest for command processing. This list determines which fix category APARs must be resolved for the SYSMODs being accepted.

A fix category APAR provides a fix for a held SYSMOD and the APAR is associated with one or more fix categories. Fix category APARs are identified by FIXCAT HOLD entries. If a fix category specified on a FIXCAT HOLD for a SYSMOD being accepted matches any of those specified on the FIXCAT operand of the command, then the SYSMOD is held for the APAR reason ID from the FIXCAT HOLD and will not be accepted until the APAR is resolved. If a fix category specified on a FIXCAT HOLD for a SYSMOD being

accepted does not match any of those specified on the FIXCAT operand of the command, or if the list of fix categories is null, the SYSMOD is not held for the APAR reason ID from the FIXCAT HOLD.

The values specified on the FIXCAT operand will override the list of values, if any, defined by the FIXCAT subentry in the active OPTIONS entry. FIXCAT() can be used to specify a null list, which means no fix category APARs must be resolved during current accept processing.

Fix category values can be 1 to 64 characters in length and can contain any nonblank character in the range X'41' - X'FE' except the single quotation mark, comma, left parenthesis, and right parenthesis. They can be specified in two ways:

- Explicitly, by fully specifying a particular fix category value. For example, IBM.Device.zIIP. In this case, all HOLDDATA associated with this fix category is applicable to command processing.
- Implicitly, by partially specifying a fix category value using any number of asterisks (*) as global characters and percent signs (%) as placeholders.
  - A single asterisk indicates that zero or more characters can occupy that position. For example, IBM.Device*, *z/OS or IBM*z/OS. In the first case, all HOLDDATA associated with a fix category that begins with the character string IBM.Device is applicable. In the second case, all HOLDDATA associated with a fix category that ends with the character string z/OS is applicable. In the third case, all HOLDDATA associated with a fix category that begins with the character string IBM and ends with the character string z/OS is applicable.
  - A single percent sign indicates that any one single character can occupy that position. For example, IBM.Device.20%4. In this case, HOLDDATA associated with any of the following fix categories is applicable: IBM.Device.2084, IBM.Device.2094, and IBM.Device.20t4. HOLDDATA associated with fix category IBM.Device.20914, however, is not applicable.

Fix category values can contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified Fix category value. For example, a specified value of IBM.FUNCTION.HEALTHCHECKER matches the value of IBM.Function.HealthChecker.

Fix category values are defined by FIXCAT HOLD entries. The following examples of acceptable fix categories are based on the fix category values that are used by IBM in FIXCAT HOLD entries:

```
IBM.Device.2094.zAAP
*
IBM.Function*
IBM.Device.20%4.*
*.HealthChecker
```

FORFMID
indicates that only SYSMODs for the specified FMIDs or FMIDSETs should be accepted.

Note:

1. Functions containing a ++VER DELETE statement are not automatically included by the FORFMID operand. You must specify them on the SELECT operand.
2. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

FUNCTIONS
indicates that all eligible functions should be accepted.

Note:

1. FUNCTIONS can also be specified as FUNCTION.
2. If FUNCTIONS is specified along with SELECT, all eligible functions are included in addition to the SYSMODs specified on SELECT.
3. If FUNCTIONS is specified along with SOURCEID, all functions associated with the specified source IDs are included.
4. Functions that contain a ++VER DELETE statement are not automatically included by the FUNCTIONS operand. You must specify them on the SELECT operand.

GROUP
indicates that if any SYSMODs specifically defined as requisites for eligible SYSMODs have not yet been accepted, SMP/E should automatically include them.

Note:

1. GROUP can also be specified as G.
2. GROUP is mutually exclusive with GROUPEXTEND.
3. GROUP might include SYSMODs at a higher service level than the level specified by the SOURCEID operand.
4. If you specify GROUP without any other SYSMOD selection operands (such as a SYSMOD type, SOURCEID, FORFMID, or SELECT), GROUP is ignored.
5. Processing done for SYSMODs specified on the SELECT operand is not necessarily done for SYSMODs included by the GROUP operand. For example, if REDO is specified, only SYSMODs specified on the SELECT operand can be reaccepted; SYSMODs included by the GROUP operand are not.
6. Functions containing a ++VER DELETE statement are not automatically included by the GROUP operand. You must specify them on the SELECT operand.
7. If a SYSMOD that would have been included by the GROUP operand is excluded by the EXCLUDE or EXSRCID operand, SMP/E does not include it.

GROUPEXTEND
indicates that if a SYSMOD specifically defined as a requisite for an eligible SYSMOD has not been accepted and cannot be processed for one of the reasons shown in Table 3 on page 14, SMP/E should automatically include a superseding SYSMOD. Table 3 on page 14 shows what GROUPEXTEND includes, depending on why the requisite cannot be processed.

Table 3. What GROUPEXTEND includes (ACCEPT processing)

For a requisite that is:	**GROUPEXTEND includes:**
• Held for an error reason ID	• A SYSMOD that supersedes the requisite or • A SYSMOD that matches or supersedes the error reason ID
One of these reasons: • Held for a system reason ID • Held for a user reason ID • Accepted in error • Not available	• A SYSMOD that supersedes the requisite

You can specify NOAPARS or NOUSERMODS (or both) to limit the types of SYSMODs that are included by GROUPEXTEND to resolve error reason IDs. The default is to include all eligible SYSMODs, regardless of SYSMOD type.

NOAPARS
indicates that SMP/E should exclude APARs that resolve error reason IDs.

NOUSERMODS
indicates that SMP/E should exclude USERMODs that resolve error reason IDs.

Note:

1. GROUPEXTEND can also be specified as GEXT.
2. GROUPEXTEND is mutually exclusive with GROUP.
3. If you specify both BYPASS and GROUPEXTEND, SMP/E does not include any superseding SYSMODs needed to take the place of requisites or error reason IDs that have been bypassed.

During CHECK processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify GROUPEXTEND without BYPASS.

4. GROUPEXTEND might include SYSMODs at a service level higher than that specified by the SELECT or SOURCEID operand.
5. Functions and excluded SYSMODs are not automatically included by GROUPEXTEND.
6. Processing done for SYSMODs specified on the SELECT operand is not necessarily done for SYSMODs specified by the GROUPEXTEND operand. For example, if RED0 is specified, only SYSMODs specified on the SELECT operand are reaccepted; SYSMODs included by the GROUPEXTEND operand are not.
7. If a SYSMOD that would have been included by the GROUPEXTEND operand is excluded by the EXCLUDE or EXSRCID operand, SMP/E does not include it.
8. When GROUPEXTEND is specified, SMP/E examines more SYSMODs than it does if GROUP were specified. Because of this additional processing, the ACCEPT command runs longer than if GROUP was specified, and a larger region size might be needed. On the other hand, GROUPEXTEND reduces the amount of time you would otherwise spend searching for missing requisites.

JCLINREPORT
indicates that SMP/E is to write the JCLIN reports after processing inline JCLIN. This is the default when inline JCLIN is processed at ACCEPT time (ACCJCLIN is set in the DLIBZONE entry).

Note: JCLINREPORT can also be specified as JCLR.

NOJCLIN
indicates that SMP/E should not process inline JCLIN for the specified SYSMODs. For example, if you are reaccepting SYSMODs, you may not want to process inline JCLIN that would change distribution zone entries that should not be changed.

If you include a list of SYSMOD IDs, SMP/E skips JCLIN processing only for the specified SYSMODs. If you do not include a list of SYSMOD IDs, SMP/E skips JCLIN processing for all SYSMODs.

Note: If inline JCLIN is not being processed at ACCEPT time (ACCJCLIN is not set in the DLIBZONE entry), you do not need to specify NOJCLIN.

NOJCLINREPORT
indicates that SMP/E should not write any JCLIN reports after processing inline JCLIN.

Note: NOJCLINREPORT can also be specified as NOJCLR.

PTFS
indicates that all eligible PTFs should be accepted.

Note:

1. PTFS can also be specified as PTF.
2. PTFS is the default SYSMOD type for mass-mode processing. If no other SYSMOD types are specified, only PTFs are processed, even if PTFS was not specified.
3. If PTFS is specified along with SELECT, all eligible PTFs are included in addition to the SYSMODs specified on SELECT.
4. If PTFS is specified along with SOURCEID, all PTFs associated with the specified source IDs are included.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ACCEPT command.

Before SMP/E processes the ACCEPT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ACCEPT command. Otherwise, the ACCEPT command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ACCEPT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

REDO
indicates that if any SYSMOD specified on SELECT has already been successfully accepted, it should be reaccepted.

Note:

1. If you specify REDO, you must also specify SELECT.
2. If GROUP or GROUPEXTEND is also specified, REDO does not reaccept SYSMODs included by the GROUP or GROUPEXTEND operand. It only processes SYSMODs specified on the SELECT operand.
3. When reaccepting a function SYSMOD, be sure to also reaccept all PTFs, APARs, and USERMODs for the same FMID that have already been accepted to prevent intersecting elements from being regressed. Otherwise, the correct service level of the intersecting elements may not be installed.

RETRY
indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

YES
indicates that SMP/E should try to recover and should retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If retry processing does not reclaim sufficient space and input to the utility was batched (copy or link-edit utility only), SMP/E debatches the input and retries the utility for each member separately. If this final attempt fails, the resulting *x*37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see *z/OS SMP/E User's Guide*. For more information about OPTIONS entries, see *z/OS SMP/E Reference*.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if YES is specified.

NO
indicates that SMP/E should not try to recover from the error.

REUSE
indicates that if a module was successfully assembled during previous SMP/E processing, it should not be reassembled. Instead, the existing object module from SMPWRK3 should be reused.

Note: The REUSE operand must be used with great care. SMP/E does not ensure that the same set of SYSMODs are being processed after a failure. If new maintenance is received after the initial ACCEPT command and before the ACCEPT REUSE command, SMP/E may use the wrong level of object modules.

SELECT
Specifies one or more SYSMODs that should be accepted.

You may specify any combination of individual SYSMOD IDs and FMIDSET names, provided that there are no duplicate values. For each FMIDSET specified, all FMIDs defined in the FMIDSET are processed as if they were explicitly specified in the SELECT list.

Note:

1. SELECT can also be specified as S.
2. To reaccept a SYSMOD, it is not enough to specify that SYSMOD on the SELECT operand. You must also specify REDO.
3. To process functions containing a ++VER DELETE statement, you must specify them on the SELECT operand.

4. When using FMIDSETs on the SELECT operand, remember that:
   - A value specified in the SELECT list is processed as an FMIDSET if the GLOBAL zone contains an FMIDSET entry by that name.
   - A value specified in the SELECT list is processed as a SYSMOD ID if it is not defined as an FMIDSET in the GLOBAL zone and it is a valid SYSMOD ID.
   - If the value in the SELECT list is valid both as a SYSMOD ID and as an FMIDSET name, it is processed (for SELECT) as an FMIDSET. If you want to select a SYSMOD that has the same name as an FMIDSET, you must define that SYSMOD in an FMIDSET and then include that FMIDSET name in the SELECT list.

     If this same value is specified on the EXCLUDE operand, it is processed as a SYSMOD ID (because only SYSMOD IDs are valid on EXCLUDE) and will **not** be rejected as a duplication of the identical FMIDSET name in the SELECT list.
   - Any given value (whether it represents a SYSMOD ID, an FMIDSET, or both) may **not** appear more than once in the SELECT list.
   - Any given SYSMOD ID may not simultaneously appear in both the SELECT and EXCLUDE lists, unless it is also a valid FMIDSET name.
   - A SYSMOD ID may be explicitly specified in the SELECT list and also included in an FMIDSET that is also specified in the SELECT list, provided the SYSMOD ID does not have the same name as the FMIDSET. The duplicate SYSMOD ID is ignored.

SOURCEID
indicates that SYSMODs associated with the specified source IDs should be accepted.

Note:

1. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are selected.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU are selected.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are selected.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are selected.
     - A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for example, SYSMODs that contain any of these source IDs are selected: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not selected.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source IDs:

```
RSU0709
RSU*
IBM.Device.20%4
IBM.Device.*.zAAP
```

2. A given source ID can be explicitly specified **only once** on the SOURCEID operand.
3. The same source ID **cannot** be explicitly specified on both the EXSRCID and SOURCEID operands.
4. If a source ID is specified implicitly or explicitly on both the SOURCEID operand and the EXSRCID operand, all SYSMODs with that source ID are excluded from processing.

5. If a given SYSMOD has multiple source IDs and at least one of those source IDs is specified either explicitly or implicitly on the SOURCEID operand, and another one is specified either explicitly or implicitly on the EXSRCID operand, the SYSMOD is excluded from processing.

   For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify `SOURCEID(SMC*)` and `EXSRCID(PUT0703)`, the SYSMOD is excluded from processing.
6. Functions containing a ++VER DELETE statement are not automatically included by the SOURCEID operand. You must specify them on the SELECT operand.
7. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

USERMODS
indicates that all eligible USERMODs should be accepted.

Note:

1. `USERMODS` can also be specified as `USERMOD`.
2. If `USERMODS` is specified along with `SELECT`, all eligible USERMODs are included, in addition to the SYSMODs specified on SELECT.
3. If `USERMODS` is specified along with `SOURCEID`, all USERMODs associated with the specified source IDs are included.

XZGROUP(*list*)
indicates that you wish to override SMP/E's default method for determining the zones to be checked for cross-zone requisites.

You may specify either:

- A list of ZONESETs and zones that are to be used to establish the zone group for this command. Each value in the list must be a valid ZONESET or zone name.
- XZGROUP() to provide a null list, which means that no cross-zone requisite checking is to be done for this command. A null list is not valid if the XZREQ operand is also specified.

The XZGROUP operand always requires a list or null list. That is, `XZGROUP` (without parentheses) is not allowed.

Note:

1. If XZGROUP is specified, whatever ZONESETs the user specifies are used to establish the initial zone group, even if the set-to zone is not in a ZONESET and the XZREQCHK subentry is not set.
2. If no XZGROUP operand was specified on the ACCEPT command, SMP/E reads all ZONESET entries. If a ZONESET entry has its XZREQCHK subentry set to YES and it contains the set-to zone, then all the other zones within the ZONESET entry become part of the initial zone group for the ACCEPT command.
3. After the initial zone group is established, it is culled by removing all target zones for ACCEPT processing. In other words, only zones having the same type as the set-to zone are left in the final zone group used for cross-zone requisite checking.

XZREQ
indicates that SMP/E should install unsatisfied cross-zone requisites into the set-to zone.

XZREQ causes cross-zone requisites to become primary candidates for installation. To do this, SMP/E checks secondary zones in the currently established zone group for CIFREQ data that is applicable to functions installed or being installed into the set-to zone.

Note:

1. SYSMODs selected with the XZREQ operand are in addition to any SYSMODs selected with the FORFMID and SOURCEID operands.

2. If XZREQ is specified along with SELECT, the specifically selected SYSMODs are included along with any unsatisfied cross-zone requisites.
3. If FORFMID is specified, only cross-zone requisites for the specified FMIDs become primary candidates for installation.
4. When the XZREQ operand is specified without EXSRCID operand, FORFMID operand, the SELECT operand, or the SOURCEID operand, only unsatisfied cross-zone requisites become primary candidates.
5. If any SYSMOD types are specified, processing is limited to those SYSMOD types, except for those SYSMODs that might be needed to satisfy processing for these operands:
   - GROUP
   - GROUPEXTEND
   - SELECT
   - XZREQ
6. If the XZREQ operand is specified, the XZGROUP operand may not be specified as a null list.

Syntax notes

Figure 1 on page 20 shows how SMP/E chooses which SYSMODs to process, on the basis of the operands specified on the ACCEPT command.

- If you specify any of the operands in the top part of the chart, or if you do not specify the SELECT operand, SMP/E does **mass-mode** processing.
- If you specify the SELECT operand, SMP/E does **select-mode** processing.
- If you specify the SELECT operand plus operands from the top part of the chart, SMP/E does both **select-mode** and **mass-mode** processing.

For more information about select-mode and mass-mode processing, see “Candidate selection” on page 28.

Remember the following when coding the ACCEPT command:

- SMP/E accepts SYSMODs specified on SELECT, regardless of other ACCEPT operands (such as a SYSMOD type, SOURCEID, EXSRCID, or FORFMID). Therefore, if you want to accept a specific SYSMOD, you only need to specify the SYSMOD ID on SELECT. For example, to accept a specific APAR, you do not also have to include the APAR operand.

  Note: If you do specify a SYSMOD type along with SELECT, SMP/E accepts all SYSMODs of the specified type plus the selected SYSMOD.
- If you specify more than one SYSMOD type, a SYSMOD needs to match only one of the specified types.
- If the SOURCEID, FORFMID, and SYSMOD type operands are specified together, only those SYSMODs meeting all the conditions are accepted.

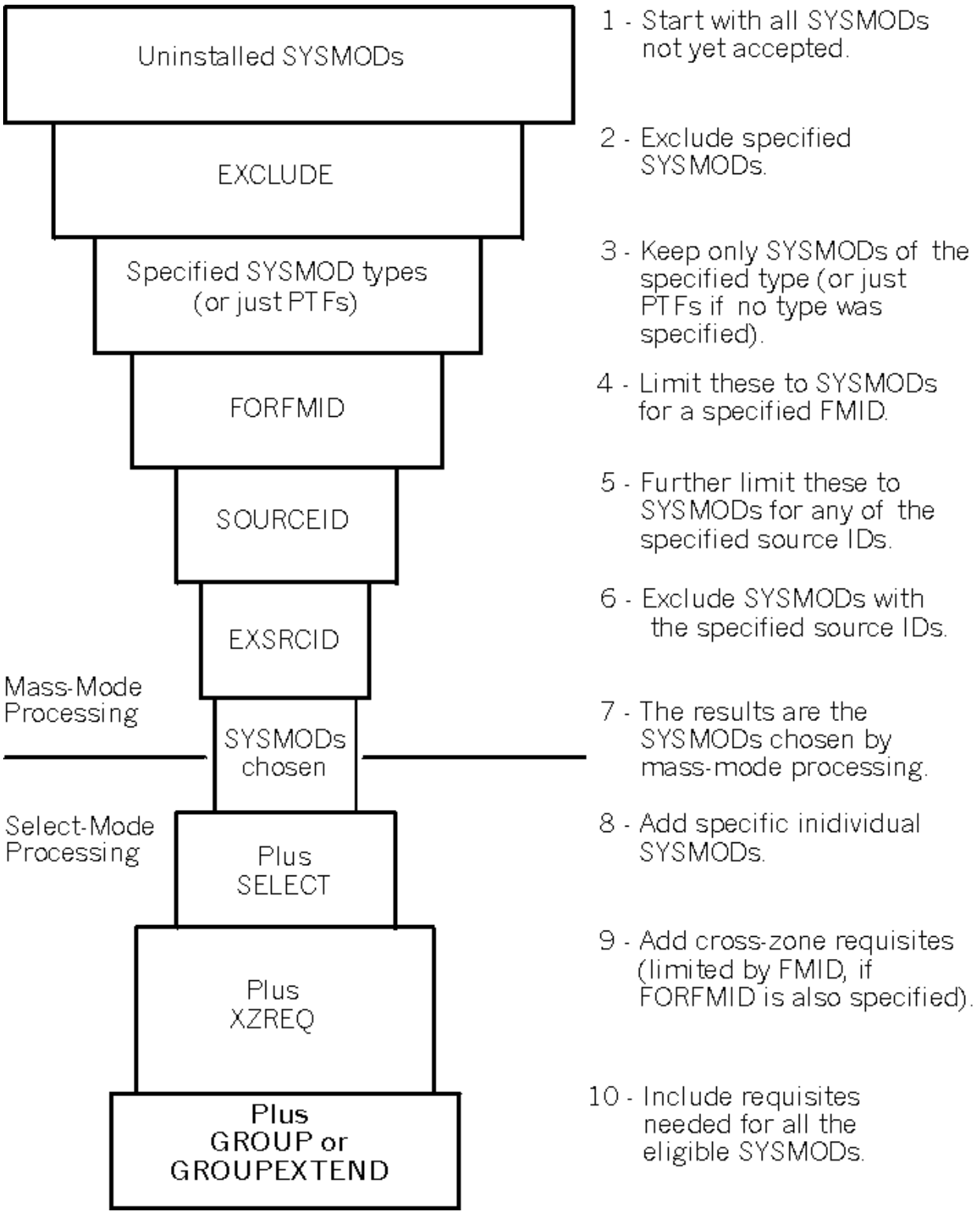


Figure 1. Combining SYSMOD selection operands on the ACCEPT command

Data sets used

These data sets might be needed to run the ACCEPT command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

Distribution library	SMPMTS	SMPTLIB	SYSLIB
Link library	SMPOUT	SMPWKDIR	SYSUT1
SMPCNTL	SMPPARM	SMPWRK1	SYSUT2
SMPCSI	SMPPTS	SMPWRK2	SYSUT3
SMPHRPT	SMPRPT	SMPWRK3	SYSUT4
SMPJHOME	SMPSCDS	SMPWRK4	Text library
SMPLOG	SMPSNAP	SMPWRK6	*zone*
SMPLOGA	SMPSTS	SYSPRINT	

Note:

1. SMPHRPT is an optional DD statement. If SMPHRPT is defined, the HOLD reports are directed there.
2. SMPJHOME is an optional DD statement used to specify the Java™ runtime directory. SMP/E requires the Java runtime when installing JARUPD elements.
3. SMPWKDIR is an optional DD statement used to specify a directory in the UNIX file system for the storage of temporary files created during SMP/E processing. If the SMPWKDIR DD statement is not provided, SMP/E will use the /tmp directory for temporary work files.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated using the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
5. SMPPARM is required only if exit routines have been defined in SMPPARM member GIMEXITS.

Usage notes

This section provides usage notes for the ACCEPT command.

Adding new elements to the distribution libraries

Any SYSMOD can introduce a new element to the distribution libraries without any processing outside the scope of SMP/E. To add a new element, you just have to identify the distribution library (that is, the DISTLIB operand on the appropriate modification control statement). SMP/E assumes the functional level to be that to which the SYSMOD is applicable, and the service level to be from the SYSMOD itself.

DISTLIB operand checking

When an element is selected to be installed and a distribution zone 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 distribution zone element entry. If the DISTLIB values are not equal, SMP/E issues a message to inform you of an error condition and terminates the SYSMOD containing the element.

If service and function SYSMODs containing the same element are being processed, and no element entry exists on the distribution zone, the service SYSMODs must specify the same DISTLIB as the function SYSMODs on the element modification control statements. If they do not, SMP/E issues an error message and the service SYSMOD is terminated.

If two service SYSMODs update or replace the same element, have different DISTLIB operand values, and are both eligible for processing, but no entry for the element exists on the distribution zone, the service SYSMODs must specify the same DISTLIB on the element . If they do not, SMP/E issues an error message and the service SYSMODs are terminated.

DISTSRC, ASSEM, and DISTMOD operands

Because SMP/E cannot determine from the data processed by JCLIN what source is contained in a totally copied library, the DISTSRC, ASSEM, and DISTMOD operands are provided to pass this information to SMP/E when a macro being replaced or updated, must cause the reassembly of source.

- The DISTSRC operand value specifies the name of the distribution library containing the source.
- The ASSEM and PREFIX operand values specify a list of sources that should be assembled during APPLY processing.
- The DISTMOD operand value specifies the name of the distribution library containing the load modules.

These four operands are specified on ++MAC and ++MACUPD statements. The DISTMOD operand is also specified on ++SRC and ++SRCUPD statements.

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

If neither a SRC nor an ASSEM entry exists for a module in the ASSEM operand values, a SRC entry is created. The DISTSRC operand value is placed in the SRC entry as the DISTLIB subentry.

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

After the macro update or replacement is accomplished, all modules specified in the ASSEM and PREFIX operand lists are assembled. If no member is found in the source distribution library or in the distribution library for a source 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 you specified in the RC subentry of the ASM UTILITY entry (or the SMP/E default of 4 if the RC subentry is null), the processing of the SYSMOD stops. If the resulting object text from a successful assembly can be link-edited into a load module, the link-edit is performed.

Alias processing

When an element with aliases is processed, both the element and its aliases are updated. SMP/E does not check the aliases against elements maintained in the distribution zone. The user must make sure an element's alias does not match the name of an element maintained by SMP/E in the distribution zone.

Aliases for an element are determined as follows:

- Replacement elements (MACs, MODs, data elements, and program elements):
  1. If a list of aliases is specified on the SMP/E modification control statement, these aliases are used. The new list replaces any alias subentries in the distribution zone element entry.
  2. If no list of aliases is specified on the SMP/E modification control statement, the aliases found as alias subentries in the distribution zone element entry are used.
- Update elements (ZAPs and MACUPDs):
  1. If a list of aliases is specified on the SMP/E modification control statement, these aliases are used. Any alias subentries in the distribution zone element entry are ignored for update processing of the element. Macro aliases (in the distribution library) existing before this list of aliases was presented to SMP/E are not updated. They remain in the distribution library. Alias subentries in the distribution zone element entry are not updated or replaced by the aliases in this list.
  2. If no list of aliases is specified on the SMP/E modification control statement, the aliases found as alias subentries in the distribution zone element entry are used.

ACCEPT CHECK facility

The intent of the CHECK option is to perform a test run informing you of possible error conditions and providing reports of SYSMOD status, libraries that will be updated, regression conditions, and SYSMODs that will be deleted. During CHECK processing, the list of distribution zone entries is maintained in storage; data is written to the distribution zone as a temporary storage medium. CHECK processing deletes any data written to the distribution zone. Consequently, no permanent updates are made to the distribution zone.

SYSMOD termination

Termination of a SYSMOD causes a return code of 8. Termination of a ++FUNCTION causes a return code of 12. Termination occurs in response to any of the following conditions:

- Missing requisites:
  - The requisite SYSMOD is not available on the PTS/CSI data sets. (It has not been received.)
  - The requisite SYSMOD has been excluded.
  - The requisite SYSMOD was terminated (possibly because of other missing requisites).
  - The requisite SYSMOD did not meet the applicability criteria.
  - The requisite SYSMOD was not included in the SELECT list, and neither GROUP nor GROUPEXTEND was specified.
  - GROUP was specified to include the requisite, but the requisite SYSMOD is being held or is not available on the PTS or CSI data sets. (It has not been received.)
  - GROUPEXTEND was specified to supersede the failing requisite, but a superseding SYSMOD was not available for processing.
- MODID error conditions.
- Attempting to change the ownership of an element that is being updated rather than replaced.
- DISTLIB operand checking failure.
- DD statement missing for a distribution library.
- Utility return codes: Return codes from the utilities called to update, assemble, copy, and link-edit elements to the distribution library are examined to determine the success or failure of an operation. If these return codes exceed a predefined value, the SYSMODs whose elements are involved in the operation are terminated. For details on handling *x*37 abends, see the description of the RETRY operand under “Operands” on page 7.
- Related SYSMOD failure: When SMP/E excludes an element from a SYSMOD because another SYSMOD being processed supplies a higher level of the element, SMP/E 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 are terminated because of a “related SYSMOD failure.”

Avoiding SYSMOD termination

BYPASS

Certain error conditions that cause the termination of a SYSMOD can be avoided by specifying the BYPASS operand on the ACCEPT command. 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
Indicates that SYSMODs should be processed even though their MODID verification checks have failed.

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

PRE
Indicates that SYSMODs should be processed even though their PRE requisite conditions are not met.

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

XZIFREQ
Indicates that SYSMODs should be processed even though their cross-zone requisite conditions are not met.

Utility return code thresholds

The value SMP/E uses to determine the success or failure of a called utility is kept in the UTILITY entries and can be changed by UCLIN.

ACCEPT termination

Termination can be caused by any of the following conditions. For each condition, SMP/E issues an error message:

- Termination of processing of any function SYSMOD.
- Two function SYSMODs are specified in the SELECT list and one specifies the other in the DELETE operand of its ++VER statement.
- Two function SYSMODs are specified in the SELECT list, or are selected in mass mode, and one specifies the other in the NPRE operand of its ++VER statement.
- A function SYSMOD specifying a previously accepted SYSMOD in the NPRE operand of its ++VER statement is specified in the SELECT list.
- A function SYSMOD specified in the SELECT list has been deleted by a previously accepted SYSMOD; that is, a SYSMOD entry on the distribution zone indicates that the SYSMOD has been deleted.
- A function SYSMOD specified in the SELECT list has been superseded by a previously accepted SYSMOD; that is, a SYSMOD entry on the distribution zone indicates that the SYSMOD is superseded. A service SYSMOD in the same situation is not processed, but the ACCEPT command is not terminated.
- A function SYSMOD is terminated before selection processing is complete. SMP/E issues a return code of 12 and does not produce a SYSMOD status report.

Automatic reinstallation of SYSMODs

The selection of a function SYSMOD that is being accepted for the first time may cause a SYSMOD that was accepted earlier to be selected for reinstallation. 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 this PTF was first accepted when only function GVT3100 was installed, the first ++VER statement would have been used and the conditional requisite data supplied on the ++IF would have been saved. If GVT3101 is subsequently installed, the saved ++IF data would require reinstallation of this same PTF.

Note: Because SMP/E does not process a SYSMOD with more than one VER that appears to be valid, GVT3101 must DELETE GVT3100 for this construction to work properly.

Output

The following reports may be produced during ACCEPT processing:

- Bypassed HOLD Reason Report
- Causer SYSMOD Summary report
- Cross-Zone Requisite SYSMOD report
- Deleted SYSMOD report
- File Allocation report
- Element Summary report
- JCLIN Cross-Reference report
- JCLIN Summary report

- MOVE/RENAME/DELETE report
- SYSMOD Regression report
- SYSMOD Status report
- Summary of Bypassed and Unresolved HOLD Reason Report
- Unresolved HOLD Reason Report

These reports are described in Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the ACCEPT command.

Example 1: Accepting all SYSMODs from a given source

If you used the SOURCEID operand during RECEIVE processing to group all the SYSMODs processed, you may choose to install only that set of SYSMODs. You can do this with the SOURCEID operand of the ACCEPT command. Suppose you received an ESO containing service levels PUT0701 and PUT0702. The ESO contained ++ASSIGN statements that assigned each PTF a SOURCEID value corresponding to the service level it is part of. Now you want to install all the applicable PTFs from those tapes into the distribution libraries described by zone MVSDLB1. You can do this with the following commands:

```
SET      BDY(MVSDLB1)        /* Process MVSDLB1 DLIB zone. */.
ACCEPT   SOURCEID(PUT0701,   /* Process these service      */
                  PUT0702)   /* levels                     */
         GROUP               /* and any requisites.        */.
```

Example 2: Accepting all SYSMODs for selected functions

At times, you may only want to install changes for a single function or for a certain group of functions. You can do this with the FORFMID operand on the ACCEPT command. Assume that you want to install service for function JXX1234 and for all the functions on your system that are related to telecommunication. You first need to define an FMIDSET for the telecommunication functions. You can do this with the following commands:

```
SET      BDY(GLOBAL)         /* Process global zone.       */.
UCLIN                        /* UCLIN to set up
                                FMIDSET.                   */.
ADD      FMIDSET(TC)         /* Define TC FMIDSET.         */
         FMID(JXX0001        /* Use these FMIDs.           */
              JXX0002)       /*                            */.
ENDUCL    save               /* End UCL set up.            */.
```

You can now use the following commands to install PTFs for function JXX1234 and for the functions in FMIDSET TP:

```
SET      BDY(MVSDLB1)        /* Process MVSDLB1 DLIB zone. */.
ACCEPT   FORFMID(JXX1234,    /* ACCEPT for selected FMIDs. */
                 TC)         /*                            */.
```

Example 3: Accepting with the GROUP operand

At times, you may know that a particular SYSMOD is required on your system, but you may not know all its requisite SYSMODs. You can use the GROUP operand of ACCEPT to have SMP/E determine all the requisites and install them automatically. This method is often used when a new function is being installed. Suppose you want to install a new function, HYY1234, with all its service and any requisite SYSMODs. You can do this with the following commands:

```
SET      BDY(MVSDLB1)        /* Process MVSDLB1 DLIB zone. */.
ACCEPT   FORFMID(HYY1234)    /* For one function.          */
         FUNCTIONS PTFS      /* Functions and PTFs         */
         GROUP               /* plus requisites.           */.
```

The FORFMID operand indicates that only SYSMODs applicable to this function should be installed. The FUNCTIONS operand indicates that HYY1234 can be installed. The PTFS operand indicates that only PTFs for HYY1234 should be installed (no APARs or USERMODs are included). The GROUP operand indicates that **all** requisite SYSMODs should also be accepted. These requisites can be applicable to other functions but may not be APARs or USERMODs.

Example 4: Accepting with the GROUPEXTEND operand

Assume that you want SMP/E to automatically include the requisites for some SYSMODs you plan to install. However, you are not sure whether all of the requisites are available. (They may not have been received, or they might be held because they are in error.) In these cases, you would like SMP/E to check whether a superseding SYSMOD is available for the unsatisfied requisites. To have SMP/E do this additional checking, you can use the GROUPEXTEND operand:

```
SET      BDY(MVSDLB1)       /* Process MVSDLB1 DLIB zone. */.
ACCEPT   FORFMID(HYY1234)   /* For one function.          */
         FUNCTIONS PTFS     /* Functions and PTFs plus    */
         GROUPEXTEND        /* requisites or supersedes.  */.
```

SMP/E accepts HYY1234 and any functions or PTFs applicable to HYY1234. Because of the GROUPEXTEND operand, SMP/E also accepts all requisites for those SYSMODs, even if the requisites are not applicable to HYY1234. If SMP/E cannot find a requisite, it looks for a SYSMOD that supersedes the requisite and uses it to satisfy the requirement. Likewise, if a requisite is being held for an error reason ID, SMP/E looks for a SYSMOD that supersedes the requisite, or that either satisfies or supersedes its error reason ID, and uses it to satisfy the requirement.

Example 5: Accepting with the CHECK operand

In Example 3, SMP/E was directed to automatically include SYSMODs needed for the selected function and service. To review which SYSMODs is included before you actually install them, you can use the CHECK operand of ACCEPT, as shown in the following commands:

```
SET      BDY(MVSDLB1)       /* Process MVSDLB1 DLIB zone. */.
ACCEPT   FORFMID(HYY1234)   /* For one FMID.              */
         FUNCTIONS PTFS     /* Functions and PTFs         */
         GROUP              /* plus requisites            */
         CHECK              /* in check mode.             */.
```

After running this command, you should check the SYSMOD Status Report to see which SYSMODs would have been installed if you had not specified CHECK. If the results of this trial run are acceptable, you can run the commands again without the CHECK operand to actually install the SYSMODs.

Example 6: Combining ACCEPT operands

You may want to further divide the work to be done by specifying combinations of the ACCEPT operands. The following is an example using all the SYSMOD selection operands of ACCEPT:

```
SET      BDY(MVSDLB1)       /* Process MVSDLB1 DLIB zone. */.
ACCEPT   SOURCEID(PUT0701   /* For these service levels.  */
                  PUT0702)  /*                            */
         FORFMID(HYY1234    /* For selected functions     */
                 TP)        /*                            */
         FUNCTIONS PTFS     /* install all type SYSMODs   */
         SELECT (UZ00001    /* plus these three SYSMODs   */
                 UZ00002    /* for other functions,       */
                 UZ00003)   /*                            */
         EXCLUDE(UZ00010    /* but not these three,       */
                 UZ00011    /*                            */
                 UZ00012)   /*                            */
         GROUP              /* plus all requisites.       */.
```

By issuing these commands, you direct SMP/E to accept all the SYSMODs from service levels PUT0701 and PUT0702 that are applicable either to function SYSMOD HYY1234 or to one of the function SYSMODs identified in the FMIDSET entry TP. Any other SYSMODs required to install those SYSMODs are also installed. Both FUNCTIONS and PTFS SYSMODs are eligible for selection. In addition, SYSMODs UZ00001,

UZ00002, and UZ00003 are accepted, even though they are not part of PUT0701 or PUT0702, and they may not belong to function SYSMOD HYY1234 or to FMIDSET entry TP. SMP/E does not install SYSMOD UZ00010, UZ00011, or UZ00012, even if they are requisites for other eligible SYSMODs.

Example 7: Doing ACCEPT before APPLY

Assume that you want to install PTFs from PUT0701 and PUT0702 into the distribution libraries to prepare for a full system generation. Therefore, you have not applied the SYSMODs before accepting them. To do this, you must use the BYPASS(APPCHK) operand to have SMP/E ignore whether the PTFs have been applied. You can use the following commands:

```
SET      BDY(MVSDLB1)       /* Process MVSDLB1 DLIB zone. */.
ACCEPT   SOURCEID(PUT0701   /* For these service levels   */
                  PUT0702)  /*                            */
         GROUP              /* plus all requisites.       */
         BYPASS(HOLDSYS     /* Okay to install SYSMODs    */
               (FULLGEN     /* that need SYSGEN or        */
                IOGEN)      /* IOGEN.                     */
                APPCHK)     /* Okay to accept before      */
                            /* apply.                     */.
```

Example 8: Installing service for all ESO service levels

Assume that you want to install the preventive service received from all ESO tapes into zone MYZONE1 without having to specify all possible ESO service levels on the SOURCEID operand. You can use the following commands:

```
SET      BDY(MYZONE1)       /* Process MYZONE1 DLIB zone. */.
ACCEPT   PTFS               /* Install all PTFs           */
         SOURCEID(PUT*)     /* for all service levels     */
         CHECK              /* in check mode.             */.
```

Example 9: Excluding SYSMODs with certain source IDs

Assume that you have received an ESO with PTFs up to service level PUT0703 and you now want to install service from all but the latest two service levels (PUT0702 and PUT0703) into zone MYZONE2. You can use the following commands:

```
SET      BDY(MYZONE2)       /* Process MYZONE2 DLIB zone. */.
ACCEPT   PTFS               /* Install all PTFs           */
         SOURCEID(PUT*)     /* for all service levels     */
         EXSRCID(PUT0702    /* except for PUT0702         */
                 PUT0703)   /* and PUT0703,               */
         GROUPEXTEND        /* and any requisites         */
         CHECK              /* in check mode.             */.
```

Example 10: Bypassing system reason IDs

Assume that you have received the SYSMODs for service level 0701. For some of them, ++HOLD statements specified a system reason ID of ACTION, indicating that you need to take certain actions before installing the SYSMODs (the required actions were described in the comments for the ++HOLD statements). You have completed the necessary actions for each SYSMOD, and have applied the SYSMODs and tested them to your satisfaction. Now you are ready to accept them. You can use the following commands:

```
SET      BDY(MYZONE2)        /* Process MYZONE2 DLIB zone. */.
ACCEPT   PTFS                /* Install all PTFs           */
         SOURCEID(PUT0701)   /* for service level 0701.    */
         BYPASS(             /* Bypass holds for all       */
         HOLDSYSTEM(ACTION)) /* SYSMODs held for ACTION.   */.
```

Suppose, instead, you have completed the necessary actions for only certain SYSMODs in service level 0701 (PTFs UZ12345 and UZ34567). You are ready to accept those specific held SYSMODs, but want the

other SYSMODs requiring actions to be held from ACCEPT processing. To limit the SYSMODs for which the hold is bypassed, specify the desired SYSMOD IDs with the ACTION reason ID:

```
SET      BDY(MYZONE2)         /* Process MYZONE2 DLIB zone. */.
ACCEPT   PTFS                 /* Install PTFs               */
         SOURCEID(PUT0701)    /* for service level 0701.    */
         BYPASS(              /* Bypass holds for specific  */
         HOLDSYSTEM(ACTION(   /* SYSMODs held for ACTION:   */
         UZ12345,UZ34567)))   /* List them here.            */.
```

Example 11: Excluding SYSMODs selected with an FMIDSET

A SYSMOD ID defined in an FMIDSET specified on the SELECT list may be excluded from processing with the EXCLUDE operand, as shown in this example:

If FMIDSTX contains FUNC001, FUNC002, FUNC003, and FUNC004, then, to ACCEPT all but FUNC003, the command would be:

```
ACCEPT SELECT(FMIDSTX) EXCLUDE(FUNC003)
```

Processing

Generally, ACCEPT processing is very similar to APPLY processing, except that the distribution zone, rather than the target zone, controls processing and the distribution libraries, rather than the target libraries, are updated.

SYSMOD selection

This section outlines the process by which SYSMODs and the elements from the SYSMODs are selected.

Operands related to SYSMOD selection

The following ACCEPT command operands can be used to specify to SMP/E which SYSMODs are to be processed:

APARS
BYPASS(APPLYCHECK)
EXCLUDE
EXSRCID
FORFMID
FUNCTIONS
GROUP
GROUPEXTEND
PTFS
SELECT
SOURCEID
USERMODS
XZREQ

Candidate selection

The SYSMOD selection operands of the ACCEPT command can be specified separately or in combination in order to control the SYSMODs that SMP/E is to process. When specified separately, SMP/E selects only those SYSMODs that meet the one selection criterion specified. When you specify them in combination, SMP/E does the following checking to build the complete candidate list:

1. SMP/E assumes that normal SYSMOD processing is:
   a. RECEIVE

b. APPLY

c. ACCEPT

Therefore, before SMP/E accepts a SYSMOD, it checks to make sure you have applied it. SMP/E does this checking by looking at the applicable target zone to see if the SYSMOD has been installed there (not by looking at any information in the global zone SYSMOD entry). A SYSMOD is considered installed in the target zone if the entry indicates the SYSMOD is applied or superseded. The applicable target zone is determined from the RELATED subentry in the distribution zone DZONE entry.

In some circumstances, you may want to accept a SYSMOD before it is applied—for example, when preparing the distribution libraries before doing a full system generation. In this case, you must specify the BYPASS(APPLYCHECK) operand, telling SMP/E not to check the target zone to make sure the SYSMOD has been applied.

Note: The BYPASS(APPLYCHECK) function was previously provided by the NOAPPLY operand, which is no longer supported.

2. SMP/E checks the global zone and the specified distribution zone to determine which SYSMODs in the global zone and SMPPTS have not already been accepted into the distribution zone.

   SMP/E checks each such SYSMOD to see if it meets the criteria of any additional selection operands.

   a. If you specify the EXCLUDE operand, SMP/E makes sure the SYSMOD was not specified in the exclude list.

   b. If you specify one or more of the SYSMOD-type operands (that is, FUNCTIONS, PTFS, APARS, or USERMODS), SMP/E checks to make sure the SYSMOD type was one of those specified.

      If you do not specify a SYSMOD-type operand, the default is for SMP/E to process only PTF SYSMODs.

   c. If you specify the FORFMID operand, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values contained in a specified FMIDSET.

   d. If you specify the SOURCEID operand, SMP/E makes sure one of the SOURCEIDs of the SYSMOD matches a source ID that you have specified, either explicitly or implicitly, on the SOURCEID operand.

   e. If you specify the EXSRCID operand, SMP/E makes sure none of the SOURCEIDs of the SYSMOD matches a source ID you have specified, either explicitly or implicitly, on the EXSRCID operand.

      Note: If a given SYSMOD has multiple source IDs and you specify at least one of them implicitly or explicitly on the SOURCEID operand, and another one explicitly or implicitly on the EXSRCID operand, that SYSMOD is excluded from processing.

      Similarly, if you specify a given source ID implicitly or explicitly on the EXSRCID operand and also implicitly or explicitly on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.

   Each SYSMOD that satisfies all of these conditions is considered a candidate for the ACCEPT process. In other words, by specifying the SYSMOD type operands, the FORFMID operand, or the SOURCEID operand in combination, you cause SMP/E to select only SYSMODs that meet all the specified conditions.

3. If you specify the SELECT operand, each SYSMOD specified in the select list is considered a candidate, regardless of its SYSMOD type, FMID value, or SOURCEID value. That is, SELECT has an additive effect on the SYSMOD selection. This is called *select-mode* processing. If you do not specify SELECT, this is called *mass-mode* processing.

   Note: If SELECT is the only operand you specify, only SYSMODs in the select list are processed.

4. If you specify the XZREQ operand, unsatisfied cross-zone requisites that are needed in the set-to zone become candidates for installation. These SYSMODs are in addition to other SYSMODs that are chosen due to other operands, such as FORFMID and SOURCEID. If FORFMID is specified, only cross-zone

requisites for the FMIDs specified on the FORFMID operand become candidates for installation. Other operands on the command, such as EXSRCID, have no effect on which cross-zone requisites become candidates for installation.

5. If you specify the GROUP or GROUPEXTEND operand, SMP/E checks each of the candidate SYSMODs to determine whether they have any requisites that must also be accepted. SMP/E automatically includes the following SYSMODs, as long as they are not specified on the EXCLUDE operand or excluded by the EXSRCID operand:

   - Any SYSMOD not already accepted and specified as a prerequisite (that is, specified in the ++VER statement PRE operand) of one of the candidate SYSMODs
   - Any SYSMOD not already accepted and specified as a corequisite (that is, specified in the ++VER statement REQ operand) of one of the candidate SYSMODs
   - Any SYSMOD not already accepted and specified as a conditional requisite (that is, specified in the ++IF statement REQ operand) of one of the candidate SYSMODs
   - Any SYSMOD not already accepted and specified as a conditional requisite (CIFREQ subentry) in the distribution zone SYSMOD entry for one of the candidate SYSMODs

   If one of these requisites is held or not available and you specified GROUPEXTEND, SMP/E checks the global zone for any SYSMODs that have been received and that either supersede the requisite, or that match or supersede its HOLDERROR reason ID. The lowest-level SYSMOD found is then used to satisfy the requisite.

   - If you specified NOAPARS with GROUPEXTEND, SMP/E does not include APARs that resolve error reason IDs for the held requisites.
   - If you specified NOUSERMODS with GROUPEXTEND, SMP/E does not include USERMODs that resolve error reason IDs for the held requisites.

   Once a SYSMOD is added to the candidate list, it is eligible to be checked for additional requisites. The FORFMID, SOURCEID, and SYSMOD type operands have no effect on SYSMODs brought in because of the GROUP or GROUPEXTEND operand. The following example accepts all those SYSMODs with a source ID of PUT0701 that are applicable to EBB1102, plus any additional SYSMODs that are required, even though their source ID is not PUT0701 or their FMID is not EBB1102:

```
SET      BDY(DLIB1)          /* Set to DLIB1 zone.       */.
ACCEPT   SOURCEID(PUT0701)   /*                          */
         FORFMID(EBB1102)    /*                          */
         GROUP               /*                          */.
```

Applicability checking

Once the ACCEPT candidate list is completed, SMP/E performs additional checking to make sure the selected SYSMODs are applicable to the system.

General applicability

SMP/E makes sure that each SYSMOD meets certain requirements before it is accepted by the system:

- Each SYSMOD must contain **at least one** ++VER statement whose SREL value matches one of the SREL values for that distribution zone and whose FMID value (if present) names a function SYSMOD that has been (or is being) accepted. Function SYSMODs having no ++VER statement FMID operand are applicable if the SREL matches.

  Each SYSMOD must have **at most one** ++VER statement whose SREL matches the SREL in the distribution zone entry and whose FMID value exists in the distribution zone (or is being accepted) and has not been superseded.

  If a SYSMOD is found that contains multiple applicable ++VER statements, SMP/E cannot accept that SYSMOD because SMP/E cannot determine which ++VER statement to use.

Note: If an FMID has been superseded or deleted, SMP/E does not consider it accepted. Therefore, a SYSMOD can contain two ++VER statements that apply to two functions, one of which supersedes the other.

- The SYSMOD must not have already been accepted successfully to the distribution zone.
  - To reaccept a SYSMOD, you must specify the SYSMOD in the SELECT operand and include the REDO operand on the ACCEPT command.
  - SYSMODs partially accepted during an earlier invocation are considered eligible for processing without any special action. These SYSMODs are identified by the ACCEPT ERROR indicator in their distribution zone SYSMOD entry.
- The SYSMOD must not have been partially restored during a previous SMP/E RESTORE attempt. A partially restored SYSMOD can be identified by the RESTORE ERROR indicator in its distribution zone SYSMOD entry.
- The SYSMOD must not have been superseded by an earlier SYSMOD. If a SYSMOD is found to be superseded by another SYSMOD being accepted, no elements are selected from the superseded SYSMOD.
- The SYSMOD must not have been explicitly deleted by a previous SYSMOD.

Unconditional requisites (PRE and REQ)

Unconditional requisites are SYSMODs that are required in all functional environments. Each SYSMOD must have all its unconditional requisites satisfied. Unconditional requisites are those specified in the SYSMOD's ++VER statement PRE and REQ operands of the SYSMOD. A requisite is considered satisfied if:

- The requisite SYSMOD is already accepted.
- The requisite SYSMOD is superseded by a SYSMOD already accepted.
- The requisite SYSMOD is being accepted.
- The requisite SYSMOD is superseded by a SYSMOD being accepted.

Conditional requisites (IFREQ)

Conditional requisites are SYSMODs required only for a particular functional environment. All conditional requisites of each SYSMOD must be resolved. Conditional requisites are specified on the ++IF statement immediately following the applicable ++VER statement. If the function specified in the FMID operand of the ++IF statement is accepted or is being accepted, each SYSMOD specified in the ++IF statement REQ operand must be satisfied. These requisites are satisfied in the same manner as unconditional requisites.

If the function specified in the FMID operand of the ++IF statement is not already installed, these requisites must be satisfied when the function is later installed. SMP/E, therefore, saves the information from the ++IF statement as CIFREQ subentries in the distribution zone SYSMOD entry for that function. When the function is accepted, SMP/E checks the CIFREQ subentries for any requisites of previously accepted SYSMODs and ensures that these requisites are satisfied.

Cross-zone requisites

Cross-zone requisites are very similar to conditional requisites. Like conditional requisites, they are also caused by an ++IF statement. For a cross-zone requisite, however, the SYSMOD containing the ++IF exists in one zone, but the function and SYSMODs identified by the FMID and REQ operands specified on the ++IF statement are in another zone.

Negative requisites (NPRE)

If the NPRE operand is specified on a the ++VER statement for a SYSMOD, the specified SYSMOD ID must not already be installed, must not be installed concurrently, and must not be superseded by a SYSMOD being installed concurrently.

Note: The NPRE operand is valid only in function SYSMODs and is used to specify one or more mutually exclusive functions.

Superseding SYSMODs (SUP)

SMP/E checks to make sure the SYSMOD has not been superseded by another SYSMOD that is already installed or by another SYSMOD being accepted concurrently. If the SYSMOD is superseded, it is not accepted, and the superseding SYSMOD is used instead.

Note: If the superseding SYSMOD is not processed because it is held or excluded or because some of its requisites are missing, processing continues as though the SYSMOD did not exist. The SYSMOD that would have been superseded is installed instead.

If a SYSMOD that is already accepted is being superseded, SMP/E ensures each of the elements contained in the superseded SYSMOD are also contained in either the superseding SYSMOD or in a SYSMOD from the requisite set for the superseding SYSMOD (unless the element is being deleted by the superseding SYSMOD).

Exception SYSMODs (HOLD)

SMP/E makes sure each SYSMOD has no unresolved exception data associated with it. Exception data is information specified on the ++HOLD statement. Each ++HOLD statement has a REASON operand specifying a character string that identifies the reason why the SYSMOD has been put into exception status. The following types of exception data are supported by SMP/E:

- HOLDERROR
- HOLDFIXCAT
- HOLDSYSTEM (internal and external)
- HOLDUSER

In addition, the ++HOLD statement may contain a CLASS operand, which specifies an alternative way to resolve exception data through the use of the BYPASS operand.

Exception data is considered resolved when one or more of the following are true:

- HOLDERROR and HOLDFIXCAT exception data is considered resolved if any of the following conditions apply:
  - The SYSMOD named as the reason ID for the exception is already applied or is superseded by a SYSMOD that is already applied.
  - The SYSMOD named as the reason ID for the exception is being applied concurrently or is being superseded by a SYSMOD being applied concurrently.
  - The applicable BYPASS operand is specified.
- HOLDSYSTEM (internal) exception data is considered resolved if
  - the SYSMOD ID specified on the ++HOLD defining the exception is found as a SYSMOD entry in the distribution zone OR
  - the SYSMOD ID specified on the ++HOLD defining the exception is being superseded by a SYSMOD being accepted concurrently OR
  - the applicable BYPASS operand is specified.
- HOLDSYSTEM (external) exception data is considered resolved if the applicable BYPASS operand is specified.
- HOLDUSER exception data is considered resolved if the applicable BYPASS operand is specified.

If all the exception data associated with a given SYSMOD is not resolved, SMP/E does not accept that SYSMOD. The SYSMOD is treated as though it had been specifically excluded. Messages are issued showing which exception data is not resolved, and the SYSMOD and reason IDs associated with the exception data are displayed in the SYSMOD Summary report.

Each category of exception data is resolved differently:

- For HOLDERROR exception data the reason ID is actually the number of the APAR that caused the SYSMOD to be placed in exception status. As subsequent service is processed, the APAR will probably be superseded by a PTF. When this happens, the exception data is resolved and the first PTF is automatically processed. Therefore, it is generally not necessary to use the BYPASS operand to process SYSMODs with error reason IDs.

  During any mass installation of SYSMODs, it should be expected that some SYSMODs are not accepted, because unresolved APARs are associated with them. During the installation of preventive service, these SYSMODs should not be investigated further; they will be installed later when a subsequent SYSMOD is produced that supersedes the reason ID associated with the exception data that is causing them to be held.

  During the installation either of corrective service (that is, installing a PTF or an APAR because of a known problem in the system) or of a new function specifically requiring a SYSMOD, the reason IDs associated with the HOLDERROR exception data should be taken as the first piece of data for research. Research may provide a fix for the problem, in which case the SYSMOD and the fix can be accepted concurrently. If a fix is not available, you can either wait for one, or accept the SYSMOD using the appropriate BYPASS operand.
- For HOLDFIXCAT exception data, the reason ID is the number of the APAR that caused the SYSMOD to be placed in the exception status. The APAR is associated with one or more fix categories. It is optional whether the HOLDFIXCAT exception data affects processing for the held SYSMOD, based on the fix categories of the HOLDFIXCAT and the fix categories of interest specified by the user. The fix categories of interest are specified on the FIXCAT operand or in the FIXCAT subentry of the active OPTIONS entry. If a fix category value on the HOLDFIXCAT exception matches any of those of interest to the user, then the held SYSMOD is not accepted until the APAR reason ID is resolved. If there are no matching fix categories, then the SYSMOD is not held for that HOLDFIXCAT exception.

  For HOLDFIXCAT exceptions that must be resolved, the APAR is considered resolved when any one of the following conditions is true:

  – The SYSMOD named as the reason ID (the APAR) is already accepted or has been superseded by a SYSMOD that is already accepted.
  – The SYSMOD named as the reason ID (the APAR) is being accepted concurrently or is being superseded by a SYSMOD that is being accepted concurrently.
  – An applicable BYPASS operand of HOLDCLASS or HOLDFIXCAT is specified.
- For HOLDSYSTEM (internal) exception data the SYSMOD ID specified on the ++HOLD modification control statement may be either

  – the SYSMOD ID of the containing SYSMOD OR
  – a SYSMOD ID of a SYSMOD superseded by the containing SYSMOD.

  When it is the latter, the reason that the current SYSMOD contains the ++HOLD is because it was originally in the SYSMOD whose SYSMOD ID appears on the ++HOLD and that SYSMOD has been superseded by the current SYSMOD. The exception data is considered resolved if the SYSMOD specified on the ++HOLD is already installed or is being superseded by another SYSMOD that is being installed concurrently with the held SYSMOD. When this is the case, it is assumed that the user has already addressed the reason for the hold.

  When it is the former, the held SYSMOD should be accepted by use of the BYPASS(HOLDSYS(reason_id)) operand once the reason for the hold is addressed.
- For HOLDSYSTEM (external) exception data the associated reason ID is a 1- to 7-character string used to identify some action that must be taken before or after a SYSMOD is installed. System reason IDs are not SYSMOD IDs and are not specified in the supersede list of a SYSMOD. Therefore, SMP/E does not automatically release them.

  SYSMODs held in this manner should be accepted by use of the BYPASS(HOLDSYS(reason_id)) operand. If you were to remove the system reason ID by using the ++RELEASE statement, you would then be able to install the SYSMOD, but you would also lose the information about any special processing required in order to accept that SYSMOD on another system.

- For HOLDUSER exception data the associated reason ID is a 1-to 7-character string meaningful to the user. User reason IDs are not SYSMOD IDs and are not specified in the supersede list of a SYSMOD. Therefore, SMP/E does not automatically release them.

  SYSMODs held in this manner should be accepted by use of the BYPASS(HOLDUSER) operand. If you were to remove the hold associated with user reason ID by using the ++RELEASE statement, you would then be able to install the SYSMOD, but you would also lose sight of the fact the SYSMOD has some special significance to the you, the user.

SYSMOD installation

After determining which SYSMODs are to be accepted and which elements should be selected from each SYSMOD, SMP/E begins actually installing these elements by performing the following tasks:

1. Determine the order in which the SYSMODs should be processed.
2. Perform delete processing for any SYSMODs in which the DELETE operand is specified on the ++VER statement.
3. Move specified elements, and delete or rename specified LMOD entries if appropriate.

   Note: Items 3 and 4 are combined and are done as each SYSMOD is processed.
4. Process any inline JCLIN.

   Note: Inline JCLIN is processed at ACCEPT time only if ACCJCLIN is set in the DLIBZONE entry.
5. Call system utilities to install the selected elements.
6. Update the applicable distribution zone entries.
7. Produce summary reports identifying all processing done.

The following sections describe each of these tasks in greater detail.

SYSMOD processing order

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

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

If no prerequisite order can be determined between SYSMODs, function SYSMODs are processed first, followed by service SYSMODs (PTFs, then APARs, then USERMODs).

Deleted SYSMODs

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

When one function SYSMOD deletes another, SMP/E attempts to remove all information on the deleted SYSMOD from the distribution zone. SMP/E also removes from the distribution libraries all elements that are currently owned by the deleted function SYSMOD. The following processing is done:

1. SMP/E determines whether there are any function SYSMODs in the hierarchy of the function SYSMOD being deleted, and considers those function SYSMODs also eligible for delete processing.
2. SMP/E deletes all SYSMODs that have an FMID value equal to one of the function SYSMODs being deleted.
3. SMP/E identifies all the elements that are currently owned by a function SYSMOD that is to be deleted.

4. SMP/E deletes from the distribution libraries all the elements of the SYSMODs to be deleted. If the elements have been successfully deleted, SMP/E deletes the entries for them from the distribution zone.
5. If the distribution zone contains an LMOD entry for a load module composed entirely of modules that are deleted, the LMOD entry is deleted.
6. If the load module contains modules not being deleted, the LMOD entry is not deleted.

   However, for each module deleted from the load module, SMP/E adds a MODDEL subentry for the module to the LMOD entry. MODDEL subentries document the connection between the deleted modules and the load module. If any of these deleted modules are ever reintroduced, an LMOD subentry is added to the MOD entries, and the MODDEL subentries are removed from the LMOD entry.
7. The distribution zone SYSMOD entries for all implicitly deleted SYSMODs are deleted. A distribution zone SYSMOD entry is created for each explicitly deleted SYSMOD. This entry has a DELBY subentry naming the function causing the deletion. The SYSMOD entries for the explicitly deleted SYSMODs prevent the deleted function SYSMODs from being reprocessed by ACCEPT.

   Note: Some functions may both delete and supersede another function. In this case, the SYSMOD entry contains a SUPBY subentry instead a DELBY subentry. This allows SYSMODs naming the deleted function as a requisite to be installed nevertheless.

The result of this process is that all SYSMODs within the hierarchy of the specified function SYSMOD are deleted.

In the example in Figure 2 on page 35, function SYSMODs HDE1203, HDE1303, and HDE1403, and service SYSMODs UZ00009, UZ00010 and UZ00004 are deleted, because DELETE(HDE1203) is specified on the ++VER statement.

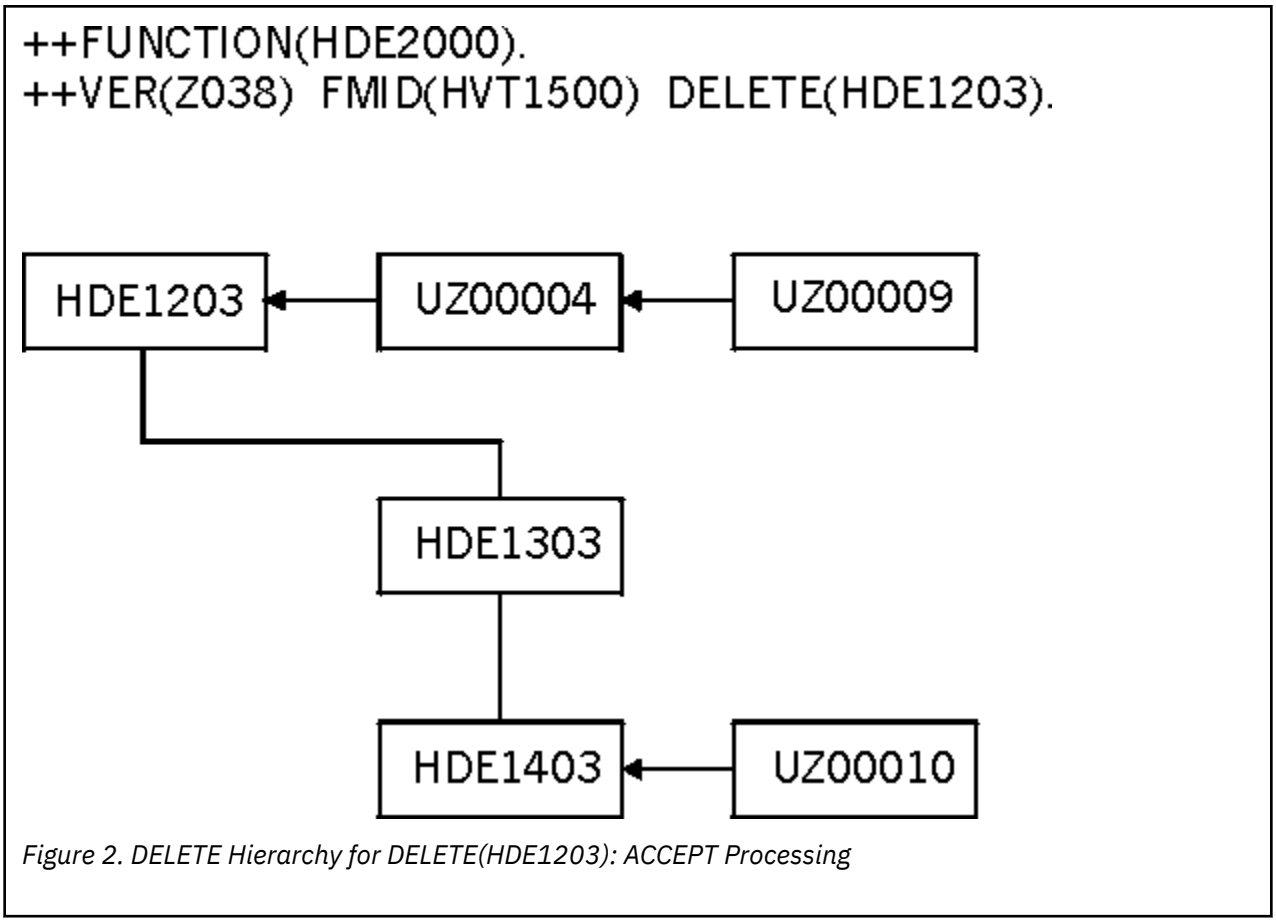


Figure 2. DELETE Hierarchy for DELETE(HDE1203): ACCEPT Processing

CIFREQ subentries in the SYSMOD entry for a function that is deleted (either explicitly or implicitly) are retained in the SYSMOD entry along with the DELBY subentry.

Thus, for the example in Figure 2 on page 35, when function HDE2000 is applied, CIFREQ subentries in the SYSMOD entry for function HDE1203 are retained, as are any CIFREQ subentries in the SYSMOD entries for functions HDE1303 and HDE1403. Any CIFREQ subentries for conditional requisites specified by the deleted SYSMODs are also retained in the appropriate SYSMOD entries.

Note: SMP/E assumes that when a function is deleted, the deleting function replaces all the required elements of the deleted function. Although you can build a function SYSMOD that does nothing but delete another function, it is your responsibility to make sure you still have a functionally complete system after the product has been deleted. One item commonly overlooked is the IHASU*xx* macros, which are used to indicate whether an SU has been installed. If you delete a product, thus causing its IHASU*xx* macro to be deleted, and do not replace that macro with your own version, indicating that the SU is not installed, you may lose the system generation capability for that system, because system generation requires that all IHASU*xx* macros be present.

During ACCEPT processing, when a function is deleted from a distribution zone by another function, its FMID is not removed from the FMID list in the global zone. This is because the deleted function may still be applied in other target zones or accepted in other distribution zones.

Inline JCLIN

Inline JCLIN can be saved for products without SYSGEN support to make building a new system easier. Inline JCLIN data for a SYSMOD is supplied following the ++JCLIN statement. In order to initialize the distribution zone, JCLIN is processed before elements. Later, when a system is built using the existing distribution libraries and the GENERATE command, this JCLIN data can be copied into the target zone. This eliminates the need for a separate step to obtain JCLIN information for products without SYSGEN support.

Remember the following restrictions when you plan to save inline JCLIN at ACCEPT time:

- Before using SMP/E to build the distribution libraries, you must set the ACCJCLIN indicator in the associated DLIBZONE entry. This tells SMP/E to save inline JCLIN in the distribution zone. If ACCJCLIN was not set when you first built the distribution libraries, it should not be set until the next time the libraries are built.
- After you install a product and set the ACCJCLIN indicator, you must keep ACCJCLIN set in the DLIBZONE entry. This makes sure that any time you accept service for that product, its JCLIN is updated in the distribution zone.
- The only way to save inline JCLIN in the distribution zone is through the ACCEPT command. The JCLIN command does not update the distribution zone.
- Saving JCLIN at ACCEPT does **not** take the place of a stage 1 SYSGEN for products that **do** have SYSGEN support.
- Because additional data is being saved in the distribution zone, the CSI data set containing the distribution zone may require more DASD space.

Note:

1. Inline JCLIN is not processed for superseded or deleted SYSMODs.
2. Inline JCLIN does **not** cause SMP/E to update the distribution libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in the SYSMOD. The element statements in the SYSMOD determine which elements should be installed.

The NOJCLIN operand on the ACCEPT command prevents the processing of inline JCLIN. NOJCLIN can be used if the JCLIN contains data that would overlay user-modified entries in the distribution zone.

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

For more information about JCLIN processing, see Chapter 9, “The JCLIN command,” on page 153.

Moving elements

Macros, modules, and source can be moved from one distribution library to another by use of the ++MOVE statement. This processing is done before element selection.

The ++MOVE statement is further described in the "SMP/E Modification Control Statements" section in *z/OS SMP/E Reference*.

Element selection

SMP/E uses the element statements provided in a SYSMOD to determine which elements should be installed in the distribution libraries. The selection of elements from a SYSMOD is based on relationships among SYSMODs being installed, other SYSMODs already installed, and modification identifiers of the corresponding elements installed in the distribution libraries. Three modification identifiers are kept for each element:

- FMID (function modification identifier): The FMID of an element is the function-type SYSMOD that owns the element. Generally, the FMID of an element is established (and later changed) by the installation of a function SYSMOD. The FMID of the element is the function SYSMOD that installed the element in the distribution libraries.
- RMID (replacement modification identifier): The RMID of an element is the last SYSMOD that replaced the element (or caused the FMID of that element to change). The RMID of an element is established by the SYSMOD that first introduces the element to the distribution libraries. The RMID of an element is changed by the installation of a SYSMOD that supplies a replacement for the element. An element can be replaced with an element defined by replacement , or with a module resulting from an assembly.
- UMID (update modification identifier): The UMIDs of an element are the set of SYSMODs that have installed updates to the distribution library element. A UMID is added to that set for each SYSMOD that installs an update to the element. Whenever a new replacement for the element is accepted, the set of UMIDs is cleared to start anew with subsequent updates installed for the new replacement. Element updates are ++ZAPs, ++MACUPDs, ++SRCUPDs, and ++JARUPDs.

  Note: Because data elements, hierarchical file system elements, and program elements can only be replaced and cannot be updated, they do not have UMIDs.

The purpose of element selection is to make sure that the correct functional level of each element is selected and that no service is inadvertently removed from the distribution libraries.

Element selection in SMP/E is divided into three cases:

- The FMID of the SYSMOD being installed matches the FMID of the element in the distribution libraries.
- The FMID of the SYSMOD being installed differs from the FMID of the element on the distribution libraries.
- A function SYSMOD is reinstalled.

The following topics describe processing for each case.

FMIDs match: MODID verification

In this case, SMP/E is dealing with elements belonging to the same function, and element processing is based on service relationships expressed by means of the PRE and SUP operands.

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

All elements

The SYSMOD being installed must specify the RMID of the associated distribution library element on the PRE or SUP operand of its ++VER MCS. If the RMID of the distribution library element is the same as its FMID, the element has not been replaced by any SYSMOD. Therefore, the SYSMOD being installed need not specify the RMID value in the PRE or SUP operand.

If the element being installed is a ++SRC/++SRCUPD, or ++MAC/++MACUPD element, resulting in an assembly that replaces a distribution library module, the SYSMOD being installed must specify, as one of its PRE or SUP operand values, the RMID of the corresponding distribution library module to be replaced by the assembly. If the distribution library module is itself the result of an assembly (RMIDASM indicator set in the MOD entry), an exception to this requirement is made, because the reassembly picks up any changes caused by the SYSMOD that last replaced the module through an assembly.

If the SYSMOD being processed does not specify the RMID of the element on the PRE or SUP operand of its ++VER MCS, SMP/E does not accept that SYSMOD, because doing so would regress the service supplied by the SYSMOD represented by the RMID. If you want to allow the RMID SYSMOD to be regressed, the BYPASS(ID) operand of the ACCEPT command can be specified.

Replacement elements

The SYSMOD being installed must be a prerequisite for, or must supersede, all UMIDs associated with the distribution library element.

Assemblies resulting from ++SRC/++SRCUPD and ++MAC/++MACUPD elements are considered to be replacement modules; the SYSMOD being installed must specify on the PRE or SUP operand all UMIDs of the corresponding distribution library modules to be replaced by the assembly. No exception is made for a SYSMOD that does not specify, on the PRE or SUP operand, all UMIDs associated with modules that have been assembled, because any UMIDs associated with the module are ZAPs that would be overlaid by a new assembly.

If the SYSMOD being processed does not specify each UMID of the element on the PRE or SUP operand of the ++VER MCS, SMP/E does not accept that SYSMOD, because doing so would regress the service supplied by the SYSMODs that the UMIDs represent. If you want to allow the regression to occur, you can use the BYPASS(ID) operand on the ACCEPT command. The MODID check condition is then reported as a warning, and the elements are installed on the distribution libraries.

Update elements

When processing ++SRC/++SRCUPD, and ++MAC/ +MACUPD elements, it is assumed that previous updates are still present and are incorporated with the current update. Therefore, a SYSMOD need not state a relationship (PRE or SUP) to a previous update.

The SYSMOD being installed need not specify each UMID of the element on the PRE or SUP operand of the ++VER MCS. If any element UMIDs in the distribution library are not specified in the SUP or PRE operands, 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 distribution library. The warning is given because SMP/E is unable to determine with certainty that the two modifications have a relationship or that there is an intersection. Thus, it is the responsibility of the developer or the service team (that is, whoever supplies the update type SYSMOD) to make sure that this SYSMOD specifies the correct relationships with all previous SYSMODs.

When processing ++JAR/++JARUPD elements, the SYSMOD must identify as a prerequisite or supersede the SYSMOD that last replaced the most recently selected ++JAR element (the RMID for the element). Therefore, the SYSMOD must specify as a PRE or SUP, the SYSMOD that last supplied a ++JAR for the element. In addition, the SYSMOD supplying the ++JAR/++JARUPD elements must identify as a prerequisite or supersede all SYSMODs which have previously updated the most recently selected ++JAR element (the UMID for the element). In other words, the current SYSMOD must specify as a PRE or SUP, all SYSMODs that have supplied ++JARUPDs for the element since it was last replaced.

FMIDs differ

In this case, SMP/E is dealing with elements belonging to different functions, and element selection is based on functional relationships expressed by means of FMID and VERSION. Elements may be excluded (that is, not selected), and processing of the SYSMOD continues under the assumption that a functionally higher version of the element is already installed on the distribution library.

An element is **excluded** from the SYSMOD being installed unless one of the following conditions is met:

- The function SYSMOD being installed names the FMID of the distribution library element in the ++VER FMID operand. In this case, the function being installed is superior to the function that owns the distribution library element; therefore, the element is selected.
- The modification control statement associated with the element from the SYSMOD being installed has a VERSION operand, and the FMID of the distribution library element is named in the VERSION list. In this case, the element from the SYSMOD being installed is considered to be functionally superior to the distribution library element, and it is selected.

  If there is no VERSION operand on the message control statement of an element, the SYSMOD IDs named in the VERSION operand on the ++VER are used as previously described.

  In this situation, SMP/E may be dealing either with a function SYSMOD or with a nonfunction SYSMOD that is changing the functional ownership (FMID) of the elements.

  Note: If a SYSMOD containing an element update (++SRCUPD, ++MACUPD, ++ZAP or ++JARUPD) attempts to change the ownership (FMID) of the element (with the VERSION operand), the SYSMOD cannot be installed.

When an element is selected, its FMID becomes that of the SYSMOD from which it is selected. No further MODID checking is done for these elements. SYSMODs are constructed so that when the functional ownership of a module changes, either the SYSMOD changing the ownership or data stored in the distribution zone SYSMOD entries (the CIFREQ data) contains sufficient information to prevent any service or functional regressions.

Reinstalling a function

Element selection gets more complicated only for **function** SYSMODs that are being reinstalled and have elements that intersect with corresponding elements having the same FMID as themselves (see "FMIDs match: MODID verification" on page 37).

The processing for this situation proceeds as in "FMIDs match: MODID verification" on page 37. When a MODID check error condition is detected, however, SMP/E checks to determine whether the service level of the distribution library element is higher than that of the element from the SYSMOD being reinstalled. If so, the element from the SYSMOD being reinstalled is not selected, and processing of the SYSMOD continues. If not, the SYSMOD is terminated with a MODID check error.

ACCEPT CHECK processing

If you specified the CHECK operand on the ACCEPT command, processing stops at this point. SMP/E produces all the usual ACCEPT reports, assuming that any SYSMOD not already reported as having a problem will be successful during the real ACCEPT run. These reports can be used to determine these situations:

- Missing DD statements
- Missing requisite SYSMODs
- Regressions
- SYSMOD in hold exception status
- Processing of inline JCLIN

  Note: Inline JCLIN is processed at ACCEPT time only if ACCJCLIN is set in the DLIBZONE entry.

Element installation

Once the proper SYSMODs have been selected and the proper functional and service level of each element has been determined, SMP/E begins the process of calling utility programs to get the elements installed in the appropriate distribution libraries. The following sections describe the process of installing each of the element types supported by SMP/E.

Compressing the distribution libraries

To have SMP/E compress the distribution libraries before installing SYSMODs, you can use the COMPRESS operand on the ACCEPT command. There are two ways to specify which libraries are to be compressed:

- List the specific libraries on the COMPRESS operand (including libraries that may not be affected by the ACCEPT command).
- Specify ALL on the COMPRESS operand; then only libraries in which elements will be installed by this ACCEPT command are eligible for compression.

Once the libraries have been determined, actual processing depends on the library type:

- For **source** libraries, any source that is to be replaced (not updated) by one of the SYSMODs being installed is deleted from the library.
- For **macro** libraries, no deleting is done because the SYSMOD replacing the macro might fail, and other SYSMODs might cause assemblies that use the macro.
- For **data element**, **hierarchical file system element**, and **program element** libraries, any data element, hierarchical file system element, or program element that is to be replaced by one of the SYSMODs being installed is deleted from the library.
- For **load** libraries, SMP/E determines which load modules within the library contain only modules being replaced by the SYSMODs being installed. Such load modules are deleted from the library. If a load module contains a module not being replaced, that load module is not deleted.

SMP/E then calls the copy utility to do the actual compress operation.

Note: The purpose of deleting the members before compressing the library is to reclaim as much space as possible before attempting to install the SYSMODs. SMP/E processes one library at a time, first deleting members, and then compressing it.

Macro replacements

One of the steps in actually updating the distribution libraries is to install macro replacements. Multiple SYSMODs can be accepted, each of which may contain a replacement for the same macro.

When two or more SYSMODs replacing the same macro are accepted concurrently, SMP/E determines the version at the highest function and service level. For a full explanation of how SMP/E determines the functional and service level of an element, see “Element selection” on page 37.

SMP/E then schedules the actual update of the distribution macro library. The library to be updated is determined from the DISTLIB information in the distribution zone MAC entry. (For further information about the initial setting of the MAC DISTLIB, see “Usage notes” on page 21.) The actual update is done by calling either the copy utility or the update utility.

- The copy utility is used in these cases:
  - The macro was packaged in a relative file (the RELFILE operand was specified).
  - The macro was packaged in a text library (the TXLIB operand was specified) and it had no alias names (that is, no MALIAS names are in the distribution zone MAC entry and no MALIAS operands are on the ++MAC statement).
  - The macro was packaged inline, it had no alias name, and the SSI operand was not specified.

  The SSI operand is ignored when TXLIB or RELFILE is specified.
- The update utility is called in these cases:
  - The macro was packaged in a text library, and the element had an alias name.
  - The macro was packaged inline and (1) it had an alias name, or (2) the SSI operand was specified.

Upon return from either utility, SMP/E issues a message indicating whether the macro has been replaced successfully.

Macro updates

After all the macro replacements have been done, any macro updates present can be scheduled. Again, multiple SYSMODs may contain updates for the same macro. When two or more updates to the same macro are being processed concurrently, the text from each is merged. Looking at the sequence numbers in columns 73 to 80, SMP/E processes the updates in the order expressed by the PRE operands on the ++VER statements, by internal defaults, or both, as follows:

- If SMP/E finds a processing order relationship between all of the SYSMODs being processed, the merge occurs according to that order.
- 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 do have a processing order relationship.
- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates by SYSMOD type: PTFs first, APARs second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to update the distribution library.

The library to be updated is determined from the DISTLIB information in the MAC entry for the distribution zone. For further information about the initial setting of the MAC DISTLIB, see "Usage notes" on page 21. Upon return from the update utility, SMP/E issues a message telling whether the update was successful.

Source replacements

Another step in actually updating the distribution libraries is to install source replacements. Multiple SYSMODs can be accepted, each of which can contain a replacement for the same source.

When two or more SYSMODs replacing the same source are accepted concurrently, SMP/E determines which version is at the highest function and service level. For a full explanation of how SMP/E does this, see "Element selection" on page 37.

SMP/E then schedules the actual update of the distribution source library. The library to be updated is determined from the DISTLIB information in the distribution zone SRC entry. (For further information about the initial setting of the SRC DISTLIB, see "Usage notes" on page 21.) The actual update is done by calling either the update utility or the copy utility.

- The copy utility is used if the source replacement was packaged, either in a text library (that is, the TXLIB operand was specified) or in a RELFILE (that is, the RELFILE operand was specified). The SSI operand is ignored when TXLIB or RELFILE is specified.
- The update utility is called if the source replacement was packaged inline and the SSI operand was specified.

Upon return from either utility, SMP/E issues a message telling whether the source was replaced successfully.

Source updates

After all the source replacements have been done, any source updates present can be scheduled. Again, multiple SYSMODs can contain updates for the same source. When two or more updates to the same source are processed concurrently, the text from each is merged. Looking at the sequence numbers in columns 73 to 80, SMP/E processes the updates in the order expressed by the PRE operands on the ++VER statements, by internal defaults, or both, as follows:

- If SMP/E finds a processing order relationship between all of the SYSMODs being processed, the merge occurs according to that order.
- 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 do have a processing order relationship.

- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates by SYSMOD type: PTFs first, APARs second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to update the distribution library. The library to be updated is determined from the DISTLIB information in the distribution zone SRC entry. (For further information about the initial setting of the SRC DISTLIB, see "Usage notes" on page 21.) Upon return from the update utility, SMP/E issues a message telling whether the update was successful.

Assemblies

This section discusses these topics:

- Assembling source
- Assemblies caused by macros
- Reusing previous assemblies

Assembling source

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

- Was ASSEM specified on the ACCEPT command?
- Are there any updates to the source that the SYSMOD supplying the object does not know about (UMIDs in the distribution zone SRC entry)?
- Has another SYSMOD assembled the module without this SYSMOD knowing about it (RMID of the distribution zone MOD entry for the module to be assembled)?
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of these questions is yes, the module is assembled if SMP/E can determine where the resultant assembled object module should go.

SMP/E knows where to put the assembled object module if there is a distribution zone MOD entry and a resultant object module. That is, the DISTLIB is not SYSPUNCH. (For more information on how SMP/E processes the object module after assembly, see "Module replacements" on page 43.)

Assemblies caused by macros

A SYSMOD may supply macros requiring the assemblies of modules. The required assemblies are found in the ASSEM and PREFIX operands on the ++MAC or ++MACUPD statement. The SRC entry matching the name of the module is used as the source for the assembly. If no SRC entry can be found, no assembly is done.

The SYSMOD may also supply an object deck for the modules to be assembled. To determine whether to do the assembly or to use the object decks supplied in the SYSMOD, SMP/E considers the following questions:

- Was ASSEM specified on the ACCEPT command?
- Are there any updates to the macro that the SYSMOD supplying the object does not know about (UMIDs in the distribution zone macro entry)?
- Has another SYSMOD assembled the module without this SYSMOD knowing about it (RMID of the distribution zone MOD entry for the module to be assembled)?
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of these questions is yes, the module will be assembled if SMP/E can determine where the resultant assembled object module should go. SMP/E knows where to put the assembled object module if there is a distribution zone MOD entry and a resultant object module, that is, if the DISTLIB is not SYSPUNCH. (For more information on how SMP/E processes the object module after assembly, see "Module replacements" on page 43.)

Whenever a macro modification in a APAR- or USERMOD-type SYSMOD causes a module to be assembled, the ASSEMBLE indicator in the module's distribution zone MOD entry is set on. If any subsequent PTF-, APAR-, or USERMOD-type SYSMODs contain modifications to macro or source elements affecting a module whose ASSEMBLE indicator has been set, they cause the module to be reassembled in spite of the presence of an object module in the SYSMOD. The reassembly prevents regression of the assembled module during the installation of subsequent SYSMODs that might replace the affected module but do not contain the macro modification introduced by the earlier SYSMOD.

To prevent future reassemblies of modules in which the ASSEMBLE indicator has been set, use the UCLIN function to set it off (DEL).

Reusing previous assemblies

If SMP/E is run after a failure, assemblies are rerun to ensure that the proper source and macros are used. If the same set of SYSMODs is being processed after a failure, the assemblies run before the failure need not be rerun. If the REUSE operand is specified in the ACCEPT command, the previously assembled objects for the failed SYSMOD are used.

Assembled object decks are stored on the SMPWRK3 data set and remain there until the SYSMODs causing the assemblies are successfully processed. To be able to reuse assemblies, SMPWRK3 must not be scratched after an ACCEPT step.

Note: SMP/E does not check to make sure the same SYSMODs are being rerun after a failure; the user must be careful when taking advantage of the REUSE facility.

Module replacements

The modules (++MODs and assemblies from ++SRCs) selected from a SYSMOD are link-edited directly into a distribution library. From the appropriate distribution zone MOD entry, SMP/E determines:

- The distribution library (based on the DISTLIB subentry)
- The link-edit characteristics (based on the LEPARM subentry)

Because there is a one-for-one correspondence between the object module being replaced and the members in the distribution library, there is no need to save any link-edit control cards in the distribution zone. When SMP/E link-edits a module into the distribution library, no link-edit include card is generated for the version of the module already there. Therefore, when you replace a distribution library module, you must provide SMP/E with the replacement pieces for all the parts of that module; that is, if the module contains multiple CSECTs, the SYSMOD must contain all the CSECTs, or the missing ones will be lost.

If the module is packaged in a link library (LKLIB) or a relative file (RELFILE), SMP/E copies the module to the distribution library. Any aliases specified for such a module must exist in the LKLIB or relative file in order to be copied.

Load module attributes and link-edit parameters

The parameters passed to the link-edit utility include load module attributes such as RENT, REUS, and REFR. SMP/E determines the attributes and parameters to be passed, as follows:

- If SMP/E has determined that the binder is available on the system and SMP/E is processing CSECT deletes, the STORENX link-edit parameter is passed to the link-edit utility.
- If LEPARM is specified on the ++MOD statement, the attributes supplied are used, and the corresponding distribution zone MOD entry is updated (or created) with these attributes.
- If LEPARM is not specified, SMP/E checks the following for link-edit attributes, in the order indicated, and passes the attributes that it finds:
  1. A distribution zone MOD entry containing link-edit attributes
  2. An LKLIB data set or SMPTLIB data set containing the module, if one is available from another SYSMOD that is in process
  3. The distribution library that is supposed to contain the module

If the module or its link-edit attributes are not found in any of the places indicated, the RENT, REUS, and REFR attributes are used.

The parameters passed to the link-edit utility are the attributes determined in the preceding steps, **plus** the link-edit utility parameters specified in the UTILITY entry for link-edit processing.

Note: SMP/E provides for special processing for any module with a distribution library of SYSPUNCH. This indicates to SMP/E that the module was assembled during the system generation process and that the assembly was stored in a temporary data set, the SYSPUNCH data set. After being used during the system generation process, that data set is deleted. Therefore, during accept, when SMP/E sees any module going to SYSPUNCH, no processing is done for that module.

Multitasking of link-edit utility invocations

When multiple output libraries must be updated for link-edit processing, SMP/E may initiate a separate subtask for each such library, provided that:

- The maximum number of such subtasks (10 subtasks) has not been reached.
- The link-edit utility being used is reentrant (the Binder is reentrant but the old linkage editor is not)
- The logical SYSPRINT file to be used for link-edit operations has either a DDDEF that specifies a SYSOUT class or an entry in the SMPPARM GIMDDALC member that specifies a SYSOUT class. The logical SYSPRINT is specified by the PRINT subentry for the link-edit UTILITY entry in effect. The default is SYSPRINT.

This multitasking of link-edit steps should result in a better elapsed time for the SMP/E process when many link-edits must be done to process a set of SYSMODs.

You can tell if multitasking of link-edit operations is occurring by looking at the link-edit completion messages being issued by SMP/E. If the message ends with "-- SYSPRINT FILE xxxxxxxx.", which indicates the ddname used for the link-edit output, then multitasking is being done.

Note: If message filtering is turned on (MSGFILTER=YES in the active OPTIONS entry), link-edit completion messages might only be written to SMPLOG but not to SMPOUT.

SMP/E ensures that libraries are processed in the right order for CALLLIBS consideration, even when multitasking of link-edit operations is in effect.

Module updates

For each ++ZAP element within a SYSMOD, SMP/E determines the distribution library for the module and then attempts to install the superzap to that library.

When installing the ZAP, SMP/E performs two passes: the first to process the VER control cards, and the second to process the REP control cards. The REP pass is performed only if the VER pass for the module is successful.

Replacements for other elements

Still another step in updating the distribution libraries is the installation of replacements for data elements, hierarchical file system elements, and program elements. Multiple SYSMODs can be accepted, each of which may contain a replacement for the same element. When two or more SYSMODs replacing the element are accepted concurrently, SMP/E determines which version is at the highest function and service level. For a full explanation of how SMP/E determines the functional and service level of an element, see “Element selection” on page 37.

SMP/E then schedules the actual update of the distribution library. The library to be updated is determined from the DISTLIB information in the distribution zone element entry. The actual update is done by either the copy utility or SMP/E.

Replacing data elements

Note: For a list of data element types, refer to the "Data element " section in *z/OS SMP/E Reference*.

The actual update to the library is done by either SMP/E or the copy utility.

- SMP/E updates the library in these cases:
  - The data element was packaged inline and has been transformed by GIMDTS. All control information is removed, the transformed data element is changed back to its original format, and the distribution library is updated with the element.
  - The data element must be reformatted to be compatible with the distribution library. For more information on reformatting data elements, see "Reformatting data elements" on page 98.
  - The distribution library is a sequential data set.
- The copy utility updates the library in all other cases. A COPY control statement is passed to the copy utility.

Replacing program elements

The actual update to the library is done by the copy utility. A COPYMOD control statement is passed to the copy utility.

If the program element was packaged in a relative file (the RELFILE operand was specified), or in a link library (the LKLIB operand was specified), then the copy utility is used to perform the replacement directly.

If a program element is packaged inline, SMP/E first retransforms the program element into its original VS or VBS format in a temporary data set. Then, if the distribution library and the data set that contained the original program element are of different types (that is, one is a PDS and the other a PDSE), SMP/E allocates a temporary SMPTLOAD data set of the same type as the data set that contained the original program element and uses the copy utility to reload a program element and its aliases to the SMPTLOAD data set. SMP/E then uses the copy utility to place the program element and its aliases into the distribution library. The input data set for the copy operation is the SMPTLOAD data set, if one was required, or the retransformed temporary data set, if an SMPTLOAD data set was not required.

At the end of processing, SMP/E issues a message indicating whether the program element has been replaced successfully.

Replacing hierarchical file System elements

The actual update to the library is done by either SMP/E or the copy utility. (The HFS copy utility is not used during ACCEPT processing.)

SMP/E updates the library if the hierarchical file system element was packaged inline and has been transformed by GIMDTS. All control information is removed, the transformed data element is changed back to its original format, and the distribution library is updated with the element.

The copy utility updates the library in all other cases. A COPY control statement is passed to the copy utility.

Java archive file replacements

During the element installation phase of ACCEPT processing, elements supplied by SYSMODs are installed into the distribution libraries in the same manner as hierarchical file system elements. After all hierarchical file system elements and JAR file element replacements have been copied to their respective distribution libraries, then SMP/E will process JAR file updates (++JARUPDs).

Java archive file updates

One or more ++JARUPDs may be processed for a single Java Archive (JAR) file, and the ++JARUPDs for that JAR file are processed in the order implied by the processing order of the SYSMODs which contain the ++JARUPDs. That is, if a SYSMOD contains a ++JARUPD and specifies a prerequisite for another SYSMOD that contains a ++JARUPD for the same JAR file, then the ++JARUPD from the prerequisite SYSMOD is processed first. If no processing order can be implied from the containing SYSMODs, then no particular order of processing for the ++JARUPDs is assumed.

Following are the steps required to process one or more ++JARUPDs for a single JAR file:

1. Create the primary temporary work directory in the SMPWKDIR directory for processing all ++JAR and ++JARUPD elements. If the SMPWKDIR directory is not specified on a DD statement or DDDEF entry, SMP/E will use the /tmp directory.
2. Copy the JAR file to be updated into the primary temporary work directory.
3. Create an update subdirectory for processing the ++JARUPD elements for this JAR file.
4. Copy a ++JARUPD element to the primary temporary work directory.
5. Extract the component files from the ++JARUPD element into the update subdirectory.
6. Repeat steps 4 through 5 for each ++JARUPD to be processed for this JAR file.
7. Update the JAR file with the component files from all the ++JARUPD elements.
8. Cleanup the ++JARUPD elements and all component files in the update subdirectory.
9. Copy the updated JAR file into its distribution directory.

Recording after completion

Results of processing are recorded in the following entries.

Distribution zone element entries

ACCEPT processing creates, modifies, and may delete distribution zone element entries.

- Entry update indicator: When an entry is added by a SYSMOD being processed, the SYSMOD's SYSMOD ID is placed in the LASTUPD subentry of the distribution zone element entry.
- ALIAS subentries: The updates to an element's ALIAS subentries are discussed under “Alias processing” on page 22.
- LMOD subentries: When the LMOD operand is specified on a ++MOD statement, the values in the operand list are added to the distribution zone MOD entry as LMOD subentries.
- MODID subentries:
  - The FMID subentry is replaced with the FMID of the SYSMOD from which the modification to the element was selected. If the SYSMOD is a function SYSMOD, the FMID is set to the SYSMOD ID of the function itself.
  - The RMID subentry is changed when a replacement element or assembly is accepted. The RMID is set to 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 for the module is set even if the actual assembly was suppressed because the SYSMOD supplied an assembled version of the module. (See “Source replacements” on page 41 and “Assemblies” on page 42 for further information.)

    For function SYSMODs, if the replacement element's modification control statement specified an RMID for the element (the RMID operand), the specified value is used.
  - All UMID subentries are deleted when a replacement element is accepted: For function SYSMODs, if the replacement element's message control statement specified a list of UMIDs for the element (the UMID operand), these UMIDs replace any existing UMIDs for the element.
  - UMID subentries are added when updates for the element are accepted. The UMIDs are the 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, the UMID subentry for the superseded SYSMOD is deleted from the element entry.
- CSECT names

  The CSECT information from the ++MOD statement is saved in the MOD entry for the distribution zone. The information saved is determined in the following manner:

- If the SYSMOD that the selected version of the module came from contained the CSECT operand, the CSECT names there are either added to the distribution zone MOD entry (if no CSECT information was already there) or are used to replace the existing list of CSECT names in the distribution zone MOD entry.
- If the SYSMOD that the selected version of the module came from did not contain the CSECT operand, SMP/E determines whether any other SYSMOD applicable to the same FMID was also being accepted. If so, and if any of those SYSMODs contained the CSECT operand, the CSECT information from the SYSMOD at the highest service level is used.

SMPSCDS BACKUP entries, SMPMTS MTSMAC entries, and SMPSTS STSSRC entries

For each SYSMOD successfully accepted, SMP/E deletes the following entries (if applicable):

- BACKUP entries from the SMPSCDS
- MTSMAC entries from the SMPMTS
- STSSRC entries from the SMPSTS

There is only one exception: If the BYPASS(APPLYCHECK) operand was specified, SMP/E assumes that the SYSMODs have not been applied and, therefore, there are no entries to delete.

Note: The correct SMPSCDS, SMPMTS, or SMPSTS data set to use is determined solely by a DD statement in the JCL used when SMP/E is invoked or by a DDDEF entry set up in the distribution zone. In both cases, the data set should be the one used for the related target zone named in the DZONE entry for the distribution zone.

Distribution zone SYSMOD entries

For each SYSMOD processed, a SYSMOD entry is created in the distribution zone. 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 statement, subentries for each of the elements included in the SYSMOD package, and indicators that are set when ++IF and ++JCLIN statements are present.

A SYSMOD is considered successfully processed when all of its selected elements have been accepted by the appropriate distribution libraries **and** all of its requisites have been successfully processed. Because SMP/E processes any number of SYSMODs with elements in common, some SYSMODs may have elements that need not be installed in a distribution library; 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.

SMP/E sets the REGEN indicator in the distribution zone when a SYSMOD is accepted. It does not set REGEN in the target zone when the SYSMOD is applied. However, if a distribution zone is copied into a target zone (for example, as part of a system generation), the REGEN indicator is copied into the target zone. Therefore, REGEN can help you determine how a SYSMOD was installed in the target libraries.

- If REGEN is set in the target zone SYSMOD entry, the distribution zone was copied into the target zone. The SYSMOD was, therefore, installed by use of a generation procedure, such as SYSGEN.
- If REGEN is not set in the target zone SYSMOD entry, the distribution zone was not copied into the target zone. The SYSMOD was, therefore, installed by use of the APPLY command.

Superseded SYSMODs

When one SYSMOD is superseded by another, SMP/E makes a record of this by adding the name of the superseding SYSMOD to the entry for the superseded SYSMOD.

- When there is only one superseding SYSMOD, its name is saved in the LASTSUP subentry.
- When there are several superseding SYSMODs, the name of the last one is saved in the LASTSUP subentry, and a complete list is saved in the SUPBY subentry.

If the superseded SYSMOD has not been previously accepted, its distribution zone SYSMOD entry contains only the LASTSUP information. Such a SYSMOD entry is called a "dummy entry". Because the superseded SYSMOD had not been previously accepted, SMP/E does not know what type of SYSMOD it was (function, PTF, APAR, or USERMOD).

Deleted SYSMODs

When one SYSMOD is deleted by another, SMP/E makes a record of this by adding the name of the deleting SYSMOD to the DELBY subentry of the deleted SYSMOD. If the deleted SYSMOD has not been previously accepted, its distribution zone SYSMOD entry contains only the DELBY information. Such a SYSMOD entry is called a "dummy entry". Although the deleted SYSMOD had not been previously accepted, SMP/E assumes it was a function SYSMOD, because only function SYSMODs can be explicitly deleted.

Conditional requisite data

For each SYSMOD named as an FMID in a ++IF statement, a SYSMOD entry is created with CIFREQ subentries representing the conditional requisite requirements.

If the SYSMOD existed previously, the CIFREQ data is simply added to the existing entry; otherwise, a new SYSMOD entry is created to save the CIFREQ data for use if the FMID is installed later.

Global zone SYSMOD entries

For each SYSMOD that is successfully accepted, SMP/E deletes the SMPPTS modification control statement entry, the global zone SYSMOD entry, and the SMPTLIB data sets for any elements that were packaged in relative files. There are two exceptions:

- If you specified the BYPASS(APPLYCHECK) operand, the entries and SMPTLIB data sets are not deleted. In this case, SMP/E assumes that the SYSMODs have not been applied and, therefore, may be applied later.
- If you set the NOPURGE indicator in the OPTIONS entry you are using, the entries and SMPTLIB data sets are not deleted. In this case, SMP/E does not delete the entries, because you have instructed it not to.

Note:

1. The global zone SYSMOD entry is also deleted when the SYSMOD is rejected.
2. When the SYSMOD entry is deleted during ACCEPT processing, the exception SYSMOD data (that is, HOLD data) associated with the SYSMOD, including internal SYSTEM HOLD data, is not deleted.

If the global zone SYSMOD entry is not deleted, the distribution zone name is added to it as an ACCID subentry. Therefore, the ACCID subentries in the global zone reflect the distribution libraries into which each SYSMOD has been accepted.

Zone and data set sharing considerations

The following identifies the phases of ACCEPT processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

2. SYSMOD selection

Cross-zones
Read with shared enqueue.

Global zone
Read with shared enqueue.

SMPPTS
Read with shared enqueue.

Target zone
Read with shared enqueue.

DLIB zone
Update with exclusive enqueue.

3. Element selection

SMPPTS
Read with shared enqueue.

DLIB zone
Update with exclusive enqueue.

4. Utility calling

DLIB zone
Update with exclusive enqueue.

5. Cross-zone requisite reporting phase

Set-to zone
Update with exclusive enqueue.

Cross-zones
Read with shared enqueue.

Global zone
Read with shared enqueue.

6. Global zone update

Global zone
Update with exclusive enqueue.

SMPPTS
Update with exclusive enqueue.

DLIB zone
Update with exclusive enqueue.

7. Termination

All resources are freed.

Chapter 3. The APPLY command

The APPLY command is used to cause SMP/E to install the elements supplied by a SYSMOD into the operating (or target) system libraries. The APPLY process:

- Selects SYSMODs present in the global zone and applicable to the specified target system
- Makes sure all other required SYSMODs have either been applied or are being applied concurrently
- Selects the elements from those SYSMODs on the basis of the functional and service level of those elements in the target system and the relationship between the SYSMODs being installed, making sure that the installation of another SYSMOD does not cause any current service to regress
- Calls system utilities to install the elements into the target system libraries
- Records the functional and service levels of the new elements in the target zone
- Records the application of the SYSMOD in the target zone
- Performs cross-zone processing
- Updates the SYSMOD entries in the global zone
- Creates, when library change file recording is in effect, records that reflect any successful utility work performed by APPLY processing to update target libraries. For more information about library change file recording, see *z/OS SMP/E Reference*.

The APPLY process is controlled by:

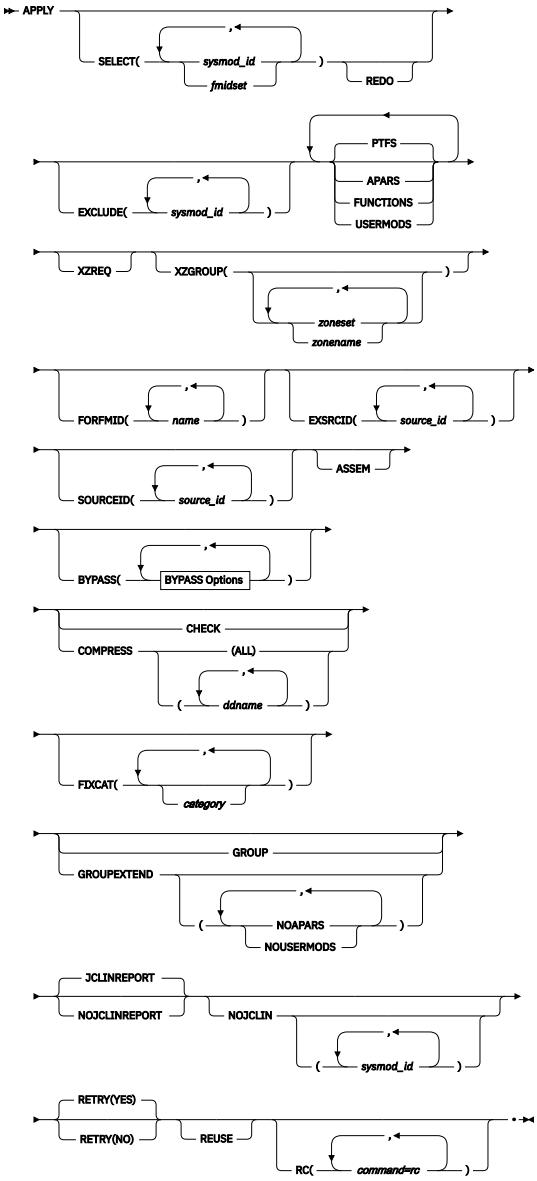
- The information in the target zone reflecting the status and structure of the target system libraries
- Information on the SYSMODs indicating their applicability
- Information in the OPTIONS and UTILITY entries
- Operands on the user's APPLY command

Zones for SET BOUNDARY

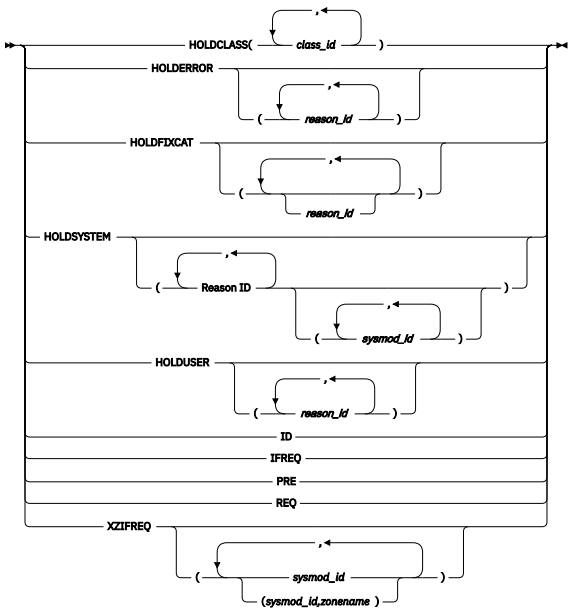
For the APPLY command, the SET BOUNDARY command must specify the target zone associated with the target libraries in which the SYSMODs are installed.

Syntax

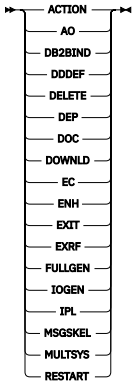
APPLY Command



BYPASS Options



HOLDSYSTEM Reason IDs Used by IBM



Operands

APARS
indicates that all eligible APARs should be applied.

Note:

1. APARS can also be specified as APAR.
2. If APARS is specified along with SELECT, all eligible APARs are included in addition to the SYSMODs specified on SELECT.
3. If APARS is specified along with SOURCEID, all APARs associated with the specified source IDs are included.

ASSEM
indicates that if any SYSMOD contains both source code and object code for the same module, the source code should be assembled and should replace the object code.

BYPASS
You can specify any of these options:

HOLDCLASS
HOLDERROR
HOLDFIXCAT
HOLDSYSTEM

HOLDUSER
ID
IFREQ
PRE
REQ
XZIFREQ
XZIFREQ(*list*)

Note: If you specify both BYPASS and GROUPEXTEND, SMP/E does not include superseding SYSMODs needed to take the place of requisites or error reason IDs that have been bypassed.

During CHECK processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify GROUPEXTEND without BYPASS.

BYPASS(HOLDCLASS(*value*,...))
indicates that exception SYSMODs associated with the specified class names should not be held. The list of class names is required.

These are the hold classes you can specify:

Class
Explanation

ERREL
The SYSMOD is held for an error reason ID but should be installed anyway. IBM has determined that the problem the SYSMOD resolves is significantly more critical than the error reflected by the holding APAR.

HIPER
The SYSMOD is held with a hold class of HIPER (High Impact)

PE
The SYSMOD is held with a hold class of "PTF in Error".

UCLREL
UCLIN needed for the SYSMOD has been handled by IBM and no longer requires your attention.

YR2000
Identifies PTFs that provide Year 2000 function, or fix a Year 2000-related problem.

BYPASS(HOLDERROR)
indicates that exception SYSMODs associated with the specified error reason IDs should not be held. The list of reason IDs is optional.

If you include a list of reason IDs, only the ones you specify are bypassed. If you do not include a list, all error reason IDs are bypassed.

Note: HOLDERROR can also be specified as HOLDERR.

BYPASS(HOLDFIXCAT)
indicates that held SYSMODs associated with the specified fix category reason IDs should not be held. The list of reason IDs is optional. If a list of reason IDs is included, only the ones specified are bypassed. If a list is not included, all fix category reason IDs are bypassed.

BYPASS(HOLDSYSTEM)
indicates that exception SYSMODs associated with the specified system reason IDs should not be held. The list of reason IDs is optional, as is the list of SYSMOD IDs for a particular reason ID. Generally, you should specify BYPASS(HOLDSYSTEM) on all APPLY CHECK commands, and BYPASS(HOLDSYSTEM(*reason_id*,...)) on all APPLY commands for all system reason IDs for which appropriate action has been (or will be) taken.

How you specify the reason IDs determines which system reason IDs are bypassed. Make sure the appropriate action has been taken for all SYSMODs whose reason IDs are to be bypassed.

- If you do not include a list of reason IDs, all system reason IDs are bypassed.

- If you include a list of reason IDs without a list of SYSMOD IDs, all the SYSMODs with the specified reason IDs are bypassed.

  If you include a list of SYSMOD IDs for a particular reason ID, that reason ID is bypassed only for the specified SYSMODs. Other SYSMODs held for that reason remain held, unless the hold is released by some other BYPASS operand (such as CLASS).

Note: HOLDSYSTEM can also be specified as HOLDSYS.

These are the system reason IDs currently used by IBM:

ID
Explanation

ACTION
The SYSMOD needs special handling before or during APPLY processing, ACCEPT processing, or both.

AO
The SYSMOD may require action to change automated operations procedures and associated data sets and user exits in products or in customer applications. The PTF cover letter describes any changes (such as to operator message text, operator command syntax, or expected actions for operator messages and commands) that can affect automation routines.

DB2BIND
A DB2 application REBIND is required for the SYSMOD to become effective.

DDDEF
Data set changes or additions as required.

DELETE
The SYSMOD contains a ++DELETE MCS, which deletes a load module from the system.

DEP
The SYSMOD has a software dependency.

DOC
The SYSMOD has a documentation change that should be read before the SYSMOD is installed.

DOWNLD
Code that is shipped with maintenance that needs to be downloaded.

DYNACT
The changes supplied by the SYSMOD may be activated dynamically without requiring an IPL. The HOLD statement describes the instructions required for dynamic activation. If those instructions are not followed, then an IPL is required for the SYSMOD to take effect.

EC
The SYSMOD needs a related engineering change.

ENH
The SYSMOD contains an enhancement, new option or function. The HOLD statement provides information to the user regarding the implementation and use of the enhancement.

EXIT
The SYSMOD contains changes that may affect a user exit. For example, the interface for an exit may be changed, an exit may need to be reassembled, or a sample exit may be changed.

EXRF
The SYSMOD must be installed in both the active and the alternative Extended Recovery Facility (XRF) systems at the same time to maintain system compatibility. (If you are not running XRF, you should bypass this reason ID.)

FULLGEN
The SYSMOD needs a complete system or subsystem generation to take effect.

IOGEN
The SYSMOD needs a system or subsystem I/O generation to take effect.

IPL
The SYSMOD requires an IPL to become effective. For example, the SYSMOD may contain changes to LPA or NUCLEUS, the changes may require a CLPA, or a failure to perform an IPL might lead to catastrophic results, such as could be caused by activation of a partial fix.

Note: If you plan to perform an IPL with CLPA after the SYSMOD has been applied, then no further investigation of the HOLD is required; simply bypass the IPL reason ID. However, if you are not planning to perform an IPL with CLPA, then the details of the HOLD statement must be investigated to determine what kind of actions are required to activate the SYSMOD.

MSGSKEL
This SYSMOD contains message changes that must be compiled for translated versions of the message changes to become operational on extended TSO consoles.

If you want to use translated versions of the messages, you must run the message compiler once for the library containing the English message outlines, and once for each additional language you want to be available on your system. For details, see *z/OS MVS Planning: Operations*.

If you want to use **only** the English version of the messages, you do not need to run the message compiler. You should bypass this reason ID.

MULTSYS
Identifies fixes that need to be applied to multiple systems, in one of three cases: preconditioning, coexistence, or exploitation.

RESTART
To become effective, the SYSMOD requires a special subsystem restart operation. The HOLD statement contains information regarding the required restart actions.

BYPASS(HOLDUSER)
indicates that exception SYSMODs associated with the specified user reason IDs should not be held. The list of reason IDs is optional.

If you include a list of reason IDs, only the ones you specify are bypassed. If you do not include a list, all user reason IDs are bypassed.

BYPASS(ID)
indicates that SMP/E should ignore any errors it detects when checking the SYSMOD's RMID and UMIDs. BYPASS(ID) should be used with caution and only after careful consideration because ignoring RMID and UMID errors may regress modifications included in previously applied SYSMODs.

BYPASS(IFREQ)
indicates that SMP/E should ignore any conditional requisites that are missing.

BYPASS(PRE)
indicates that SMP/E should ignore any missing prerequisites.

BYPASS(REQ)
indicates that SMP/E should ignore any requisites that are missing.

BYPASS(XZIFREQ)
indicates that SMP/E is to continue APPLY processing for a SYSMOD, even if SMP/E detects a missing cross-zone requisite. SMP/E will identify such missing cross-zone requisites with a warning message, instead of terminating the APPLY processing.

BYPASS(XZIFREQ(*list*))
indicates that SMP/E is to continue APPLY processing for a SYSMOD, even if SMP/E detects a missing cross-zone requisite, provided that the missing requisite SYSMOD is included in the list provided with the XZIFREQ option. For SYSMODs identified in the list, SMP/E will identify with a warning message any that are missing cross-zone requisites. For missing requisite SYSMODs that are **not** included in the list, SMP/E will terminate APPLY processing.

Each entry in the list must be in one of the following formats:

- *sysmod_id*
- (*sysmod_id*,*zone*)

sysmod_id
specifies that SMP/E is to continue APPLY processing, even if requisite SYSMOD *sysmod_id* in any zone (other than the set-to zone) is missing.

(sysmod_id,zone)
specifies that SMP/E is to continue APPLY processing, even if requisite SYSMOD *sysmod_id* in zone *zone* is missing.

Each entry in the list must be unique. Also, a SYSMOD ID must not appear both by itself and as part of a SYSMOD/zone pair. However, a SYSMOD ID may appear in multiple SYSMOD/zone pairs, provided each of the pairs is unique.

The list provided must not be a null list; that is, BYPASS(XZIFREQ()) is not allowed.

CHECK
indicates that SMP/E should not actually update any libraries. Instead, it should just take these actions:

- Test for errors other than those that occur when the libraries are actually updated.
- Report on which libraries are affected.
- Report on any SYSMOD that would be regressed.

COMPRESS
indicates which target libraries should be compressed. SMP/E does **not** compress any libraries that are actually paths in a UNIX file system.

- If you specify ALL, any libraries in which elements will be installed by this APPLY command are compressed.
- If you specify particular ddnames, those libraries are compressed regardless of whether they will be updated.

Note:

1. COMPRESS can also be specified as C.
2. If you specify both COMPRESS and CHECK, COMPRESS is ignored. This is because SMP/E does not update any data sets for CHECK.

EXCLUDE
specifies one or more SYSMODs that should not be applied.

Note:

1. EXCLUDE can also be specified as E.
2. SMP/E does not include a SYSMOD that would be included by the GROUP or GROUPEXTEND operand if that SYSMOD is specified on the EXCLUDE operand.
3. SYSMODs that are superseded by another SYSMOD being applied are not excluded even if they are specified on the EXCLUDE operand.

EXSRCID
indicates that SYSMODs associated with the specified source IDs should **not** be applied.

Note:

1. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are excluded.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.

- A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
  - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU* are excluded.
  - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are excluded.
  - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are excluded.
- A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for example, SYSMODs that contain any of these source IDs are excluded: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not excluded.

Any number of asterisks and percent signs can be used within a single partially specified source IDs.

The following examples are valid source ID specifications:

```
RSU0709
RSU*
IBM.Device.20%4
IBM.Device.*.zAAP
```

2. A given source ID can be explicitly specified **only once** on the EXSRCID operand.
3. The same source ID **cannot** be explicitly specified on both the EXSRCID and source ID operands.
4. If a source ID is implicitly or explicitly specified on the EXSRCID operand and on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs, at least one of which is specified either implicitly or explicitly on the SOURCEID operand and another is specified on the EXSRCID operand, the SYSMOD is excluded from processing.

   For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.
6. If a SYSMOD would be included by the GROUP or GROUPEXTEND operand, but is excluded by the EXSRCID operand, SMP/E does not include it.
7. If you do not specify any SYSMOD types, SMP/E processes only PTFs. To process other types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

FIXCAT

identifies the list of fix categories of interest for command processing. This list determines which fix category APARs must be resolved for the SYSMODs being applied.

A fix category APAR provides a fix for a held SYSMOD and the APAR is associated with one or more fix categories. Fix category APARs are identified by FIXCAT HOLD entries. If a fix category specified on a FIXCAT HOLD for a SYSMOD being applied matches any of those specified on the FIXCAT operand of the command, then the SYSMOD is held for the APAR reason ID from the FIXCAT HOLD and will not be applied until the APAR is resolved. If a fix category specified on a FIXCAT HOLD for a SYSMOD being applied does not match any of those specified on the FIXCAT operand of the command, or if the list of fix categories is null, the SYSMOD is not held for the APAR reason ID from the FIXCAT HOLD.

The values specified on the FIXCAT operand will override the list of values, if any, defined by the FIXCAT subentry in the active OPTIONS entry. FIXCAT() may be used to specify a null list, which means no fix category APARs must be resolved during current accept processing.

Fix category values can be 1 to 64 characters in length and can contain any nonblank character in the range X'41' - X'FE' except the single quotation mark, comma, left parenthesis, and right parenthesis. They can be specified in two ways:

- Explicitly, by fully specifying a particular fix category value. For example, IBM.Device.zIIP. In this case, all HOLDDATA associated with this fix category is applicable to command processing.
- Implicitly, by partially specifying a fix category value using any number of asterisks (*) as global characters and percent signs (%) as placeholders.
  - A single asterisk indicates that zero or more characters can occupy that position. For example, IBM.Device*, *z/OS or IBM*z/OS. In the first case, all HOLDDATA associated with a fix category that begins with the character string IBM.Device is applicable. In the second case, all HOLDDATA associated with a fix category that ends with the character string z/OS is applicable. In the third case, all HOLDDATA associated with a fix category that begins with the character string IBM and ends with the character string z/OS is applicable.
  - A single percent sign indicates that any one single character can occupy that position. For example, IBM.Device.20%4. In this case, HOLDDATA associated with any of the following fix categories is applicable: IBM.Device.2084, IBM.Device.2094, and IBM.Device.20t4. HOLDDATA associated with fix category IBM.Device.20914, however, is not applicable.

Fix category values can contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified Fix category value. For example, a specified value of IBM.FUNCTION.HEALTHCHECKER matches the value of IBM.Function.HealthChecker.

Fix category values are defined by FIXCAT HOLD entries. The following examples of acceptable fix categories are based on the fix category values that are used by IBM in FIXCAT HOLD entries:

```
IBM.Device.2094.zAAP
*
IBM.Function*
IBM.Device.20%4.*
*.HealthChecker
```

FORFMID
indicates that only SYSMODs for the specified FMIDs or FMIDSETs should be applied.

Note:

1. Functions containing a ++VER DELETE statement are not automatically included by the FORFMID operand. You must specify them on the SELECT operand.
2. If you do not specify any SYSMOD types, SMP/E processes only PTFs. To process other types of SYSMODs, you must specify the desired SYSMOD types.

FUNCTIONS
indicates that all eligible functions should be applied.

Note:

1. FUNCTIONS can also be specified as FUNCTION.
2. If you specify FUNCTIONS along with SELECT, all eligible functions are included in addition to the SYSMODs specified on SELECT.
3. If you specify FUNCTIONS along with SOURCEID, all functions associated with the specified source IDs are included.
4. Functions containing a ++VER DELETE statement are not automatically included by the FUNCTIONS operand. You must specify them on the SELECT operand.

GROUP
indicates that if any SYSMODs specifically defined as requisites for eligible SYSMODs have not yet been applied, SMP/E should automatically include them.

Note:

1. GROUP can also be specified as G.
2. GROUP is mutually exclusive with GROUPEXTEND.
3. GROUP may include SYSMODs at a service level higher than that specified by the SOURCEID operand.

4. If you specify GROUP with no other SYSMOD selection operands (such as a SYSMOD type, SOURCEID, FORFMID, or SELECT), GROUP is ignored.
5. Processing done for SYSMODs specified on the SELECT operand is not necessarily done for SYSMODs included by the GROUP operand. For example, if RED0 is specified, only SYSMODs specified on the SELECT operand can be reapplied; SYSMODs included by the GROUP operand are not.
6. Functions containing a ++VER DELETE statement are not automatically included by the GROUP operand. You must specify them on the SELECT operand.
7. If a SYSMOD would be included by the GROUP operand, but is excluded by the EXCLUDE or EXSRCID operand, SMP/E does not include it.

GROUPEXTEND
indicates that if a SYSMOD specifically defined as a requisite for an eligible SYSMOD has not been applied and cannot be processed for one of the reasons shown in Table 4 on page 60, SMP/E should automatically include a superseding SYSMOD. As the table shows, what GROUPEXTEND includes depends on why the requisite cannot be processed.

Table 4. What GROUPEXTEND includes (APPLY processing)

For a requisite that is:	GROUPEXTEND includes:
• Held for an error reason ID	• A SYSMOD that supersedes the requisite OR • A SYSMOD that matches or supersedes the error reason ID
One of these situations: • Held for a system reason ID • Held for a user reason ID • Applied in error • Not available	• A SYSMOD that supersedes the requisite

You can specify NOAPARS or NOUSERMODS (or both NOAPARS and NOUSERMODS) to limit the types of SYSMODS that are included by GROUPEXTEND to resolve error reason IDs. The default is to include all eligible SYSMODs, regardless of SYSMOD type.

NOAPARS
indicates that SMP/E should exclude APARs that resolve error reason IDs.

NOUSERMODS
indicates that SMP/E should exclude USERMODs that resolve error reason IDs.

Note:

1. GROUPEXTEND can also be specified as GEXT.
2. GROUPEXTEND is mutually exclusive with GROUP.
3. If you specify both BYPASS and GROUPEXTEND, SMP/E does not include superseding SYSMODs needed to take the place of requisites or error reason IDs that have been bypassed.

   During CHECK processing, if you want to see whether any superseding SYSMODs are available for requisites that have been bypassed, specify GROUPEXTEND without BYPASS.
4. GROUPEXTEND may include SYSMODs at a service level higher than what is specified by the SELECT or SOURCEID operands.
5. Functions and excluded SYSMODs are not automatically included by GROUPEXTEND.
6. Processing done for SYSMODs specified on the SELECT operand is not necessarily done for SYSMODs included by the GROUPEXTEND operand. For example, if RED0 is specified, only SYSMODs specified on the SELECT operand can be reapplied; SYSMODs included by the GROUPEXTEND operand cannot.

7. If a SYSMOD would be included by the GROUPEXTEND operand, but is excluded by the EXCLUDE or EXSRCID operand, SMP/E does not include it.
8. When GROUPEXTEND is specified, SMP/E examines more SYSMODs than it does if GROUP were specified. Because of this additional processing, the APPLY command runs longer than if GROUP was specified, and a larger region size might be needed.

   On the other hand, GROUPEXTEND reduces the amount of time you would otherwise spend searching for missing requisites.

JCLINREPORT
indicates that SMP/E is to write the JCLIN reports after processing inline JCLIN. This is the default.

Note: JCLINREPORT can also be specified as JCLR.

NOJCLIN
indicates that SMP/E should not process inline JCLIN for the specified SYSMODs. For example, if you are reapplying SYSMODs, you may not want to process inline JCLIN that would change target zone entries that should not be changed.

If you include a list of SYSMOD IDs, SMP/E skips JCLIN processing only for the specified SYSMODs. If you do not include a list of SYSMOD IDs, SMP/E skips JCLIN processing for all SYSMODs.

NOJCLINREPORT
indicates that SMP/E should not write any JCLIN reports after processing inline JCLIN.

Note: NOJCLINREPORT can also be specified as NOJCLR.

PTFS
indicates that all eligible PTFs should be applied.

Note:

1. PTFS can also be specified as PTF.
2. PTFS is the default SYSMOD type for mass-mode processing. If you do not specify any other SYSMOD types, only PTFs are processed, even if PTFS was not specified.
3. If you specify PTFS along with SELECT, all eligible PTFs are included in addition to the SYSMODs specified on SELECT.
4. If you specify PTFS along with SOURCEID, all PTFs associated with the specified source IDs are included.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the APPLY command.

Before SMP/E processes the APPLY command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the APPLY command. Otherwise, the APPLY command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the APPLY command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

REDO
indicates that if any SYSMOD specified on SELECT has already been successfully applied, it should be reapplied.

Note:

1. If you specify REDO, you must also specify SELECT.

2. If GROUP or GROUPEXTEND is also specified, REDO does not reapply SYSMODs included by the GROUP or GROUPEXTEND operand. It processes only SYSMODs specified on the SELECT operand.
3. If you use REDO to reapply a SYSMOD with inline JCLIN, you may not be able to restore that SYSMOD. This is the case if the target zone entries were updated the first time the SYSMOD was applied. When the SYSMOD is reapplied by use of the REDO operand, the target zone entries are first copied to the SMPSCDS as BACKUP entries, and then updated again for the SYSMOD. As a result, the BACKUP entries and the target zone entries are at the same level, and SMP/E has no record of the target zone entries before the SYSMOD was installed.
4. When reapplying a function SYSMOD, be sure to also reapply all PTFs, APARs, and USERMODs for the same FMID that have already been applied to prevent intersecting elements from being regressed. Otherwise, the correct service level of the intersecting elements may not be installed.

RETRY
indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

YES
indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If retry processing does not reclaim sufficient space and input to the utility was batched (copy or link-edit utility only), SMP/E debatches the input and retries the utility for each member separately. If this final attempt fails, the resulting *x*37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see *z/OS SMP/E User's Guide*. For more information about OPTIONS entries, see *z/OS SMP/E Reference*.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if you specify YES.

NO
indicates that SMP/E should not try to recover from the error.

REUSE
indicates that if a module was successfully assembled during previous SMP/E processing, it should not be reassembled. Instead, the existing object module from SMPWRK3 should be reused.

Note: The REUSE operand must be used with great care. SMP/E does not ensure that the same set of SYSMODs are being processed after a failure. If new maintenance is received after the initial APPLY command and before the APPLY REUSE command, SMP/E may use the wrong level of object modules.

SELECT
specifies one or more SYSMODs that should be applied.

You may specify any combination of individual SYSMOD IDs and FMIDSET names, provided that there are no duplicate SYSMOD IDs nor any duplicate FMIDSET names. For each FMIDSET specified, all FMIDs defined in the FMIDSET are processed as if they were explicitly specified in the SELECT list.

Note:

1. SELECT can also be specified as S.
2. To reapply a SYSMOD, it is not enough to specify that SYSMOD on the SELECT operand. You must also specify RED0.
3. To process functions containing a ++VER DELETE statement, you must specify them on the SELECT operand.
4. When using FMIDSETs on the SELECT operand, remember that:
   - A value specified in the SELECT list is processed as an FMIDSET if the GLOBAL zone contains an FMIDSET entry by that name.
   - A value specified in the SELECT list is processed as a SYSMOD ID if it is not defined as an FMIDSET in the GLOBAL zone and it is a valid SYSMOD ID.

- If the value in the SELECT list is valid both as a SYSMOD ID and as an FMIDSET name, it is processed (for SELECT) as an FMIDSET. If you want to select a SYSMOD that has the same name as an FMIDSET, you must define that SYSMOD in an FMIDSET and then include that FMIDSET name in the SELECT list.

  If this same value is specified on the EXCLUDE operand, it will be processed as a SYSMOD ID (because only SYSMOD IDs are valid on EXCLUDE) and will **not** be rejected as a duplication of the identical FMIDSET name in the SELECT list.
- Any given value (whether it represents a SYSMOD ID, an FMIDSET, or both) may **not** appear more than once in the SELECT list.
- Any given SYSMOD ID may not simultaneously appear in both the SELECT and EXCLUDE lists, unless it is also a valid FMIDSET name.
- A SYSMOD ID may be explicitly specified in the SELECT list and also included in an FMIDSET that is also specified in the SELECT list, provided the SYSMOD ID does not have the same name as the FMIDSET. The duplicate SYSMOD ID is ignored.

SOURCEID
indicates that SYSMODs associated with the specified source IDs should be applied.

Note:

1. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are selected.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU are selected.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are selected.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are selected.
     - A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for instance, SYSMODs that contain any of these source IDs are selected: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not selected.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source IDs:

   ```
   RSU0709
   RSU*
   IBM.Device.20%4
   IBM.Device.*.zAAP
   ```
2. A given source ID can be explicitly specified **only once** on the SOURCEID operand.
3. The same source ID **cannot** be explicitly specified on both the EXSRCID and the SOURCEID operands.
4. If a source ID is implicitly or explicitly specified both on the SOURCEID operand and on the EXSRCID operand, all SYSMODs with that source ID are excluded from processing.
5. If a given SYSMOD has multiple source IDs, at least one of which is specified either implicitly or explicitly on the SOURCEID operand and another on the EXSRCID operand, the SYSMOD will be excluded from processing.

For example, assume that PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.

6. Functions containing a ++VER DELETE statement are not automatically included by the SOURCEID operand. You must specify them on the SELECT operand.
7. If you do not specify any SYSMOD types, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.
9. SOURCEIDs for fix category values are assigned to the resolving (fixing) PTFs named on the FIXCAT ++HOLDs, during the RECEIVE of those ++HOLDs. The SOURCEIDs can only be assigned to a PTF, if that PTF has been received before the HOLDDATA is received. This requirement is met if the PTF was received during a previous RECEIVE command, or during the same RECEIVE with SYSMODS HOLDDATA, since the SYSMODs are received first then the HOLDDATA.

USERMODS
indicates that all eligible USERMODs should be applied.

Note:

1. USERMODS can also be specified as USERMOD.
2. If USERMODS is specified along with SELECT, all eligible USERMODs are included in addition to the SYSMODs specified on SELECT.
3. If USERMODS is specified along with SOURCEID, all USERMODs associated with the specified source IDs are included.

XZGROUP(*list*)
indicates that you wish to override SMP/E's default method for determining the zones to be checked for cross-zone requisites.

You may specify either:

- A list of ZONESETs and zones that are to be used to establish the zone group for this command. Each value in the list must be a valid ZONESET or zone name.
- XZGROUP() to provide a null list, which means that no cross-zone requisite checking is to be done for this command. A null list is not valid if the XZREQ operand is also specified.

The XZGROUP operand always requires a list or null list. That is, XZGROUP (without parentheses) is not allowed.

Note:

1. If XZGROUP is specified, whatever ZONESETs the user specifies are used to establish the initial zone group, even if the set-to zone is not in a ZONESET and the XZREQCHK subentry is not set.
2. If no XZGROUP operand is specified on the APPLY command, SMP/E reads all ZONESET entries. If a ZONESET entry has its XZREQCHK subentry set to YES and it contains the set-to zone, then all the other zones within the ZONESET entry become part of the initial zone group for the APPLY command.
3. After the initial zone group is established, it is culled by removing all distribution zone for APPLY processing. In other words, only zones having the same type as the set-to zone are left in the final zone group used for cross-zone requisite checking.

XZREQ
indicates that SMP/E should install unsatisfied cross-zone requisites into the set-to zone.

XZREQ causes cross-zone requisites to become primary candidates for installation. To do this, SMP/E checks secondary zones in the currently established zone group for CIFREQ data that is applicable to functions installed or being installed into the set-to zone.

Note:

1. SYSMODs selected with the XZREQ operand are in addition to any SYSMODs selected with the FORFMID and SOURCEID operands.
2. If XZREQ is specified along with SELECT, the specifically selected SYSMODs are included along with any unsatisfied cross-zone requisites.
3. If SOURCEID is specified, only cross-zone requisites with the specified SOURCEID values become primary candidates for installation.
4. If FORFMID is specified, only cross-zone requisites for the specified FMIDs become primary candidates for installation. Otherwise, all unsatisfied cross-zone requisites become primary candidates for installation.
5. When the XZREQ operand is specified without the FORFMID operand, the SOURCEID operand, or the SELECT operand, only unsatisfied cross-zone requisites become primary candidates.
6. PTFS is the default SYSMOD type for mass-mode processing. If no SYSMOD types are specified, only PTFs are processed, even if PTFS was not specified.
7. If any SYSMOD types are specified, processing is limited to those SYSMOD types, except for those SYSMODs that might be needed to satisfy processing for these operands:
   - GROUP
   - GROUPEXTEND
   - SELECT
   - XZREQ
8. If EXSRCID is specified, any unsatisfied cross-zone requisite with one of the specified source IDs is excluded from processing.
9. If the XZREQ operand is specified, the XZGROUP operand may not be specified as a null list.

Syntax notes

Figure 3 on page 66 shows how SMP/E chooses which SYSMODs to process, on the basis of the operands specified on the APPLY command.

- If you specify any of the operands in the top part of the chart, or if you do not specify the SELECT operand, SMP/E does *mass-mode* processing.
- If you specify the SELECT operand, SMP/E does *select-mode* processing.
- If you specify SELECT plus operands from the top part of the chart, SMP/E does both *select-mode* and *mass-mode* processing.

For more information about select-mode and mass-mode processing, see "Candidate selection" on page 77.

Remember the following when coding the APPLY command:

1. SMP/E applies SYSMODs specified on SELECT regardless of other APPLY operands (such as a SYSMOD type, SOURCEID, EXSRCID, or FORFMID). Therefore, if you want to apply a specific SYSMOD, you only need to specify the SYSMOD ID on SELECT. For example, to apply a specific APAR, you do not also have to include the APAR operand.

   Note: If you do specify a SYSMOD type along with SELECT, SMP/E applies all SYSMODs of the specified type plus the selected SYSMOD.
2. If you specify more than one SYSMOD type, a SYSMOD needs to match only one of the specified types.
3. If the SOURCEID, FORFMID, and SYSMOD type operands are specified together, only those SYSMODs meeting all the conditions are applied. (For a SYSMOD type, the SYSMOD must meet just one of the type conditions.)

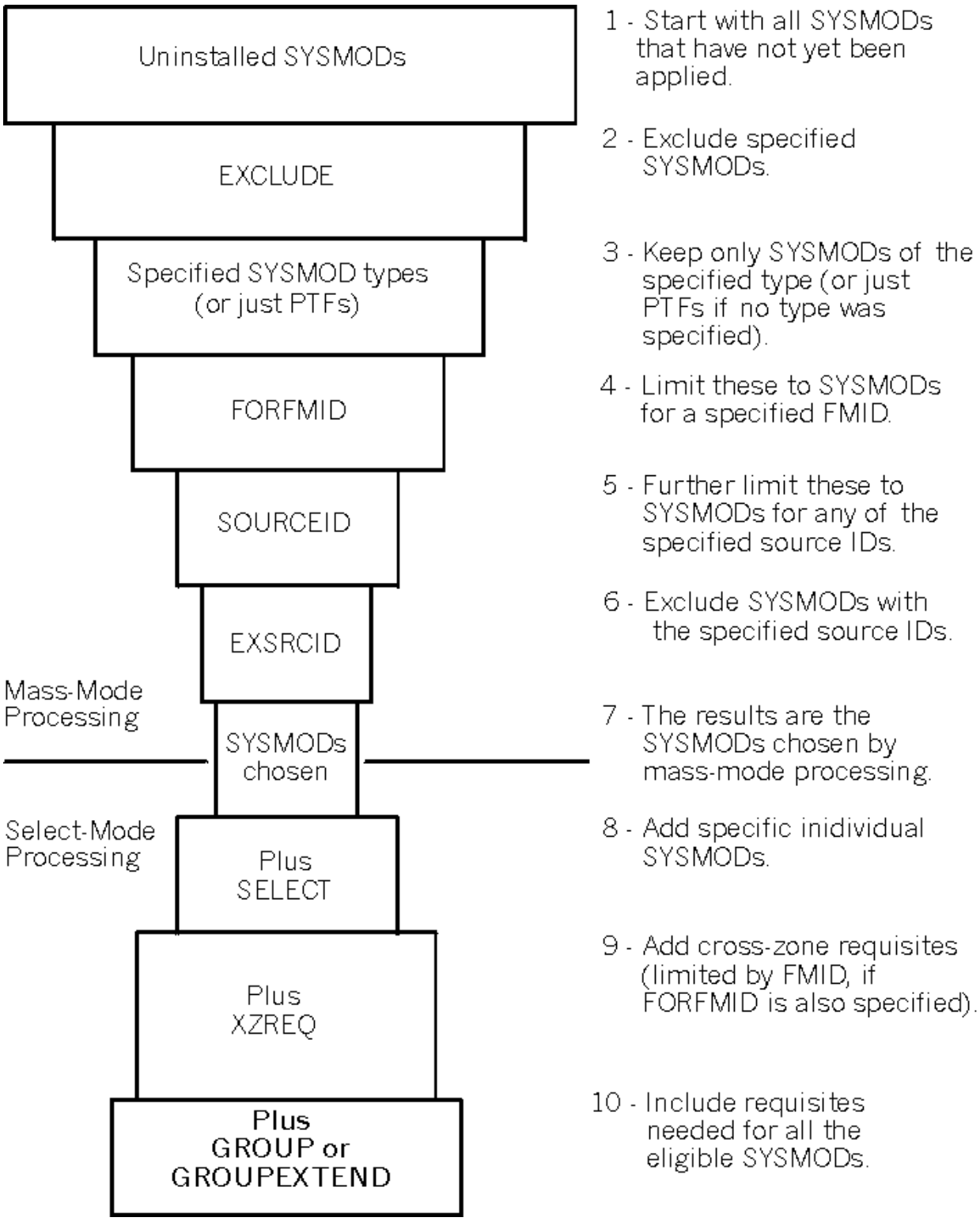


Figure 3. Combining SYSMOD selection operands on the APPLY Command

Data sets used

These data sets might be needed to run the APPLY command. They can be defined by DD statements or, ordinarily, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

Distribution library	SMPLOGA	SMPSTS	SYSPRINT
Link library	SMPLTS	SMPTLIB	SYSUT1
SMPCNTL	SMPMTS	SMPWKDIR	SYSUT2
SMPCSI	SMPOUT	SMPWRK1	SYSUT3
SMPDATA1	SMPPARM	SMPWRK2	SYSUT4
SMPDATA2	SMPPTS	SMPWRK3	Target library
SMPHRPT	SMPRPT	SMPWRK4	Text library
SMPJHOME	SMPSCDS	SMPWRK6	*zone*
SMPLOG	SMPSNAP	SYSLIB	

Note:

1. SMPHRPT is an optional data set. If SMPHRPT is defined, the HOLD reports are directed there.
2. SMPJHOME is an optional DD statement used to specify the Java runtime directory. SMP/E requires the Java runtime when either of the following conditions applies.
   - When SMP/E is installing JARUPD elements, or
   - When a UNIX shell script is invoked during the installation of a file system element, and the shell script issues a Java command.
3. SMPWKDIR is an optional DD statement used to specify a directory in a UNIX file system for the storage of temporary files created during SMP/E processing. If the SMPWKDIR directory is not specified on a DD statement or DDDEF entry, SMP/E will use the /tmp directory for temporary work files.
4. The SMPLTS data set is required only when a load module with CALLLIBS is being processed.
5.
6. The SMPDATA1 and SMPDATA2 data sets are required only when the CHANGEFILE subentry of the active OPTIONS entry is set to "YES" for the target zone that APPLY is operating on.
7. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
8. SMPPARM is required only if exit routines have been defined in SMPPARM member GIMEXITS.

Usage notes

This section provides usage notes for the APPLY command.

Adding new elements other than modules to the target libraries

The target library for macros, source, data elements, hierarchical file system elements, and program elements is found as a SYSLIB subentry for an element entry in the target zone. If no SYSLIB subentry is present and no SYSLIB is specified on the element's MCS, a check is made to determine whether the element's distribution library has been totally copied to a target library. If it has (as shown by the presence of a target zone DLIB entry for the distribution library), the target library to which the DLIB was copied is determined to be the target library for the element.

Note: There should be only one SYSLIB subentry for the DLIB entry, and the SYSLIB subentry must specify the ddname of the target library for the elements.

Adding new modules to the target libraries

The SMP/E APPLY process can be used to install a SYSMOD that introduces a new module into the system libraries. To apply a module for the first time, SMP/E requires that the DISTLIB operand be specified on the ++MOD statement.

If the module is to be installed into an existing load module, this fact can be identified to SMP/E in two ways:

- If no additional link-edit control statements (other than the INCLUDE statement) are required to rebuild the load module, the LMOD operand on the ++MOD statement can be used to indicate that the new module is to be link-edited into the existing load module. If no LMOD entry exists in the target zone for the specified load module, an error results.
- If adding the new module requires the addition of a link-edit control card in order to rebuild the load module (for example, another ORDER card is required or a new ENTRY is established), inline JCLIN is required to redefine the load module structure.

If the module is part of a copied library, SMP/E already has all the information necessary to install the module fully; therefore, no special operands or processing is required. SMP/E uses the DLIB entry to determine that the distribution library has been totally copied, to build a new target zone MOD entry with a LMOD subentry equal to the module name, and to build a LMOD entry (with a name equal to the module name), using the SYSLIB value from the DLIB entry and the LEPARMs from the ++MOD LEPARMs.

Checking the DISTLIB operand

When an element is selected for application and a target zone entry for that element already exists, the value of the DISTLIB operand on the element is compared with the DISTLIB subentry in the target zone element entry. If they are not equal, SMP/E 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 no element entry exists in the target zone, the service SYSMODs must specify the same DISTLIB as the function SYSMODs on the elements. If they do not, SMP/E issues an error message and the APPLY command is terminated.

If two service SYSMODs update or replace the same element, have different DISTLIB operand values, and are both eligible for processing, but no entry for the element exists in the target zone, the service SYSMODs must specify the same DISTLIBs on the element . If they do not, SMP/E issues an error message and the service SYSMODs are terminated.

DISTSRC, ASSEM, and DISTMOD operands

Because SMP/E cannot determine from the data processed by JCLIN what sources are contained in a totally copied library, the DISTSRC, ASSEM, and DISTMOD operands are provided to pass this information to SMP/E when a macro is replaced or updated, and the macro change must cause the source to be reassembled.

- The DISTSRC operand value specifies the name of the distribution library containing the source.
- The ASSEM and PREFIX operand values specify a list of ASSEM entries in the source or target zone that should be assembled during APPLY processing.
- The DISTMOD operand value specifies the name of the distribution library containing the load modules.

These four operands are specified on ++MAC and ++MACUPD statements. The DISTMOD operand is also specified on ++SRC and ++SRCUPD statements.

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

If neither a SRC nor an ASSEM entry exists for a module in the ASSEM operand values, a SRC entry is created. The DISTSRC operand value is placed in the SRC entry as the DISTLIB subentry. If there is a DLIB entry in the target zone for the DISTSRC operand value, the SYSLIB subentries from the DLIB entry are placed in the SRC entry as SYSLIB subentries. If no DLIB entry exists, the SYSLIB subentry in the SRC entry is left null, and the SMPSTS is used in place of a target library.

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

If no LMOD entry exists for a module, one is created, provided the target zone contains a DLIB entry for the DISTMOD operand value. The SYSLIB subentries from the DLIB entry are placed in the LMOD entry as SYSLIB subentries, and the LMOD subentry is placed in the MOD entry. If no DLIB entry exists, 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 has been updated or replaced, all the modules specified in the ASSEM and PREFIX operand lists are assembled. If no member is found in the necessary library (the source target system library, the SMPSTS, or the distribution library) for a source specified in the ASSEM operand list, SMP/E issues a warning message and goes on processing the SYSMOD without assembling or link-editing the module. If an assembly completes with a return code greater than the one that you specified in the RC subentry of the ASM UTILITY entry (or the SMP/E default of 4, if the RC 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, the link-edit is performed.

Use of the SMPMTS and SMPSTS as target libraries

If no operating system library can be determined, macros are stored in the SMPMTS library and the source in the SMPSTS library.

Macro and source elements stored in the SMPMTS or SMPSTS remain there until they are replaced by elements from subsequent APPLY processing or until the SYSMODs that modified them are applied. In this way, the SMPMTS serves as a macro library for assemblies during APPLY processing and must be in the concatenation of the SYSLIB DD statement for APPLY. Likewise, the SMPSTS serves as a source library for assemblies during APPLY processing, but is **not** required in the SYSLIB concatenation.

Use of the SMPLTS library

The SMPLTS data set is a target library used to maintain the "base" version of a load module specifying a SYSLIB allocation in order to implicitly include modules. How the SMPLTS data set is used depends on whether the set-to zone has an UPGLEVEL subentry:

- If the set-to zone has an UPGLEVEL subentry, the "base" version of the load module or program object is built in the SMPLTS data set only if it contains both CALLLIBS and XZMOD subentries. In some cases, however, SMP/E may need to create a temporary member in the SMPLTS to resolve certain warning conditions identified by the binder. This temporary member (if created) is deleted from the SMPLTS after a successful link-edit into the target library.
- If the set-to zone does not have an UPGLEVEL subentry, the "base" version of the load module or program object that has CALLLIBS subentries is **always** built in the SMPLTS data set.

If it is necessary to build a "base" version of a load module, SMP/E builds it in two stages:

1. The "base" version of a load module stored in the SMPLTS data set contains all the modules that are defined for it from the LMOD operand on ++MOD statements or through JCLIN.
2. The executable version of the load module is built in the target libraries using the load module's SYSLIB allocation (the CALLLIBS subentry list in its LMOD entry) and the "base" version of the load module from the SMPLTS data set. This version contains modules that are implicitly defined through the SYSLIB allocation as being included in the load module.

A load module stored in the SMPLTS remains there until it is replaced by a new version from subsequent APPLY processing or (if the set-to zone has an UPGLEVEL subentry) it is deleted (by the CLEANUP command, for example) because it is no longer needed.

For more information, see “Building load modules with a SYSLIB allocation” on page 96.

Alias processing

When an element with aliases is processed, both the element and its aliases are updated. SMP/E does not check the aliases against elements maintained in the target zone. The user must make sure an element's alias does not match the name of an element maintained by SMP/E in the target zone.

Aliases for an element are determined as follows:

- Replacement elements (MACs, MODs, data elements, and program elements):
  - If a list of aliases is specified on the SMP/E MCS, these aliases are used. The new list of aliases replaces any alias subentries in the target zone element entry.
  - If no list of aliases is specified on the SMP/E MCS, the aliases found as alias subentries in the target zone element entry are used.
- Update elements (ZAPs and MACUPDs):
  - If a list of aliases is specified on the SMP/E MCS, these aliases are used. Any alias subentries in the target zone element entry are ignored for update processing of the element. Macro aliases (on the target system) existing before this list of aliases was presented to SMP/E are not updated. Alias subentries in the target zone element entry are not updated or replaced by the aliases in this list.
  - If no list of aliases is specified on the SMP/E MCS, the aliases found as alias subentries in the target zone element entry are used.

APPLY CHECK facility

The purpose of the CHECK option is to perform a dry run to inform you of possible errors, and to provide reports of SYSMOD status, libraries that will be updated, regression conditions, and SYSMODs that will be deleted. Target system libraries are not permanently updated.

During check processing, the list of target zone entries is maintained in storage; data is written to the target zone as a temporary storage medium. Check processing deletes any data written to the target zone. Consequently, no permanent updates are made to the target zone.

SYSMOD termination

Termination of a SYSMOD causes a return code of 8. Termination of a ++FUNCTION causes a return code of 12. Termination occurs in response to any of the following conditions:

- Missing requisites:
  - The requisite SYSMOD is not available on the PTS/CSI data sets. (It has not been received.)
  - The requisite SYSMOD has been excluded.
  - The requisite SYSMOD was terminated (possibly because other requisites are missing).
  - The requisite SYSMOD did not meet the applicability criteria.
  - The requisite SYSMOD was not included in the SELECT list, and neither GROUP nor GROUPEXTEND was specified.
  - GROUP was specified to include the requisite, but the requisite SYSMOD is being held or is not available on the PTS or CSI data sets. (It has not been received.)
  - GROUPEXTEND was specified to supersede the failing requisite, but a superseding SYSMOD was not available for processing.
- Inline JCLIN processing failure. The entries affected are restored to the state that existed before JCLIN processing.
- MODID error conditions.
- Attempting to change the ownership of an element that is being updated rather than replaced.
- DISTLIB operand checking failure.
- DD statement missing for a target system library.

- Utility return codes. Return codes from the utilities called to update, assemble, copy, and link-edit elements to the target system are examined to determine whether the operation has succeeded or failed. If these return codes exceed a predefined value, the SYSMODs whose elements are involved in the operation are terminated. For details on handling *x*37 abends, see the description of the RETRY operand under "Operands" on page 53.
- Related SYSMOD failure. When SMP/E excludes an element from a SYSMOD because another SYSMOD being processed supplies a higher level of the element, SMP/E 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 are terminated because of a "related SYSMOD failure."

Avoiding SYSMOD termination

BYPASS

Certain error conditions causing the termination of a SYSMOD can be avoided by specifying the BYPASS operand on the APPLY command. 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 value SMP/E uses to determine the success or failure of a called utility program is kept in the UTILITY entries and can be changed by UCLIN.

APPLY termination

APPLY processing termination causes a return code of 12. For each of the following conditions, SMP/E 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 list, and one specifies the other in the DELETE operand of its ++VER statement.
- Two function SYSMODs are specified in the SELECT list or are selected in mass mode, and one specifies the other in the NPRE operand of its ++VER statement.
- A function SYSMOD specifying a previously applied SYSMOD in the NPRE operand of its ++VER statement is specified in the SELECT list.
- A function SYSMOD deleted by a previously applied SYSMOD (that is, a SYSMOD entry in the target zone indicates that the SYSMOD has been deleted) is specified in the SELECT list.
- A function SYSMOD superseded by a previously applied SYSMOD (that is, a SYSMOD entry in the target zone indicates that the SYSMOD is superseded) is specified in the SELECT list. A service SYSMOD in the same situation is not processed, but the APPLY command is not terminated.

Automatic reapplication of SYSMODs

An applied SYSMOD can be selected for reapplication when a function SYSMOD is 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 this PTF was first applied when only function GVT3100 was installed, the first ++VER statement would have been used, and the conditional requisite data supplied on the ++IF would have been saved. If GVT3101 is installed later, the saved ++IF data requires this same PTF to be installed.

Note: Because SMP/E does not process a SYSMOD with more than one VER that appears to be valid, GVT3101 must DELETE GVT3100 in order for this construction to work properly.

Applying maintenance to a module in an LLA managed library

When you apply successive maintenance to the same load module in a library lookaside (LLA) managed library, you must refresh LLA or use the NOFREEZE option. Otherwise, you may have problems using APPLY commands for multiple SYSMODs that update load modules managed by LLA. These same problems can also occur with reruns when some of the SYSMODS updating a load module had errors the first time through.

Note: By default, the LNKLST is managed by LLA as the equivalent of FREEZE mode. The CSVLLA*xx* member can be used to make the entire LNKLST or specific data sets to be managed by LLA as NOFREEZE.

If you issue APPLY commands that make multiple changes to a member of a partitioned data set managed by LLA in the FREEZE mode, the base for each APPLY command is the same as the previous APPLY command and does not include the previous APPLY command updates. SMP/E issues a BLDL to get the input required for its APPLY processing. If the library is managed by LLA in the FREEZE mode, BLDL requests the directory entry from LLA. The updated module built by SMP/E is usually placed beyond the end of the last member of the partitioned data set. This address is different from that specified in the directory of the PDS and the LLA managed tables for the module. The directory entry on disk is updated at the end of the APPLY processing. The LLA directory entry for the module does not change until LLA is restarted or a MODIFY LLA command is issued for this library.

The recommended way to update a module in a LLA managed data set in FREEZE mode is:

1. Use IEBCOPY to copy the module to another data set
2. Issue the APPLY command against the copied member
3. Copy the updated member back to the LLA managed production library.
4. Refresh LLA by using the MODIFY LLA,UPDATE= *xx* command, where the CSVLLA *xx* member contains a LIBRARIES statement identifying the data set whose LLA directory must be updated to include the new TTR for the changed module.

Another way to update a module in a LLA managed data set in FREEZE mode is:

1. Use the MODIFY LLA,UPDATE= *xx* command, where the CSVLLA *xx* member contains a REMOVE statement identifying the data set containing the load module to be updated
2. Issue the APPLY command
3. Use another MODIFY LLA,UPDATE= *xx* command, where the CSVLLA *xx* member contains a LIBRARIES statement identifying the updated data set to be added back to LLA.

To compress an LLA managed library (FREEZE or NOFREEZE):

1. Use the MODIFY LLA,UPDATE= *xx* command, where the CSVLLA *xx* member contains a REMOVE statement identifying the data set to be compressed

2. Compress the data set
3. Use another MODIFY LLA,UPDATE= *xx* command, where the CSVLLA *xx* member contains a LIBRARIES statement identifying the compressed library to be returned to LLA management.

Output

The following reports can be produced during APPLY processing:

- Bypassed HOLD Reason Report
- Causer SYSMOD Summary report
- Cross-Zone Summary report
- Deleted SYSMOD report
- File Allocation report
- Element Summary report
- JCLIN Cross-Reference report
- JCLIN Summary report
- MOVE/RENAME/DELETE report
- SYSMOD Regression report
- SYSMOD Status report
- Summary of Bypassed and Unresolved HOLD Reason Report
- Unresolved HOLD Reason Report

See Chapter 34, "SMP/E reports," on page 457 for descriptions of these reports.

APPLY processing may also create library change file records that reflect any successful utility work performed by APPLY processing to update target libraries. For more information about library change file recording, see *z/OS SMP/E Reference*.

Examples

The following examples are provided to help you use the APPLY command.

Example 1: Applying all SYSMODs from a given source

If the SOURCEID operand was used during RECEIVE processing to group all those SYSMODs processed, you can choose to install only that set of SYSMODs. This can be done with the SOURCEID operand of the APPLY command. Suppose you received an ESO containing service levels PUT0701 and PUT0702. The ESO contained ++ASSIGN statements that assigned each PTF a SOURCEID value corresponding to the service level that it is part of. Now you want to install all the applicable PTFs from those tapes into the target libraries described by zone MVSTST1. You can do this with the following commands:

```
SET      BDY(MVSTST1)        /* Process MVSTST1 tgt zone.  */.
APPLY    SOURCEID(PUT0701,   /* Process these service      */
                  PUT0702)   /* levels                     */
         GROUP               /* and any requisites.        */.
```

Example 2: Applying all SYSMODs for selected functions

At times, you may want to install changes only for a single function or for a certain group of functions. This can be done with the FORFMID operand of the APPLY command. Assume that you want to install service for function JXX1234 and for all the telecommunication-related functions on your system. You first need to define an FMIDSET for the telecommunication functions. You can do this with the following commands:

```
SET      BDY(GLOBAL)         /* Process global zone.       */.
UCLIN                        /* UCLIN to set up
                                FMIDSET.                   */.
ADD      FMIDSET(TC)         /* Define TC FMIDSET.         */
```

```
          FMID(JXX0001       /* Include these FMIDs.       */
               JXX0002)      /*                            */.
ENDUCL                       /* End UCL set up.            */.
```

You can now use the following commands to install PTFs for function JXX1234 and for the functions in FMIDSET TP:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    FORFMID(JXX1234,   /* Apply for selected FMIDs.  */
                 TC)        /*                            */.
```

Example 3: Applying APARs and USERMODs

You may want to install just corrective fixes (APAR fixes) or user modifications (USERMODs) into the target libraries. This can be done with the APARS and USERMODS operands of the APPLY command. Assume that you want to install all APAR fixes and USERMODs for function HXY2102. You can do this with the following commands:

```
SET      BDY(MVSTGT1)       /* Process MVSTGT1 tgt zone.  */.
APPLY    FORFMID(HXY2102)   /* Apply for this FMID        */
         APARS              /* all APARs                  */
         USERMODS           /* and all USERMODs.          */.
```

The APARS and USERMODS operands indicate that SMP/E is to pick only APAR fixes and USERMODs.

Note: If you want to install specific APAR fixes or USERMODs, use the SELECT operand. You do not need to specify APARS or USERMODS, which install **all** SYSMODs of the specified type.

Example 4: Applying with the GROUP operand

At times, you may know that a particular SYSMOD is required on your system, but you may not know all its requisite SYSMODs. By using the GROUP operand of APPLY, you can have SMP/E determine all the requisites and automatically install them. This method is often used during the installation of new functions. Suppose you want to install a new function, HYY2102, plus all its service, plus any requisite SYSMODs. You can do this with the following commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    FORFMID(HYY2102)   /* For one function.          */
         FUNCTIONS PTFS     /* Function and PTFs          */
         GROUP              /* plus requisites.           */.
```

The FORFMID operand indicates that only SYSMODs applicable to this function should be installed. The FUNCTIONS operand indicates that HYY2102 can be installed. The PTFS operand indicates that only PTFs for HYY2102 should be installed (no APAR fixes or USERMODs are included). The GROUP operand indicates that **all** requisite SYSMODs should also be applied. These requisites can be applicable to other functions, but cannot be APAR fixes or USERMODs.

Example 5: Applying with GROUPEXTEND

Assume that you want to use have SMP/E automatically include the requisites for some SYSMODs you plan to install. However, you are not sure whether all the requisites are available. (They might not be received, or they may be held because they are in error.) In those cases, you want SMP/E to check whether a superseding SYSMOD is available for the unsatisfied requisites. To have SMP/E do this additional checking, you can use the GROUPEXTEND operand:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    FORFMID(HYY2102)   /* For one function.          */
         FUNCTIONS PTFS     /* Functions and PTFs plus    */
         GROUPEXTEND        /* requisites or supersedes.  */.
```

SMP/E applies HYY2102 and any functions or PTFs applicable to HYY2102. Because of the GROUPEXTEND operand, SMP/E also applies all requisites for those SYSMODs, even those not applicable to HYY2102. If SMP/E cannot find a requisite, it looks for a SYSMOD superseding the requisite and uses

it to satisfy the requirement. Likewise, if a requisite is held for an error reason ID, SMP/E looks for a SYSMOD superseding the requisite, or that either satisfies or supersedes that error reason ID, and uses it to satisfy the requirement.

Example 6: Applying with the CHECK operand

In Example 4, SMP/E was directed to automatically include SYSMODs needed for the selected function and service. At times, you may want to review which SYSMODs is included before you actually install them. This can be done by using the CHECK operand of APPLY, as in the following commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    FORFMID(HYY2102)   /* For one FMID.              */
         FUNCTIONS PTFS     /* Functions and PTFs         */
         GROUP              /* plus requisites            */
         CHECK              /* in check mode.             */.
```

After running this command, check the SYSMOD Status report to see which SYSMODs would have been installed if you had not specified CHECK. If the results of this trial run are acceptable, run the commands again, without the CHECK operand, to actually install the SYSMODs.

Example 7: Combining APPLY operands

If you want to further divide the work that is to be done, you can specify combinations of the APPLY operands. The following is an example using all the SYSMOD selection operands of APPLY:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    SOURCEID(PUT0701   /* For these service levels.  */
                  PUT0702)  /*                            */
         FORFMID(HYY2102    /* For selected functions     */
                 TP)        /*                            */
         FUNCTIONS PTFS     /* install all type SYSMODs   */
         APARS USERMODS     /*                            */
         SELECT (UZ00001    /* plus these three for       */
                 UZ00002    /* other functions,           */
                 UZ00003)   /*                            */
         EXCLUDE(UZ00010    /* but not these three,       */
                 UZ00011    /*                            */
                 UZ00012)   /*                            */
         GROUP              /* plus all requisites.       */.
```

This APPLY command causes SMP/E to apply all the SYSMODs that came from either service level PUT0701 or PUT0702 (from the SOURCEID operand) and that are applicable either to function SYSMOD HYY2102 or to one of the function SYSMODs identified in the FMIDSET entry “TP” (specified on the FORFMID operand), plus any other SYSMODs required to install those SYSMODs (specified on the GROUP operand). All SYSMOD types are eligible for selection. (This includes the FUNCTIONS, PTFS, APARS, and USERMODS operands.) In addition, the selected SYSMODs (UZ00001, UZ00002, and UZ00003) are applied even though they came from a source other than the two specified service levels and even though they may belong to function SYSMODs other than those specified (from the SELECT operand). SMP/E does not install SYSMODs UZ00010, UZ00011, or UZ00012, even though they may be applicable to the functions specified and have one of the two SOURCEID values or may be required to install other SYSMODs that are eligible (from the EXCLUDE operand).

Example 8: Installing service for all ESO service levels

Assume that you want to install the preventive service received from all ESO tapes into zone MYZONE1 without having to specify all possible ESO service levels on the SOURCEID operand. You can use the following commands:

```
SET      BDY(MYZONE1)       /* Process MYZONE1 tgt zone.  */.
APPLY    PTFS               /* Install all PTFs           */
         SOURCEID(PUT*)     /* for all service levels     */
         CHECK              /* in check mode.             */.
```

Example 9: Excluding SYSMODs with certain source IDs

Assume that you have received an ESO with PTFs up to service level PUT0703, and now you want to install service from all but the latest two service levels (PUT0702 and PUT0703) into zone MYZONE2. You can use the following commands:

```
SET      BDY(MYZONE2)       /* Process MYZONE2 tgt zone.  */.
APPLY    PTFS               /* Install all PTFs           */
         SOURCEID(PUT*)     /* for all service levels     */
         EXSRCID(PUT0702    /* except for PUT0702         */
                 PUT0703)   /* and PUT0703,               */
         GROUPEXTEND        /* and any requisites         */
         CHECK              /* in check mode.             */.
```

Example 10: Bypassing system reason IDs

Assume that you have received the SYSMODs for service level 0701. For some of them, ++HOLD statements specified a system reason ID of ACTION, indicating that you need to take certain actions before installing the SYSMODs (the required actions were described in the comments for the ++HOLD statements). You have completed the necessary actions for each SYSMOD. Now you are ready to apply them. You can use the following commands:

```
SET      BDY(MYZONE2)        /* Process MYZONE2 tgt zone.  */.
APPLY    PTFS                /* Install all PTFs           */
         SOURCEID(PUT0701)   /* for service level 0701.    */
         BYPASS(             /* Bypass holds for all       */
         HOLDSYSTEM(ACTION)) /* SYSMODs held for ACTION.   */.
```

Suppose, instead, you have completed the necessary actions for only certain SYSMODs in service level 0701 (PTFs UZ12345 and UZ34567). You are ready to apply those specific held SYSMODs, but want the other SYSMODs requiring actions to be held from APPLY processing. To limit the SYSMODs for which the hold is bypassed, specify the desired SYSMOD IDs with the ACTION reason ID:

```
SET      BDY(MYZONE2)        /* Process MYZONE2 tgt zone.  */.
APPLY    PTFS                /* Install PTFs               */
         SOURCEID(PUT0701)   /* for service level 0701.    */
         BYPASS(             /* Bypass holds for specific  */
         HOLDSYSTEM(ACTION(  /* SYSMODs held for ACTION:   */
         UZ12345,UZ34567)))  /* List them here.            */.
```

Example 11: Excluding SYSMODs selected with an FMIDSET

A SYSMOD ID defined in an FMIDSET specified on the SELECT list may be excluded from processing with the EXCLUDE operand, as shown in this example:

If FMIDSTX contains FUNC001, FUNC002, FUNC003, and FUNC004, then, to APPLY all but FUNC003, the command would be:

```
APPLY SELECT(FMIDSTX) EXCLUDE(FUNC003).
```

Example 12: Automatic release of system hold when ++HOLD keeps the originating SYSMOD ID

Suppose you encounter a problem that is eventually identified as an APAR (OZ00456) that had already been reported and fixed in module MOD1234. IBM support gave you the number of a PTF (UZ00999) that contains a fix for the APAR. You then checked to see if the PTF was available on the system having the problem. It was, but UZ00999 contained the following ++HOLD that had to be addressed (UZ00999 superseded UZ00111 and thereby had inherited its ++HOLD):

```
++HOLD(UZ00111) SYSTEM REASON(ACTION) FMID(ABCDCBA) DATE(03255)
COMMENT( ENLARGE DATA SET SYS1.SMALLLIB BEFORE
INSTALLING THIS FIX IN ORDER TO AVOID AN E37 ABEND) .
```

You enlarged the data set and were now ready to install UZ00999. Of course, UZ00999 might need other PTFs installed along with it, so you decided to use the GROUP operand on the APPLY command. In order to get UZ00999 installed, you had to bypass the ACTION hold that UZ00999 contained. However, you did not want to inadvertently bypass any other ACTION holds that might be in effect for SYSMODs brought in by the GROUP operand, so you coded the BYPASS operand so that it would release only the ACTION hold for SYSMOD UZ00999.

The APPLY command that you used was:

```
APPLY S(UZ00999) GROUP BYPASS(HOLDSYSTEM(ACTION(UZ00999)).
```

It turned out that a PRE of UZ00999 (UZ00444) was not installed and was therefore included in the APPLY operation by the GROUP operand. It also happened that UZ00444 also contained module MOD1234, which contained the fix for OZ00456, as well as another problem. In fact, UZ00444 also contained the ++HOLD that was in UZ00999, because UZ00444 also superseded UZ00111 and had inherited its ++HOLD.

In this case, because the ++HOLD statements in UZ00444 and UZ00999 had kept the originating SYSMOD ID from being installed, and because UZ00999 was having the ++HOLD bypassed, SMP/E considers the ACTION hold to be addressed and will therefore automatically release the ACTION hold against UZ00444, even though you had not specifically named UZ00444 in the BYPASS operand.

Processing

APPLY processing is very similar to ACCEPT processing except that the target zone, rather than the distribution zone, controls processing, and the target libraries, rather than the distribution libraries, are updated.

SYSMOD selection

This section outlines the process by which SYSMODs and the elements from the SYSMODs are selected.

Operands related to SYSMOD selection

The following APPLY command operands can be used to specify to SMP/E which SYSMODs are to be processed:

APARS
EXCLUDE
EXSRCID
FORFMID
FUNCTIONS
GROUP
GROUPEXTEND
PTFS
SELECT
SOURCEID
USERMODS
XZREQ

Candidate selection

The SYSMOD selection operands of the APPLY command can be specified separately or in combination to control the SYSMODs that SMP/E is to process. When you specify them separately, SMP/E selects only SYSMODs meeting the one criterion specified. When you specify them in combination, SMP/E does the following checking to build the complete candidate list:

1. SMP/E checks the global zone and the specified target zone to determine which SYSMODs present in the global zone and SMPPTS have not already been applied to the target zone. For each such SYSMOD, SMP/E checks to see if it meets the criteria of any additional selection operands.
   a. If the EXCLUDE operand was specified, SMP/E makes sure the SYSMOD was not specified in the exclude list.
   b. If one or more of the SYSMOD type operands (that is, FUNCTIONS, PTFS, APARS, or USERMODS) were specified, SMP/E checks to make sure the SYSMOD type was one of those specified. If no type operand was specified, the default is for SMP/E to process only PTF SYSMODs.
   c. If the FORFMID operand was specified, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values in a specified FMIDSET.
   d. If the SOURCEID operand was specified, SMP/E makes sure one of the source IDs of the SYSMOD matches a source ID that was either explicitly or implicitly specified on the SOURCEID operand.
   e. If the EXSRCID operand was specified, SMP/E makes sure none of the source IDs of the SYSMOD matches a source ID that was either explicitly or implicitly specified on the EXSRCID operand.

      Note: If a given SYSMOD has multiple source IDs and you specify at least one of them, implicitly or explicitly, on the SOURCEID operand, and you specify another one implicitly or explicitly, on the EXSRCID operand, that SYSMOD is excluded from processing.

      Similarly, if a given source ID is implicitly or explicitly specified on the EXSRCID operand and also on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.

   Each SYSMOD satisfying all these conditions is a candidate for the APPLY process. In other words, specifying the SYSMOD type operands, the FORFMID operand, or the SOURCEID operand in combination causes SMP/E to select those SYSMODs meeting all the specified conditions.
2. If you specify the SELECT operand, each SYSMOD specified in the select list is a candidate regardless of its SYSMOD type, FMID value, or SOURCEID value. That is, SELECT has an additive effect on the SYSMOD selection. This is called *select-mode* processing. If SELECT is not specified, this is called *mass-mode* processing.

   Note: If SELECT is the only specified operand, SMP/E processes only the SYSMODs in the select list.

3. If you specify the XZREQ operand, unsatisfied cross-zone requisites that are needed in the set-to zone become candidates for installation. These SYSMODs are in addition to other SYSMODs that are chosen due to other operands, such as FORFMID and SOURCEID. If FORFMID is specified, only cross-zone requisites for the FMIDs specified on the FORFMID operand become candidates for installation. Other operands on the command, such as EXSRCID, have no effect on which cross-zone requisites become candidates for installation.
4. If the GROUP or GROUPEXTEND operand was specified, SMP/E checks each of the candidate SYSMODs to determine if it has any requisites that must also be applied. SMP/E automatically includes the following SYSMODs, as long as they were not specified on the EXCLUDE operand or excluded by the EXSRCID operand:
   a. Any SYSMOD not already applied and specified as a prerequisite (that is, specified in the ++VER statement PRE operand) of one of the candidate SYSMODs
   b. Any SYSMOD not already applied and specified as a corequisite (that is, specified in the ++VER statement REQ operand) of one of the candidate SYSMODs
   c. Any SYSMOD not already applied and specified as a conditional requisite (that is, specified in the ++IF statement REQ operand) of one of the candidate SYSMODs
   d. Any SYSMOD not already applied and specified as a conditional requisite (CIFREQ subentry) in the target zone SYSMOD entry for one of the candidate SYSMODs

   If one of these requisites is held or not available, and you specified GROUPEXTEND, SMP/E checks the global zone for any SYSMODs that have been received and that either supersede the requisite, or

satisfy or supersede its HOLDERROR reason ID. If so, the lowest-level SYSMOD found is used to satisfy the requisite.

- If you specified NOAPARS with GROUPEXTEND, SMP/E does not include APARs that resolve error reason IDs for the held requisites.
- If you specified NOUSERMODS with GROUPEXTEND, SMP/E does not include USERMODs that resolve error reason IDs for the held requisites.

Once a SYSMOD is added to the candidate list, it is eligible to be checked for additional requisites. The FORFMID, SOURCEID, and SYSMOD type operands have no effect on SYSMODs brought in because of the GROUP or GROUPEXTEND operand. The following example applies all those SYSMODs with a source ID of PUT0701 that are applicable to EBB1102, plus any additional SYSMODs that are required, even though their source ID is not PUT0701 or their FMID is not EBB1102:

```
SET      BDY(TGT1)          /* Set to target zone.   */.
APPLY    SOURCEID(PUT0701)  /*                       */
         FORFMID(EBB1102)   /*                       */
         GROUP              /*                       */.
```

Applicability checking

Once the APPLY candidate list is completed, SMP/E does the following checking to make sure the selected SYSMODs are applicable to the system:

General applicability

SMP/E makes sure that each SYSMOD meets certain requirements before it is applied to the system.

1. Each SYSMOD must contain **at least one** ++VER statement whose SREL value matches one of the SREL values for that target zone and whose FMID value (if present) names a function SYSMOD that has been (or is being) applied. Function SYSMODs having no ++VER statement FMID operand are applicable if the SREL matches.

   Each SYSMOD must have **at most one** ++VER statement whose SREL matches the SREL in the target zone entry and whose FMID value exists in the target zone (or is being applied) and has not been superseded.

   If a SYSMOD is found containing multiple applicable ++VER statements, SMP/E is unable to apply that SYSMOD, because SMP/E cannot determine which ++VER statement to use.

   Note: SMP/E does not consider an FMID that has been superseded to be applied. Therefore, a SYSMOD can contain two ++VER statements that apply to two functions, one of which supersedes the other.

2. The SYSMOD must not have already been applied successfully to the target zone.
   - To reapply a SYSMOD, you must specify it in the SELECT operand and include the REDO operand on the APPLY command.
   - SYSMODs that have been partially applied during a previous invocation are considered eligible for processing without any special action. These SYSMODs are identified by the APPLY ERROR indicator in their target zone SYSMOD entry.
3. The SYSMOD must not have been partially restored during a previous SMP/E RESTORE attempt. These SYSMODs are identified by the RESTORE ERROR indicator in their target zone SYSMOD entry.
4. The SYSMOD must not be superseded by a previous SYSMOD. If a SYSMOD is found to be superseded by another SYSMOD being applied, no elements are selected from the superseded SYSMOD.
5. The SYSMOD must not have been explicitly deleted by a previous SYSMOD.

Unconditional requisites (PRE and REQ)

Unconditional requisites are SYSMODs that are required in all functional environments. All the unconditional requisites for each SYSMOD must be satisfied. Unconditional requisites are those specified in the SYSMOD's ++VER statement PRE and REQ operands. A requisite is considered satisfied if:

- The requisite SYSMOD is already applied.
- The requisite SYSMOD is superseded by a SYSMOD that is already applied.
- The requisite SYSMOD is being applied.
- The requisite SYSMOD is superseded by a SYSMOD being applied.

Conditional requisites (IFREQ)

Conditional requisites are SYSMODs required only for a particular functional environment. All conditional requisites of each SYSMOD must be resolved. Conditional requisites are specified on the ++IF statement immediately following the applicable ++VER statement. If the function specified in the FMID operand of the ++IF statement is applied or is being applied, each SYSMOD specified in the ++IF statement REQ operand must be satisfied. These requisites are satisfied in the same manner as unconditional requisites.

If the function specified in the FMID operand of the ++IF statement is not already installed, in order to make sure these requisites are satisfied when the function is installed, SMP/E saves the information from the ++IF statement as CIFREQ subentries in the target zone SYSMOD entry for that function. When the function is applied, SMP/E checks the CIFREQ subentries for requisites supplied by previously applied SYSMODs and makes sure these requisites are satisfied.

Cross-zone requisites

Cross-zone requisites are very similar to conditional requisites. Like conditional requisites, they are also caused by an ++IF statement. For a cross-zone requisite, however, the SYSMOD containing the ++IF exists in one zone, but the function and SYSMODs identified by the FMID and REQ operands specified on the ++IF statement are in another zone.

Negative requisites (NPRE)

If the NPRE operand is specified on a SYSMOD's ++VER statement, the SYSMOD ID specified must not already be installed, must not be installed concurrently, and must not be superseded by a SYSMOD being installed concurrently.

Note: The NPRE operand is valid only in function SYSMODs and is used to specify one or more mutually exclusive functions.

Superseding SYSMODs (SUP)

SMP/E checks to make sure that the SYSMOD has not been superseded by another SYSMOD that is already installed or by another SYSMOD being applied concurrently. If the SYSMOD is superseded, it is not applied, and the superseding SYSMOD is used instead.

Note: If the superseding SYSMOD is not processed because it is held or excluded or because it has missing requisites, processing continues as though the SYSMOD did not exist. The SYSMOD that would have been superseded is installed instead.

If a SYSMOD that is already applied is being superseded, SMP/E ensures each of the elements contained in the superseded SYSMOD are also contained in either the superseding SYSMOD or in a SYSMOD from the requisite set for the superseding SYSMOD (unless the element is being deleted by the superseding SYSMOD).

Exception SYSMODs (HOLD)

SMP/E makes sure each SYSMOD has no unresolved exception data associated with it. Exception data is information specified on the ++HOLD statement. Each ++HOLD statement has a REASON operand specifying a character string that identifies the reason why the SYSMOD has been put into exception status. The following types of exception data are supported by SMP/E:

- HOLDERROR
- HOLDFIXCAT
- HOLDSYSTEM (internal and external)
- HOLDUSER

In addition, the ++HOLD statement may contain a CLASS operand, which specifies an alternative way to resolve exception data through the use of the BYPASS operand.

Exception data is considered resolved when one or more of the following conditions are true:

- HOLDERROR and HOLDFIXCAT exception data is considered resolved if any of the following conditions apply:
  - The SYSMOD named as the reason ID for the exception is already applied or is superseded by a SYSMOD that is already applied.
  - The SYSMOD named as the reason ID for the exception is being applied concurrently or is being superseded by a SYSMOD being applied concurrently.
  - The applicable BYPASS operand is specified.
- HOLDSYSTEM (internal) exception data is considered resolved if any of the following conditions apply:
  - The SYSMOD ID specified on the ++HOLD defining the exception is already applied or is superseded by a SYSMOD that is already applied.
  - The SYSMOD ID specified on the ++HOLD defining the exception is being superseded by a SYSMOD being applied concurrently that also contains the same ++HOLD, but the ++HOLD has been bypassed for the concurrently applied SYSMOD (see "Example 12: Automatic release of system hold when ++HOLD keeps the originating SYSMOD ID" on page 76).
  - The applicable BYPASS operand is specified.
- HOLDSYSTEM (external) exception data is considered resolved if the applicable BYPASS operand is specified.
- HOLDUSER exception data is considered resolved if the applicable BYPASS operand is specified.

If all the exception data associated with a given SYSMOD is not resolved, SMP/E does not apply that SYSMOD. The SYSMOD is treated as though it had been specifically excluded. Messages are issued showing which exception data is not resolved, and the SYSMOD and reason IDs associated with the exception data are displayed in the SYSMOD Summary report.

Each category of exception data is resolved differently:

- For HOLDERROR exception data the reason ID is actually the number of the APAR that caused the SYSMOD to be placed in exception status. As subsequent service is processed, the APAR will probably be superseded by a PTF. When this happens, the exception data is resolved and the first PTF is automatically processed. Therefore, it is generally not necessary to use the BYPASS operand to process SYSMODs with error reason IDs.

  During any mass installation of SYSMODs, it should be expected that some SYSMODs are not applied, because unresolved APARs are associated with them. During the installation of preventive service, these SYSMODs should not be investigated further; they will be installed later when a subsequent SYSMOD is produced that supersedes the reason ID associated with the exception data that is causing them to be held.

  During the installation either of corrective service (that is, installing a PTF or an APAR because of a known problem in the system) or of a new function specifically requiring a SYSMOD, the reason IDs associated with the HOLDERROR exception data should be taken as the first piece of data for research. Research may provide a fix for the problem, in which case the SYSMOD and the fix can be applied concurrently. If a fix is not available, you can either wait for one, or apply the SYSMOD using the appropriate BYPASS operand.
- For HOLDFIXCAT exception data, the reason ID is the number of the APAR that caused the SYSMOD to be placed in the exception status. The APAR is associated with one or more fix categories. It is optional whether the HOLDFIXCAT exception data affects processing for the held SYSMOD, based on the fix

categories of the HOLDFIXCAT and the fix categories of interest specified by the user. The fix categories of interest are specified on the FIXCAT operand or in the FIXCAT subentry of the active OPTIONS entry. If a fix category value on the HOLDFIXCAT exception matches any of those of interest to the user, then the held SYSMOD is not applied until the APAR reason ID is resolved. If there are no matching fix categories, then the SYSMOD is not held for that HOLDFIXCAT exception.

For HOLDFIXCAT exceptions that must be resolved, the APAR is considered resolved when any one of the following conditions is true:

- The SYSMOD named as the reason ID (the APAR) is already applied or has been superseded by a SYSMOD that is already applied.
- The SYSMOD named as the reason ID (the APAR) is being applied concurrently or is being superseded by a SYSMOD that is being applied concurrently.
- An applicable BYPASS operand of HOLDCLASS or HOLDFIXCAT is specified.

- For HOLDSYSTEM (internal) exception data the SYSMOD ID specified on the ++HOLD MCS may be either
  - the SYSMOD ID of the containing SYSMOD **or**
  - a SYSMOD ID of a SYSMOD superseded by the containing SYSMOD.

  When it is the latter, the reason that the current SYSMOD contains the ++HOLD is because it was originally in the SYSMOD whose SYSMOD ID appears on the ++HOLD and that SYSMOD has been superseded by the current SYSMOD. The exception data is considered resolved if the SYSMOD specified on the ++HOLD is already installed or is being superseded by another SYSMOD that is being installed concurrently with the held SYSMOD. When this is the case, it is assumed that the user has already addressed the reason for the hold.

  When it is the former, the held SYSMOD should be applied by use of the BYPASS(HOLDSYS(reason-id)) operand once the reason for the hold is addressed.
- For HOLDSYSTEM (external) exception data the associated reason ID is a 1- to 7-character string used to identify some action that must be taken before or after a SYSMOD is installed. System reason IDs are not SYSMOD IDs and are not specified in the supersede list of a SYSMOD. Therefore, SMP/E does not automatically release them.

  SYSMODs held in this manner should be applied by use of the BYPASS(HOLDSYS"reason-id¨) operand. If you were to remove the system reason ID by using the ++RELEASE statement, you would then be able to install the SYSMOD, but you would also lose the information about any special processing required in order to apply that SYSMOD on another system.
- For HOLDUSER exception data the associated reason ID is a 1- to 7-character string meaningful to the user. User reason IDs are not SYSMOD IDs and are not specified in the supersede list of a SYSMOD. Therefore, SMP/E does not automatically release them.

  SYSMODs held in this manner should be applied by use of the BYPASS(HOLDUSER) operand. If you were to remove the hold associated with user reason ID by using the ++RELEASE statement, you would then be able to install the SYSMOD, but you would also lose sight of the fact the SYSMOD has some special significance to the you, the user.

SYSMOD installation

After determining which SYSMODs are to be applied, SMP/E performs the following tasks to install the SYSMODs:

1. Determine the order in which the SYSMODs should be processed.
2. Perform delete processing for any SYSMODs with the DELETE operand specified on their ++VER statement.
3. Move, delete, or rename specified elements or load modules.
4. Process any inline JCLIN.
5. Select elements from the SYSMODs to be applied.

   Note: The previous three items are combined and are performed as each SYSMOD is processed.

6. Call system utilities to install the selected elements.
7. Update the applicable target zone entries.
8. Produce summary reports identifying all completed processing.

The following sections describe each of these tasks in greater detail.

SYSMOD processing order

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

The order in which the SYSMODs being applied are processed is determined from the prerequisite (PRE) data supplied on the ++VER statement for the SYSMOD. SYSMODs named as prerequisites are processed before the SYSMODs naming them.

If no prerequisite order can be determined between SYSMODs, function SYSMODs are processed first, followed by service SYSMODs (PTFs, then APARs, then USERMODs).

Deleted SYSMODs

A function SYSMOD can delete another function by naming the function to be deleted as an operand of the ++VER DELETE operand. SMP/E deletes that function and all functions, PTFs, APARs, and USERMODs dependent on it. The functions specifically named in the DELETE operand list are considered *explicitly* deleted SYSMODs; all SYSMODs deleted because of their dependence on the explicitly deleted SYSMODs are termed *implicitly* deleted SYSMODs.

When one function SYSMOD deletes another, SMP/E attempts to remove from the target zone all information related to the deleted SYSMOD. In addition, SMP/E removes from the target libraries all elements currently owned by the deleted function SYSMOD. The following processing is done:

1. SMP/E determines whether there are any function SYSMODs in the hierarchy of the function SYSMOD being deleted, and considers those function SYSMODs to also be eligible for delete processing.
2. SMP/E deletes any SYSMOD having the same FMID value as one of the function SYSMODs being deleted.
3. SMP/E determines all the elements that are currently owned by one of the function SYSMODs to be deleted.
4. SMP/E deletes from the target libraries all the elements of the SYSMODs to be deleted. After the elements are successfully deleted, SMP/E deletes the element entries from the target zone.

   If a module is being deleted, SMP/E checks whether the module is contained in any cross-zone load modules. If so, SMP/E deletes the contents of the MOD entry, except the XZLMOD subentries. If the module is not reintroduced and the AUTOMATIC option is in effect for the cross-zone, the module is deleted from the cross-zone load module during cross-zone processing. For more information, see *z/OS SMP/E Reference*.
5. For load modules composed entirely of modules that are to be deleted, SMP/E deletes the load modules and any aliases for the load modules from the target libraries. If the load modules were successfully deleted, SMP/E deletes the MOD and LMOD entries from the target zone. The load modules are also deleted from the SMPLTS, if applicable.

   Note: If all the modules in a load module are being deleted or replaced, SMP/E checks whether the load module contains cross-zone modules. If so, SMP/E does **not** delete the load module. Instead, it removes the deleted modules from the load module (leaving a *stub* load module in the target libraries) and leaves the LMOD entry in the target zone.
6. For load modules composed of modules to be deleted and modules not to be deleted, SMP/E delinks the CSECTs in the deleted modules from the load module. In priority order, SMP/E obtains the CSECT information in the following manner:

   a. By gathering the list of CSECT names in the target zone MOD entry

b. By determining the CSECT operand on the ++MOD statement from the lowest function or service level SYSMOD being installed that contains that module

c. By assuming that the CSECT name is equal to the distribution library name

The MOD entries are deleted from the target zone, but the LMOD entry remains because it is still needed to process SYSMODs affecting the modules that have not been deleted.

For each deleted module, SMP/E adds to the LMOD entry a MODDEL subentry for the module name. MODDEL subentries document the connection between the deleted modules and the LMOD. If any of these deleted modules are ever reintroduced, SMP/E looks for LMODs with MODDEL subentries for the modules and automatically rebuilds the LMODs to include these modules again. The MODDEL subentries for the reintroduced modules are then removed from the LMOD entries.

Any aliases associated with the load modules are not deleted from the target libraries, but may be marked nonexecutable by the link-edit utility.

7. The target zone SYSMOD entries for all implicitly deleted SYSMODs are deleted. For each explicitly deleted SYSMOD, a target zone SYSMOD entry is created. This entry has a DELBY subentry naming the function that caused the deletion. The SYSMOD entries for the explicitly deleted SYSMODs prevent the deleted function SYSMODs from being reprocessed by APPLY.

   Note: For a function that both deletes and supersedes another function, the SYSMOD entry contains a SUPBY subentry instead of a DELBY subentry. This allows SYSMODs naming the deleted function as a requisite to still be installed.

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

Note: Always check the APPLY output to verify that all the CSECTs that were supposed to be deleted from a load module have been deleted.

In Figure 4 on page 85, function SYSMODs HDE1203, HDE1303, and HDE1403, and service SYSMODs UZ00009, UZ00010, and UZ00004 are deleted because DELETE(HDE1203) is specified on the ++VER statement. CIFREQ subentries in the SYSMOD entry for a function that is deleted (either explicitly or implicitly) are retained in the SYSMOD entry along with the DELBY subentry. So, when function HDE2000 is applied, CIFREQ subentries in the SYSMOD entry for function HDE1203 are retained, as are any CIFREQ subentries in the SYSMOD entries for functions HDE1303 and HDE1403. Likewise, any CIFREQ subentries for conditional requisites specified by the deleted SYSMODs are retained in the appropriate SYSMOD entries.

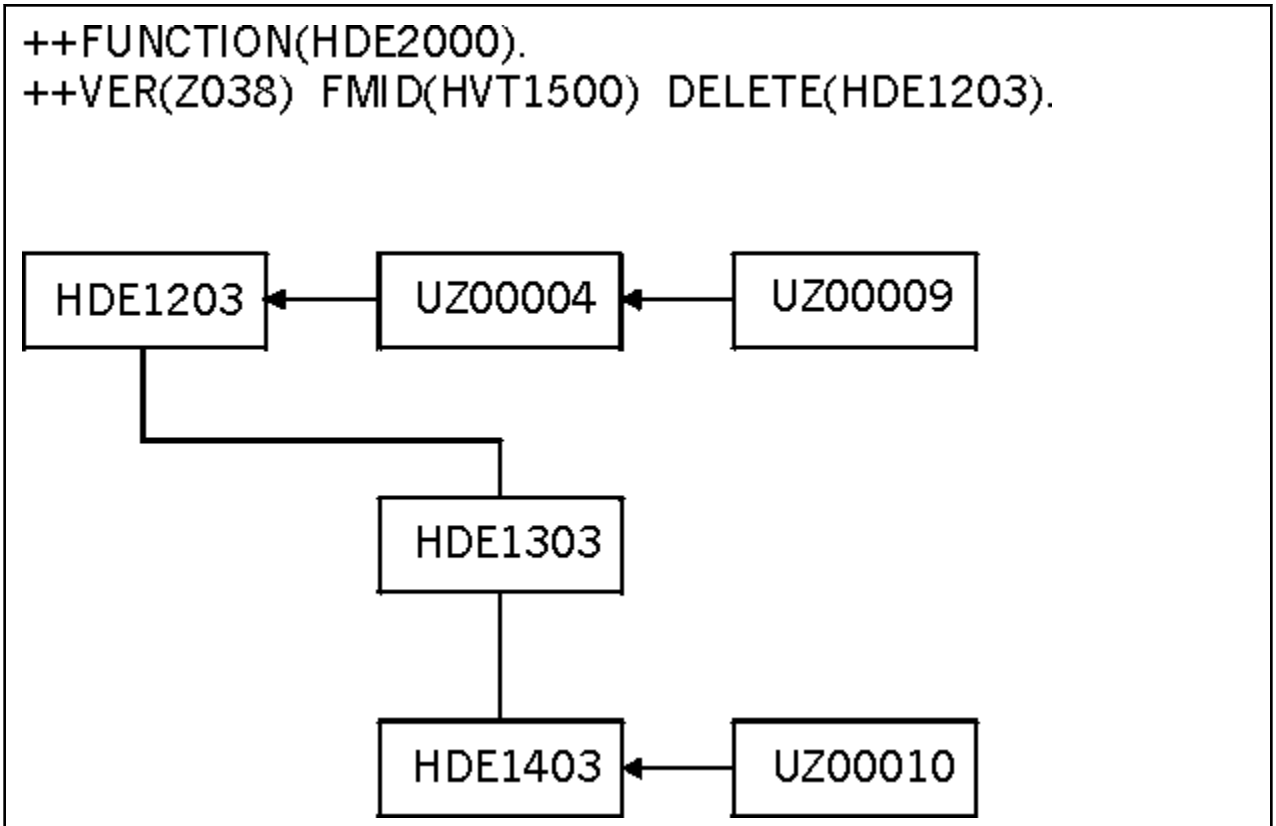


Figure 4. DELETE Hierarchy for DELETE(HDE1203): APPLY Processing

Note: Remember, SMP/E assumes that when a function is deleted, the deleting function replaces all the required elements of the deleted function. Although you can build a function SYSMOD that does nothing but delete another function, it is your responsibility to make sure your system remains functionally complete after the product has been deleted.

During APPLY processing, when a function is deleted from a target zone by another function, its FMID is not removed from the FMID list in the global zone. This is because the deleted function can still be applied in other target zones or accepted in other distribution zones.

Inline JCLIN

Inline JCLIN data for a SYSMOD is supplied following the ++JCLIN statement. JCLIN processing is done before element processing in order to prepare the target zone.

Each entry in the target zone that is affected by the JCLIN update is saved as BACKUP entries on the SMPSCDS before the update. These BACKUP entries record the SYSMOD ID of the SYSMOD that contained the inline JCLIN and the type of update performed.

Note:

1. Inline JCLIN is not processed for superseded or deleted SYSMODs.
2. Inline JCLIN does **not** cause SMP/E to update the target libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in the SYSMOD. The element statements in the SYSMOD determine which elements should be installed.
3. If SMP/E is creating an LMOD entry, and if it finds an existing LMOD entry that is for the same load module and that contains only cross-zone subentries:
   - SMP/E issues messages indicating that the cross-zone relationship might no longer be valid, and then deletes the cross-zone subentries from the load module.

- If deleting these cross-zone entries eliminates a TIEDTO relationship with a cross-zone, SMP/E deletes the associated TIEDTO value from the TARGETZONE entry for the set-to zone. For an explanation of the TIEDTO value, see the "TARGETZONE Entry" section in *z/OS SMP/E Reference*.
- Entries for related cross-zone modules are **not** updated to indicate that they are no longer part of the load module in the set-to zone.
- You must determine whether the cross-zone relationship is still valid. If so, reestablish it by using the LINK MODULE command. For more information about the LINK MODULE command, see Chapter 11, "The LINK MODULE command," on page 197.

The NOJCLIN operand on the APPLY command prevents the processing of inline JCLIN. NOJCLIN can be used if the JCLIN contains data that would overlay user-modified entries in the target zone.

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

For more information about JCLIN processing, see Chapter 9, "The JCLIN command," on page 153.

Moving, deleting, and renaming elements and load modules

Macros, modules, source, and load modules can be moved from one target library to another by use of the ++MOVE statement. In addition, load modules can be deleted from a target library by use of the ++DELETE statement, or renamed by use of the ++RENAME statement. This processing is done before element processing.

When ++DELETE is processed, the load module is deleted from the target libraries, and the LMOD entry is deleted from the target zone. For all of the modules included in the load module, the LMOD subentries are deleted from the MOD entries in the target zone. If the LMOD entry had a CALLLIBS subentry list, the load module is deleted from the SMPLTS data set (if necessary). If the LMOD entry had a side deck library subentry, the definition side deck is deleted from the side deck library.

When ++RENAME is processed, the load module is renamed in the target libraries and the LMOD entry is renamed in the target zone. If the LMOD entry has a side deck library subentry, the definition side deck is renamed in the side deck library. If the LMOD entry has a CALLLIBS subentry list, then :

- if the LMOD entry has a XZMOD subentry, the load module is renamed in the SMPLTS data set.
- if the LMOD entry does not have an XZMOD subentry, the UPGLEVEL subentry exists in the zone, and the load module exists in the SMPLTS data set, the load module is deleted from the SMPLTS data set.

Note: SMP/E will ignore a request to delete or rename a load module's definition side deck when the SIDE DECK LIBRARY subentry is SMPDUMMY.

Each entry in the target zone that is moved or renamed is saved in a BACKUP entry on the SMPSCDS before the update is performed. Each BACKUP entry in the SMPSCDS records the SYSMOD ID of the SYSMOD that contained the ++MOVE or ++RENAME statement and the type of update performed.

Note: No BACKUP entries are created when a load module is deleted.

Cross-zone load modules

If a SYSMOD being applied contains a ++RENAME statement for a load module containing cross-zone modules, SMP/E checks whether those zones indicate that cross-zone updates should be done automatically. If so, the cross-zone MOD entries are updated during cross-zone processing. If a SYSMOD being applied contains a ++DELETE statement for a load module containing cross-zone modules, SMP/E deletes the load module, as well as all the contents of the LMOD entry, except for the XZMOD and XZMODP subentries. These subentries are retained as a *stub* load module to preserve the cross-zone relationship they describe. Should this load module be reinstated at a later date, SMP/E will issue messages informing you of the previous cross-zone relationship. You can then decide whether this relationship is still valid and, if so, reestablish it with the LINK MODULE command, as described in Chapter 11, "The LINK MODULE command," on page 197.

The ++MOVE, ++DELETE, and ++RENAME statements are further described in the "SMP/E Modification Control Statements" section in *z/OS SMP/E Reference*.

Element selection

SMP/E uses the element statements provided in a SYSMOD to determine which elements should be installed in the target libraries. The selection of elements from a SYSMOD is based on relationships among SYSMODs being installed, other SYSMODs being installed, and modification identifiers of the corresponding elements installed on the target system. Three modification identifiers are kept for each element:

- FMID: Function modification identifier

  The FMID of an element is the function SYSMOD that owns the element. Generally, an element's FMID is established (and later changed) by the installation of a function SYSMOD. In this case, the element's FMID is the function SYSMOD that installed the element on the target system.

- RMID: Replacement modification identifier

  The RMID of an element is the last SYSMOD that replaced the element (or caused the element's FMID to change). An element's RMID is established by the SYSMOD that first introduces the element to the target system. The RMID of an element is changed by the installation of a SYSMOD that supplies a replacement for the element. Element replacements are ++MOD, ++MAC, ++PROGRAM, ++SRC, ++JAR, data element, and hierarchical file system element modification control statements, and modules resulting from assemblies.

- UMID: Update modification identifier

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

Note: Because data elements, hierarchical file system elements, and program elements can only be replaced and cannot be updated, they do not have UMIDs.

The purpose of element selection is to make sure that the correct functional level of each element is selected and that no service is inadvertently removed from the system.

Element selection in SMP/E is divided into three cases:

- The FMID of the SYSMOD being installed matches the FMID of the element on the target system.
- The FMID of the SYSMOD being installed differs from the FMID of the element on the target system.
- A function SYSMOD is being reinstalled.

The following sections describe processing for each case.

FMIDs match: MODID verification

In this case, SMP/E is dealing with elements belonging to the same function, and element processing is based on service relationships expressed by means of the PRE and SUP operands.

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 that supplied the same elements.

All elements

The SYSMOD being installed must specify the RMID of the associated target system element on the PRE or SUP operand. If the RMID of the target system element is the same as its FMID, the element has not been replaced by any SYSMOD, and so the SYSMOD being installed need not specify the RMID value in the PRE or SUP operand.

If the element being installed is a ++SRC/++SRCUPD or a ++MAC/++MACUPD resulting in an assembly that replaces a target system module (element), the SYSMOD being installed must specify as one of its PRE or SUP operand values the RMID of the corresponding target system module that is replaced by the assembly. If the target system module is itself the result of an assembly (RMIDASM indicator set in the MOD entry), an exception to this requirement is made, because the reassembly picks up any changes caused by the SYSMOD that last replaced the module through an assembly.

If the SYSMOD being processed does not specify the RMID of the element on the PRE or SUP operand, SMP/E does not apply that SYSMOD, because doing so would regress the service supplied by the SYSMOD represented by the RMID. If you want to allow the RMID SYSMOD to be regressed, you can specify the BYPASS(ID) operand of the APPLY command.

SMP/E makes an exception to this processing requirement if the SYSMOD being processed is itself the RMID of the element and REDO was specified on the APPLY command. In this case, the SYSMOD being processed is the highest level of the element and it is being installed again, therefore SMP/E allows the SYSMOD to be installed and to replace the current element.

Replacement elements

The SYSMOD being installed must be a prerequisite for, or must supersede, all UMIDs associated with the target system element.

Assemblies resulting from ++SRC/++SRCUPD and ++MAC/++MACUPD elements are considered replacement modules; the SYSMOD being installed must specify on the PRE or the SUP operand all UMIDs of the corresponding target system modules that are replaced by the assembly. No exception is made for a SYSMOD that does not specify on the PRE or the SUP operand all UMIDs associated with modules that have been assembled, because any UMIDs associated with the module are ZAPs that are overlaid by a new assembly.

If the SYSMOD being processed does not specify the UMID values of the elements on the PRE or the SUP operand, SMP/E does not apply that SYSMOD. If this SYSMOD were to be applied, each of the SYSMODs represented by those not specified in either the SUP or PRE operands would be regressed. To allow the regression, use the BYPASS(ID) operand on the APPLY command. The MODID check condition is then reported as a warning, and the elements are installed on the target libraries.

Update elements

When processing ++SRC/++SRCUPD and ++MAC/++MACUPD elements, it is assumed that previous updates are still there and will be incorporated with the current update. Therefore, a SYSMOD need not state a relationship (PRE or SUP) to a previous update. The SYSMOD being installed need not specify on the PRE or SUP operand the UMIDs associated with the corresponding target system element. If any element UMIDs in the target system are not specified in the SUP or PRE operands, 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. The warning is given because SMP/E is unable to determine with certainty that the two modifications in fact have a relationship or that there is an intersection. Thus, it is the responsibility of the developer or the service team (that is, whoever supplies the update type SYSMOD) to make sure that this SYSMOD specifies the correct relationships with all previous SYSMODs.

When processing ++JAR/++JARUPD elements, the SYSMOD must identify as a prerequisite or supersede the SYSMOD that last replaced the most recently selected ++JAR element (the RMID for the element). Therefore, the SYSMOD must specify as a PRE or SUP, the SYSMOD that last supplied a ++JAR for the element.

In addition, the SYSMOD supplying the ++JAR/++JARUPD elements must identify as a prerequisite or supersede all SYSMODs which have previously updated the most recently selected ++JAR element (the UMID for the element). In other words, the current SYSMOD must specify as a PRE or SUP, all SYSMODs that have supplied ++JARUPDs for the element since it was last replaced.

FMIDs differ

In this case, SMP/E is dealing with elements belonging to different functions and element selection is based on functional relationships expressed by FMID and VERSION. Elements can be excluded (that is, 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 is **excluded** from the SYSMOD being installed unless one of the following conditions is met:

- The function SYSMOD being installed names the FMID of the target system element in the ++VER FMID operand. In this case, the function being installed is superior to the function that owns the target system element; therefore, the element is selected.
- The MCS associated with the element from the SYSMOD being installed has a VERSION operand, and the FMID of the target system element is named in the VERSION list. In this case, the element from the SYSMOD being installed is considered functionally superior to the target system element, and it is selected.

  If there is no VERSION operand on the MCS of an element, the SYSMOD IDs named in the VERSION operand on the ++VER are used as previously described. In this situation, SMP/E may be dealing either with a function SYSMOD or with a nonfunction SYSMOD that is changing the functional ownership (FMID) of the elements.

  Note: If a SYSMOD containing an element update (++SRCUPD, ++MACUPD, ++JARUPD, or ++ZAP) attempts to change the ownership (FMID) of the element (with the VERSION operand), the SYSMOD cannot be installed.

When an element is selected, its FMID becomes that of the SYSMOD from which it is selected. No further MODID checking is done for these elements; SYSMODs are constructed so that when the functional ownership of a module changes, either the SYSMOD changing the ownership or the data stored in the target zone SYSMOD entries (the CIFREQ data) contains sufficient information to prevent any service or functional regressions.

Reinstalling a function

Element selection gets more complicated only for **function** SYSMODs that are being reinstalled and have elements that intersect with corresponding elements having the same FMID as themselves (see "FMIDs match: MODID verification" on page 87). The processing for this situation proceeds as in "FMIDs match: MODID verification" on page 87. When a MODID check error condition is detected, however, SMP/E checks further to determine whether the service level of the target system element is higher than that of the element from the SYSMOD being reinstalled. If so, the element from the SYSMOD being reinstalled is not selected, and processing of the SYSMOD continues. If not, the SYSMOD is terminated with a MODID check error.

APPLY CHECK processing

If the CHECK operand of APPLY was specified, processing stops at this point. SMP/E produces all the normal APPLY reports, assuming that any SYSMOD not already reported as having a problem will be successful during the real APPLY run. These reports can be used to determine these situations:

- Missing DD statements
- Missing requisite SYSMODs
- Regressions
- SYSMOD in hold exception status
- Processing of inline JCLIN

Building load modules

After selecting the elements to be installed, SMP/E builds new load modules and rebuilds existing load modules, as required. See Appendix C, "Building load modules," on page 547 for a description of the process used by SMP/E to build load modules.

Note: SMP/E checks whether load modules to be updated contain modules from a cross-zone with the same name as modules currently selected to update the load module. This is done by checking the load module's XZMOD subentries. If this condition exists, SYSMOD processing stops.

Element installation

Once the proper SYSMODs have been selected and the proper functional and service level of each element has been determined, SMP/E begins the process of calling utility programs to get the elements installed in the appropriate target libraries. The following sections describe the process of installing each of the element types supported by SMP/E.

Deleting elements

When the DELETE operand is specified on the element MCS, macros, modules, source, data elements, hierarchical file system elements, and program elements are deleted from a target library. When SMP/E processes an element specifying the DELETE operand in its MCS statement, it first saves the current element entry from the target zone in a BACKUP entry on the SMPSCDS. This BACKUP entry includes the SYSMOD ID of the SYSMOD containing the MCS statement that caused the change, plus an indicator for the type of change (DEL) made. After SMP/E saves the BACKUP entry for the element, it deletes the element from the target library and the target zone.

For load modules composed entirely of modules that are to be deleted, SMP/E deletes the load modules and any aliases for the load modules from the target libraries. SMP/E also deletes the LMOD entry from the target zone.

For load modules composed of modules to be deleted and modules not to be deleted, SMP/E delinks the CSECTs in the deleted modules from the load module. In priority order, SMP/E obtains the CSECT information in the following manner:

- From the list of CSECT names in the target zone MOD entry.
- From the CSECT operand on the ++MOD statement from the lowest function or service level SYSMOD being installed that contains that module.
- From the module name.

Any aliases associated with the load modules are not deleted from the target libraries, but might be marked nonexecutable by the link-edit utility. The MOD entries are deleted from the target zone, but the LMOD entry remains because it is still needed to process SYSMODs affecting the modules that have not been deleted.

Note: If a module is being deleted, SMP/E also checks whether the module is contained in any cross-zone load modules. If so, SMP/E deletes the contents of the MOD entry except the XZLMOD subentries. If the module is not reintroduced and the AUTOMATIC option is in effect for the cross-zone, it is deleted from the cross-zone load module during cross-zone processing. For more information, see *z/OS SMP/E Reference*.

Compressing the target libraries

You can use the COMPRESS operand of the APPLY command to have SMP/E compress the target libraries before installing SYSMODs. There are two methods of specifying which libraries are to be compressed:

- You can specify a list of specific libraries in the COMPRESS operand (including libraries that may not be affected by the APPLY command).
- You can specify ALL on the COMPRESS operand, in which case the only libraries eligible for compression are those in which elements will be installed by this APPLY command.

Note: Target libraries residing in a UNIX file system are not compressed.

Once the libraries have been determined, actual processing is dependent on the library type:

- For **source** libraries, any source to be replaced (not updated) by one of the SYSMODs being installed is deleted from the library.

- For **macro** libraries, no deleting is done. This is because the SYSMOD replacing the macro might fail, and there may be other SYSMODs that cause assemblies that use the macro.
- For **data element**, **hierarchical file system element**, and **program element** libraries, any data element, hierarchical file system element, or program element that is to be replaced by one of the SYSMODs being installed is deleted from the library.
- For **load** libraries, SMP/E determines which load modules within the library are composed only of modules copied during system generation and are being replaced by the SYSMODs being installed. Such load modules are deleted from the library. If a load module contains a module that is not being replaced, that load module is not deleted.

SMP/E then calls the copy utility to perform the actual compress operation.

Note:

1. To reclaim as much space as possible before installing the SYSMODs, members are deleted before the library is compressed. SMP/E processes one library at a time, first deleting members, then compressing the library.
2. When you install a function that deletes another function but supplies no elements, no libraries are compressed.

Macro replacements

One of the steps in actually updating the target libraries is to install macro replacements. Multiple SYSMODs can be applied, each of which can contain a replacement for the same macro. When two or more SYSMODs replacing the same macro are applied concurrently, SMP/E determines the version at the highest function and service level. For a full explanation of how SMP/E determines the functional and service level of an element, see “Element selection” on page 87.

SMP/E then schedules the actual update of the target macro library. The library to be updated is determined from the SYSLIB information in the target zone MAC entry. For further information about the initial setting of the MAC SYSLIB, see “Adding new elements other than modules to the target libraries” on page 67.

If there is no SYSLIB value in the MAC entry, SMP/E looks for a DLIB entry with the same name as the DISTLIB value in the MAC entry and uses the SYSLIB value from the DLIB entry.

Note: In this case, before you run the APPLY command, make sure there is only one SYSLIB subentry in the DLIB entry and that it specifies the ddname of the correct library.

If no SYSLIB is present, the SMPMTS is used as the target macro library. (For further information about the use of the SMPMTS as a target macro library, see “Use of the SMPMTS and SMPSTS as target libraries” on page 69.)

The actual update is done by calling either the copy utility or the update utility.

- The copy utility is used in these cases:
  - The macro was packaged in a relative file (the RELFILE operand was specified).
  - The macro was packaged in a text library (the TXLIB operand was specified) and had no alias names —that is, no MALIAS names are in the target zone MAC entry and no MALIAS operands are on the ++MAC statement.
  - The macro was packaged inline, it had no alias name, and the SSI operand was not specified.

  The SSI operand is ignored when TXLIB or RELFILE is specified.
- The update utility is called in these cases:
  - The macro was packaged in a text library, and the element had an alias name.
  - The macro was packaged inline, and either it had an alias name or the SSI operand was specified.

Upon return from either utility, SMP/E issues a message indicating whether the macro was replaced successfully.

Macro updates

After all the macro replacements have been done, any macro updates present can be scheduled. Again, multiple SYSMODs may contain updates for the same macro. When two or more updates to the same macro are being processed concurrently, SMP/E merges the text from each update 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 operands on the ++VER statements or by internal defaults, or both, as follows:

- If SMP/E finds a processing order relationship between all the SYSMODs being processed, the merge occurs according to that order.
- 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.
- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates for PTFs first, APAR fixes second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to perform the actual target library updating.

The library to be updated is determined from the SYSLIB information in the target zone MAC entry. For more information about how SMP/E determines the MAC SYSLIB, see “Macro replacements” on page 91. If no SYSLIB is present, the SMPMTS is used as the target macro library. For further information about the use of the SMPMTS as a target macro library, see “Use of the SMPMTS and SMPSTS as target libraries” on page 69. If there is no SYSLIB and the macro does not exist in the SMPMTS, the macro is obtained from the distribution library and then updated. Upon return from the update utility, SMP/E issues a message indicating whether the update was successful.

Note: SMP/E checks only whether the NAME operand on the ./ CHANGE statement specifies the same element as the ++MACUPD statement. (This checking is done during RECEIVE processing.) Other ./ CHANGE operands may produce undesired results. For example, if you code UPDATE=INPLACE and there is no SYSLIB in the MAC entry, the distribution library may be updated.

Source replacements

Another step in actually updating the target libraries is the installation of source replacements. Multiple SYSMODs can be applied, each of which can contain a replacement for the same source.

When two or more SYSMODs replacing the same source are applied concurrently, SMP/E determines the version at the highest function and service level. For full explanation of how SMP/E determines the functional and service level of an element, see “Element selection” on page 87.

SMP/E then schedules the actual update of the target source library. The library to be updated is determined from the SYSLIB information in the target zone SRC entry. For further information about the initial setting of the SRC SYSLIB, see “Adding new elements other than modules to the target libraries” on page 67.

If there is no SYSLIB value in the SRC entry, SMP/E looks for a DLIB entry with the same name as the DISTLIB value in the SRC entry and uses the SYSLIB value from that DLIB entry.

Note: In this case, before you run the APPLY command, make sure there is only one SYSLIB subentry in the DLIB entry and that it specifies the ddname of the correct target library.

If no SYSLIB is present, the SMPSTS is used as the target source library. For further information about the use of the SMPSTS as a target source library, see “Use of the SMPMTS and SMPSTS as target libraries” on page 69.

The actual update is done by calling either the update utility or the copy utility.

- The copy utility is used if the source replacement resides in either a text library (that is, the TXLIB operand was specified) or in a RELFILE (the RELFILE operand was specified). When TXLIB or RELFILE is specified, the SSI operand is ignored.
- The update utility is called if the source replacement was contained inline and the SSI operand was specified.

Upon return from either utility, SMP/E issues a message indicating whether the source has been replaced successfully.

Source updates

After all the source replacements have been done, any source updates present can be scheduled. Again, multiple SYSMODs can contain updates for the same source. When two or more updates to the same source are being processed concurrently, the text from each is merged. Looking at the sequence numbers in columns 73 to 80, SMP/E processes the updates in the order expressed by the PRE operands on the ++VER statements or by internal defaults or both as follows:

- If SMP/E finds a processing order relationship between all the SYSMODs being processed, the merge occurs according to that order.
- 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.
- If SMP/E cannot determine the processing order of the SYSMODs, it merges the updates for PTFs first, APARs second, and USERMODs third. Within each type, there is no specified order.

SMP/E then calls the update utility to do the actual updating of the target library. The library to be updated is identified from the SYSLIB information in the target zone SRC entry. For more information about how SMP/E determines the SRC SYSLIB, see "Source replacements" on page 92. If no SYSLIB is present, the SMPSTS is used as the target source library. For further information about the use of the SMPSTS as a target source library, see "Use of the SMPMTS and SMPSTS as target libraries" on page 69. If there is no SYSLIB and the source does not exist in the SMPSTS, the source is obtained from the distribution library and then updated. Upon return from the update utility SMP/E issues a message indicating whether the update was successful.

Note: SMP/E checks only whether the NAME operand on the ./ CHANGE statement specifies the same element as the ++SRCUPD statement. Other ./ CHANGE operands may produce undesired results. For example, if UPDATE=INPLACE is coded and there is no SYSLIB in the SRC entry, the distribution library may be updated.

Assemblies

This section describes these topics:

- Assembling source
- Assemblies caused by macros
- Reusing previous assemblies

Assembling source

When a SYSMOD contains source to be assembled, it can supply either just the source (++SRC/++SRCUPD) for that element, or it can supply both the source (++SRC/++SRCUPD) for the element and an object deck (++MOD) for the same element.

- **Source only:** When the SYSMOD supplies only the source and no corresponding object deck, the element is assembled.
- **Source and ++MOD:** When the SYSMOD supplies both the source and the corresponding object deck, SMP/E determines whether the object deck can simply be link-edited into the target system or whether the source must be assembled. It does so by considering the following questions:
  - Was ASSEM specified on the APPLY command?
  - Are there any updates to the source that the SYSMOD supplying the object does not know about (UMIDs in the target zone SRC entry)?
  - Has another SYSMOD, which this SYSMOD does not know about, assembled the module (RMID of the target zone MOD entry for the module to be assembled)?
  - Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of these questions is yes, the module is assembled if SMP/E can determine where the resultant assembled object should go. SMP/E knows where to put the assembled object if there is a target zone MOD entry for the resultant object and a load module into which the module should be link-edited. If the MOD entry cannot be found or is not included in the same SYSMOD, a warning message is issued. Processing of the SYSMOD continues without assembling or link-editing the module. For further information on how SMP/E processes the object deck after assembly, see "Module replacements" on page 95.

Assemblies caused by macros

A SYSMOD can supply macros that require the assembly of modules. The required assemblies are found as GENASM subentries in the target zone MAC entry and in the ASSEM and PREFIX operands on the ++MAC/++MACUPD statement. The source for these assemblies is found by looking for a target zone ASSEM entry matching the name of the module; if an ASSEM entry is not found, the SRC entry matching the name of the module is used as the source. If neither an ASSEM nor a SRC entry can be found, a warning message is issued, and SMP/E goes on processing the SYSMOD without assembling or link-editing the module.

The SYSMOD can also supply an object deck for the modules to be assembled. SMP/E determines whether to do the assembly rather than use the object decks supplied in the SYSMOD. It makes this determination by considering the following questions:

- Was ASSEM specified on the APPLY command?
- Are there any updates to the macro that the SYSMOD supplying the object does not know about (UMIDs in the target zone macro entry)?
- Has another SYSMOD, which this SYSMOD does not know about, assembled the module (RMID of the target zone MOD entry for the module to be assembled)?
- Is the ASSEMBLE indicator set for the corresponding MOD?

If the answer to any of these questions is yes, SMP/E assembles the module if it can determine where the resultant assembled object should go. SMP/E knows where to put the assembled object if there is a target zone MOD entry for the resultant object and a load module into which the module should be link-edited. (See "Module replacements" on page 95 for further information.)

Whenever a macro modification in a APAR or USERMOD type SYSMOD causes the assembly of a module, the ASSEMBLE indicator in the module's target zone MOD entry is set on. If any subsequent PTF, APAR, or USERMOD type SYSMODs contain modifications to macro or source elements affecting a module whose ASSEMBLE indicator has been set, they cause the module to be reassembled in spite of the presence of an object module in the SYSMOD. The reassembly prevents the assembled module from regressing during the installation of subsequent SYSMODs that might replace the affected module but that do not contain the macro modification introduced by the earlier SYSMOD.

To prevent future reassemblies of modules that have had their ASSEMBLE indicator set, the UCLIN function must be used to set it off (DEL).

Reusing previous assemblies

If SMP/E is run after a failure, assemblies are rerun to ensure that the proper source and macros are used. If the same set of SYSMODs is being processed after a failure, the assemblies run before the failure need not be rerun. The previously assembled objects for the failed SYSMOD are used if the REUSE operand is specified on the APPLY command.

Assembled object decks are stored on the SMPWRK3 data set. If SMPWRK3 is allocated with a final disposition of KEEP, these assembled modules are saved in SMPWRK3 until the SYSMODs causing the assemblies are successfully processed. This allows SMP/E to reuse the assembled object modules if the APPLY command fails. Once the SYSMODs have been successfully applied, SMP/E deletes the related entries from SMPWRK3 data set.

Note: SMP/E does not check to make sure the same set of SYSMODs are being rerun after a failure. The user must proceed with care in taking advantage of the REUSE facility.

Module replacements

The modules (++MODs and assemblies from ++SRCs) selected from a SYSMOD are generally link-edited to a load module library on the target system. SMP/E determines the load module to which a module belongs by checking for load module (LMOD) subentries in the target zone MOD entry for the module and by looking at the MODDEL subentries for all LMOD entries. The link-edit characteristics and control statements for the link-edit are found in the target zone LMOD entry for the appropriate load module. For details on information that is passed to the link-edit utility, see "Link-edit parameters and load module attributes" on page 95.

Note: If SMP/E cannot determine the load module for a module, it presumes that the configuration of the target system does not require the module, and does no link-edit or copy. However, it replaces the RMID subentry of the MOD entry with the ID of the SYSMOD supplying the ++MOD or causing the assembly.

Specific processing depends on whether the load module was part of a totally copied library, selectively copied, or defined by link-edit JCL. In addition, special processing is done for the nucleus load module (IEANUC01). For more information, see these sections:

- "Load modules in totally copied libraries" on page 95
- "Load modules that were selectively copied" on page 96
- "Load modules defined by Link-edit JCL" on page 96
- "MODDEL subentry processing for the nucleus load module (IEANUC01)" on page 97

Link-edit parameters and load module attributes

The parameters passed to the link-edit utility include the default link-edit **parameters** (LET, LIST, and XREF) and load module **attributes** (such as RENT, REUS, REFR, AMODE=24). SMP/E determines the attributes and parameters to be passed as follows:

- If SMP/E has determined that the binder is available on the system and SMP/E is processing CSECT deletes, the STORENX link-edit parameter is passed to the link-edit utility.
- If the load module was link-edited during initial installation, the link-edit parameters saved in the LMOD entry are used.
- If the load module was copied during initial installation (that is, the LMOD COPY indicator is on):
  - If LEPARM is specified on the ++MOD statement, the attributes supplied are used, and the corresponding target zone LMOD entry is updated (or created) with these attributes.
  - If LEPARM is not specified, SMP/E checks the following for link-edit attributes, in the order indicated, and passes the attributes that it finds:
    1. A target zone LMOD entry containing link-edit attributes
    2. An LKLIB data set or SMPTLIB data set containing the load module, if one is available from another SYSMOD that is in process
    3. The target library that is supposed to contain the load module

    If the load module or its link-edit attributes are not found in any of the places indicated, no load module attributes are passed to the link-edit utility unless the user has set some in the UTILITY entry for link-edit processing.
- If a link-edit UTILITY entry is in effect, the defaults are not passed to the link-edit utility. The values in the UTILITY entry are used instead.
- When the LMOD entry for the load module contains a CALLLIBS subentry, SMP/E uses CALL for the link to the actual target library or NCAL for link to the SMPLTS library, regardless of whether a UTILITY entry is in effect.

Load modules in totally copied libraries

SMP/E checks to determine whether the distribution library (DLIB) for the module has been totally copied to a target system library. (SMP/E makes this determination by looking for a target zone DLIB entry describing the module's DLIB.) If this is the case, SMP/E creates a load module on the target library

having the same name as the module's name, if one does not already exist. An INCLUDE statement is generated for the module, but no INCLUDE statement is generated for the current version of the load module.

Load modules that were selectively copied

If the LMOD entry for the load module contains the COPY indicator, SMP/E knows that the load module was created by copy JCL and that the load module contains only the one module. If the module is supplied on a LKLIB or relative file, SMP/E copies the module to the target system. If aliases are specified for such a module (as indicated by TALIAS values in the MOD entry), they must exist in the LKLIB or relative file in order to be copied. An INCLUDE statement is generated for the module, but no INCLUDE statement is generated for the current version of the load module.

Load modules defined by Link-edit JCL

When SMP/E links a module into a load module that was defined by link-edit JCL, the link-edit utility INCLUDE statements that are generated depend on whether the load module currently exists in a target library. (Processing is different when the load module is defined with a SYSLIB allocation. For details, see “Building load modules with a SYSLIB allocation” on page 96.)

- If the load module does not exist, an INCLUDE statement is built for each module defined in the link-edit JCLIN as being included in the load module. No INCLUDE statement is built for the load module itself.
- If the load module does exist, an INCLUDE statement is built for each module that is selected from a SYSMOD being processed and that is defined in the link-edit JCLIN as being included in the load module. In addition, an INCLUDE statement is built to include the current version of the load module from the target library. This is done to obtain the other modules that make up the load module.

Link-edit control statements saved in the target zone LMOD entry are passed to the link-edit utility as SYSLIN input.

Building load modules with a SYSLIB allocation

For each load module to be built, SMP/E determines whether a SYSLIB allocation is required. It does this by checking the corresponding LMOD entry for a CALLLIBS subentry list. If a SYSLIB allocation is required, the allocation is done before the load module is link-edited. Each ddname in the CALLLIBS subentry list is dynamically allocated using information from the corresponding DDDEF entry, and is assigned an SMP/E-generated ddname. When the CALLLIBS subentry list contains more than one name, SMP/E allocates them as a concatenation and gives it a generated ddname. If errors occur during the SYSLIB allocation, dynamic allocation error messages are issued, and the load module is not link-edited. Any SYSMODs supplying modules for the link-edit are also failed.

The procedure for building a load module with a SYSLIB allocation (one that has CALLLIBS subentries) depends on whether the load module also has XZMOD subentries and whether the set-to zone has an UPGLEVEL subentry, as follows:

- If the load module has **no** XZMOD subentries **and** the set-to zone **has** an UPGLEVEL subentry, the load module will be rebuilt from scratch and saved into its true system libraries. SMP/E will **not** save a "base" version of the load module in the SMPLTS data set.
- If the load module **has** XZMOD subentries **or** if the set-to zone has **no** UPGLEVEL subentry, then the load module is built in two stages: first, a "base" version of the load module is built in the SMPLTS data set; second, the executable version of the load module is built in the target libraries.

  1. **Building the "base" version of the load module.** The appropriate INCLUDE statements are built:
     - If the load module does **not** currently exist in the SMPLTS library, an INCLUDE statement is built for each module explicitly defined in the link-edit JCLIN as being included in the load module. No INCLUDE statement is built for the load module itself.

       Note: This is done even if the executable version of the load module exists in the target libraries.
     - If the load module **does** currently exist in the SMPLTS data set, an INCLUDE statement is built for each module that is selected from a SYSMOD being processed and that is explicitly defined in the

link-edit JCLIN as being included in the load module. In addition, an INCLUDE statement is built to include the current version of the load module from the SMPLTS library. This is done to obtain the other modules that make up the load module.

When the "base" version of a load module is link-edited into the SMPLTS, any CHANGE and REPLACE link-edit control statements defined for the load module are passed to the link-edit utility, as well as all link-edit options defined for the load module. (No link-edit control statements other than CHANGE and REPLACE are processed.) This link-edit results in unresolved external references, which is considered normal.

2. **Building the executable version of the load module.** The executable version of the load module is built in the target libraries using the load module's SYSLIB allocation (the CALLLIBS subentry list in its LMOD entry) and the "base" version of the load module from the SMPLTS data set. The only INCLUDE statement built is for the "base" version of the load module from the SMPLTS data set.

   Note:

   a. If the "base" version of the load module does not exist in the SMPLTS data set, the load module is not link-edited.

   b. A load module can reside in an executable target library before a base version of it has been built in the SMPLTS. If the load module had included cross-zone modules through the use of the LINK MODULE command, these modules are no longer included after the installation of a SYSMOD that causes the load module to be built into the SMPLTS. (Warning messages are issued to indicate this.) After the installation of such a SYSMOD, the LINK MODULE command needs to be rerun in order to include those cross-zone modules back into the load module.

Multitasking of link-edit utility invocations

When multiple output libraries must be updated for link-edit processing, SMP/E may initiate a separate subtask for each such library, provided that:

- The maximum number of such subtasks (10 subtasks) has not been reached.
- The link-edit utility being used is reentrant (the Binder is reentrant but the old linkage editor is not)
- The logical SYSPRINT file to be used for link-edit operations has either a DDDEF that specifies a SYSOUT class or an entry in the SMPPARM GIMDDALC member that specifies a SYSOUT class. The logical SYSPRINT is specified by the PRINT subentry for the link-edit UTILITY entry in effect. The default is SYSPRINT.

This multitasking of link-edit steps should result in a better elapsed time for the SMP/E process when many link-edits must be done to process a set of SYSMODs.

You can tell if multitasking of link-edit operations is occurring by looking at the link-edit completion messages being issued by SMP/E. If the message ends with "-- SYSPRINT FILE xxxxxxxx.", which indicates the ddname used for the link-edit output, then multitasking is being done.

Note: If message filtering is turned on (MSGFILTER=YES in the active OPTIONS entry), link-edit completion messages might only be written to SMPLOG but not to SMPOUT.

SMP/E ensures that libraries are processed in the right order for CALLLIBS consideration, even when multitasking of link-edit operations is in effect.

MODDEL subentry processing for the nucleus load module (IEANUC01)

SMP/E recognizes IEANUC01 as a special load module and, therefore, unlike for other load modules, MODDEL subentries for IEANUC01 do not cause SMP/E to automatically link modules back into the nucleus. Although a MODDEL subentry is created for IEANUC01 when a module is deleted from the nucleus, this subentry is ignored when the module is reintroduced.

Module updates

For each ++ZAP element within a SYSMOD, SMP/E looks for all the load modules to which the distribution module was either linked or copied (similar to the processing done for ++MODs). SMP/E then attempts to install the superzap to each of these load modules. If the load module being zapped contains a CALLLIBS

subentry, the SMPLTS version of the load module is also zapped, if the load module also contains XZMOD subentries, or if the set-to zone does not have an UPGLEVEL subentry.

In applying the ZAP, SMP/E performs two passes: the first to process the VER control cards, and the second to process the REP control cards. The REP pass is performed only if the VER pass has been completed successfully for all load modules to be changed.

Data element and program element replacements

Still another step in updating the target libraries is to install replacements for data elements and program elements. Multiple SYSMODs can be applied, each of which can contain a replacement for the same element. When two or more SYSMODs replacing the same element are applied concurrently, SMP/E determines the version at the highest function and service level. For a full explanation of how SMP/E does this, see "Element selection" on page 87.

SMP/E then schedules the actual update of the target library. The library to be updated is identified from the SYSLIB information in the target zone element entry.

If there is no SYSLIB value in the element entry, SMP/E looks for a DLIB entry with the same name as the DISTLIB value in the element entry and uses the SYSLIB value from the DLIB entry.

Note: In this case, before you run the APPLY command, make sure the DLIB entry contains only one SYSLIB subentry specifying the ddname of the correct target library.

Replacing data elements

Note: For a list of data element types, refer to the "Data Element MCS" section in *z/OS SMP/E Reference*.

The actual update to the library is done by either SMP/E or the copy utility.

- SMP/E updates the library in these cases:
  - The data element was packaged inline and has been transformed by GIMDTS. All control information is removed, the transformed data element is changed back to its original format, and the target library is updated with the element.
  - The data element must be reformatted to be compatible with the target library. For more information on reformatting data elements, see "Reformatting data elements" on page 98.
  - The target library is a sequential data set.
- The copy utility updates the library in all other cases. A COPY control statement is passed to the copy utility.

Reformatting data elements

During ACCEPT, APPLY, and RESTORE processing, SMP/E must sometimes reformat a data element to be make it compatible with the library into which it is to be installed. This can happen when the data element is being installed from:

- a variable-length record format to a fixed-length record format or from a fixed-length record format to a variable-length record format.
- a fixed-length record format to a fixed-length record format and the input and output record lengths are not equal.
- a variable-length record format to a variable-length record format and the input record length is greater than the output record length.

SMP/E uses these rules when reformatting:

- If the input record length is greater than the output record length, and SMP/E can truncate the record to the correct length by removing only trailing blanks, then SMP/E will reformat the record. Otherwise, SMP/E will issue an error message and terminate the reformatting operation.
- If the input record length is less than the output record length and the output record format is fixed, the record will be padded with blanks.

- Sequence numbers (if present) will be preserved.

Table 5 on page 99 describes the combinations of the supported record formats, appropriate actions, and identifies when reformatting or copying is necessary.

Table 5. Installation actions for input and output data sets

Input format	Output format	Condition	Action
F,FB	F/FB	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	FS/FBS	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	Fx/FBx	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	V/VB	N/A	Reformat
	Vx,VBx	N/A	Reformat
	All others	N/A	Error
FS,FBS	FS/FBS	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	F/FB	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	Fx/FBx	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	V/VB	N/A	Reformat
	Vx/VBx	N/A	Reformat
	All others	N/A	Error
Fx,FBx	Fx/FBx	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	F/FB	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	FS/FBS	Output LRECL = Input	Copy
		Output LRECL ¬= Input	Reformat
	V/VB	N/A	Reformat
	Vx/VBx	N/A	Reformat
	Vy/VBy	N/A	Error
	Fy/FBy	N/A	Error
	All others	N/A	Error
V/VB	V/VB	Output LRECL >= Input	Copy
		Output LRECL < Input	Reformat
	Vx/VBx	Output LRECL >= Input	Copy
		Output LRECL < Input	Reformat
	F/FB	N/A	Reformat
	FS/FBS	N/A	Reformat
	Fx/FBx	N/A	Reformat
	All others	N/A	Error

Table 5. Installation actions for input and output data sets (continued)

Input format	Output format	Condition	Action
Vx/VBx	Vx/VBx	Output LRECL >= Input	Copy
		Output LRECL < Input	Reformat
	V/VB	Output LRECL >= Input	Copy
		Output LRECL < Input	Reformat
	F/FB	N/A	Reformat
	FS/FBS	N/A	Reformat
	Fx/FBx	N/A	Reformat
	Fy/FBy	N/A	Error
	Vy/VBy	N/A	Error
	All others	N/A	Error
VS	VS	Output LRECL >= Input	Copy
		Output LRECL < Input	Error
	All others	N/A	Error
VBS	VBS	Output BLKSIZE = Input	Copy
		Output BLKSIZE ¬= Input	Error
	All others	N/A	Error
any other format	any format	N/A	Error

Legend:

Data Set Format Descriptors

x
control character; A (ANSI) or M (machine)

y
control character; can be A (ANSI) or M (machine), but not the same as x

Actions

Copy
If the input data element is a member of a PDS (or PDSE) and the output element is a member of a PDS (or PDSE), SMP/E uses the copy utility to install the element. If output is a sequential data set, SMP/E installs the element itself.

Reformat
SMP/E installs the element with reformatting.

Error
Incompatible input/output RECFM combination. Installation fails.

At the end of processing, SMP/E issues a message indicating whether the data element has been replaced successfully.

Replacing program elements

The actual update to the library is done by the copy utility. A COPYMOD control statement is passed to the copy utility.

If the program element was packaged in a relative file (the RELFILE operand was specified), or in a link library (the LKLIB operand was specified), then the copy utility is used to perform the update directly.

If a program element is packaged inline, SMP/E first retransforms the program element into its original VS or VBS format in a temporary data set. Then, if the target library and the data set that contained

the original program element are of different types (that is, one is a PDS and the other a PDSE), SMP/E allocates a temporary SMPTLOAD data set of the same type as the data set that contained the original program element and uses the copy utility to reload a program element and its aliases to the SMPTLOAD data set. SMP/E then uses the copy utility to place the program element and its aliases into the target library. The input data set for the copy operation is the SMPTLOAD data set, if one was required, or the retransformed temporary data set, if an SMPTLOAD data set was not required.

At the end of processing, SMP/E issues a message indicating whether the program element has been replaced successfully.

Hierarchical file system element and Java archive file replacements

Hierarchical file system element and Java Archive file replacements are processed in exactly the same way. Before any processing occurs, SMP/E checks to make sure that the hierarchical file system (HFS) copy utility is available (BPXCOPY is the default). If it is not, SMP/E continues APPLY processing. If it is later discovered that a hierarchical file system element or Java Archive file needs to be processed from a selected SYSMOD, SMP/E terminates that SYSMOD.

When the HFS copy utility is available, a hierarchical file system element or Java Archive file is processed by that utility. Before invoking the HFS copy utility, SMP/E deletes the current symbolic links of the element, and checks the linknames of the current element against those of the replacement. If any of the existing linknames are different, SMP/E deletes only those that will not be replaced. SMP/E then calls the HFS copy utility to replace the element. The HFS copy utility will copy the element into its target directory, replace current linknames, and create new linknames and all symbolic links.

If the hierarchical file system element or Java Archive file identifies a shell script (on the SHSCRIPT subentry of the element's MCS statement), SMP/E invokes the shell script to perform installation-related activities on behalf of the element. If the identified SHELLSCR element is also being replaced, to ensure the latest version of the SHELLSCR element is used, SMP/E will copy the SHELLSCR element to its target directory before invoking the shell script. Depending on the options chosen on the SHSCRIPT subentry, SMP/E invokes the shell script before or after copying the element to a UNIX file system. Shell scripts, which are themselves hierarchical file system elements, are usually provided by the product packager.

The hierarchical file system element or Java Archive file can be packaged in relative file format, text library format, or inline. If the element was packaged inline after being transformed, SMP/E retransforms the element back to its original format before invoking the HFS copy utility.

UNIX file permissions

Before manipulating files in the UNIX file system, SMP/E temporarily switches the user ID that is executing the SMP/E task to superuser authority (UID=0) and restores the user ID to the previous level of authority when the SMP/E updates are complete. This includes any actions performed on SMP/E's behalf by the HFSCOPY and link edit utilities.

This means SMP/E users are not required to have UID=0 (superuser) authority all the time, which reduces the chance of such users accidentally erasing or damaging files in a UNIX file system while performing non-SMP/E work. However, the user ID that executes an SMP/E task must be defined to the BPX.SUPERUSER facility class profile for this process to work properly.

HFS copy utility parameters and alternate ddnames

The parameters and alternate ddnames passed to the HFS copy utility include these values:

- The parameters include any PARM values specified in the HFSCOPY UTILITY entry, plus parameters generated by SMP/E based on either the contents of the hierarchical file system element or Java Archive file element entry (if one exists) or on the MCS for the element.

  The SMP/E-generated parameters include the name of the element, its installation format (TEXT or BINARY), an identifier for the resulting utility output, and any linknames associated with the element.

Note: The maximum total length of the parameters to be passed is X'FFFF' bytes. If the length exceeds this number, SMP/E truncates the parameters at the limit and passes this value to the HFS copy utility.

- The alternate ddnames include any PRINT values specified in the HFSCOPY UTILITY entry, the ddname of the input data set to be used by the utility, and the ddname of the output data set to be used by the utility (the SYSLIB ddname for the element).

Java archive file updates

One or more ++JARUPDs may be processed for a single Java Archive (JAR) file, and the ++JARUPDs for that JAR file will be processed in the order implied by the processing order of the SYSMODs which contain the ++JARUPDs. That is, if a SYSMOD contains a ++JARUPD and specifies a prerequisite for another SYSMOD that contains a ++JARUPD for the same JAR file, then the ++JARUPD from the prerequisite SYSMOD is processed first. If no processing order can be implied from the containing SYSMODs, then no particular order of processing for the ++JARUPDs will be assumed.

Following are the steps required to process one or more ++JARUPDs for a single JAR file:

1. Create the primary temporary work directory in the SMPWKDIR directory for processing all ++JAR and ++JARUPD elements. If the SMPWKDIR directory is not specified on a DD statement or DDDEF entry, SMP/E will use the /tmp directory.
2. Copy the JAR file to be updated into the primary temporary work directory.
3. Create an update subdirectory for processing the ++JARUPD elements for this JAR file.
4. Copy a ++JARUPD element to the primary temporary work directory.
5. Extract the component files from the ++JARUPD element into the update subdirectory.
6. Repeat steps 4 through 5 for each ++JARUPD to be processed for this JAR file.
7. Update the JAR file with the component files from all the ++JARUPD elements.
8. Cleanup the ++JARUPD elements and all component files in the update subdirectory.
9. Copy the updated JAR file into its target directory.

Recording after completion

Results of processing are recorded in the following entries.

Target zone element and LMOD entries

APPLY processing creates, modifies, and may delete target zone element entries.

- Entry update indicator: When an entry is added by a SYSMOD being processed or modified by inline JCLIN, the SYSMOD's SYSMOD ID is placed in the LASTUPD subentry of the target zone element entry.
- ALIAS subentries: The updates to an element's ALIAS subentries are discussed under “Alias processing” on page 70.
- LMOD subentries: When the LMOD operand is specified on a ++MOD statement, the values in the operand list are added to the target zone MOD entry as LMOD subentries.
- MODID subentries:
  - The FMID subentry is replaced with the FMID of the SYSMOD from which the modification to the element was selected. If the SYSMOD is a function SYSMOD, the FMID is set to the SYSMOD ID of the function itself.
  - The RMID subentry is changed when a replacement element or assembly is applied. The RMID is set to 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 is set for the module, even if the actual assembly was suppressed because the SYSMOD supplied an assembled version of the module. (See “Source replacements” on page 92 and “Assemblies” on page 93 for further information.)

If the replacement element's MCS specified an RMID for the element (the RMID operand), the specified value is used.

- When a replacement element is applied, all UMID subentries are deleted.

  If the replacement element's MCS specified a list of UMIDs for the element (the UMID operand), these UMIDs replace any existing UMIDs for the element.

- UMID subentries are added when updates for the element are applied. The UMIDs are the 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, the UMID subentry for the superseded SYSMOD is deleted from the element entry.

- SYSLIB subentry

  If a DLIB entry was used to determine the SYSLIB value identifying the target library for an element or load module, that SYSLIB value is added to the corresponding element entry or LMOD entry.

- CSECT names

  The CSECT information from the ++MOD statement is saved in the target zone MOD entry. The information saved is determined in the following way:

  - If the SYSMOD that the selected version of the module came from contained the CSECT operand, the CSECT names present there are either added to the target zone MOD entry (if no CSECT information was already there) or are used to replace the existing list of CSECT names in the target zone MOD entry.
  - If the SYSMOD that the selected version of the module came from did not contain the CSECT operand, SMP/E checks to see if any other SYSMOD applicable to the same FMID was also being applied. If so, and if any of those SYSMODs contained the CSECT operand, the CSECT information from the SYSMOD at the highest service level is used.

SMPSCDS BACKUP entries

BACKUP entries are created on the SMPSCDS data set associated with the target zone, so RESTORE processing can recover modifications to target zone entries if a SYSMOD is restored.

Note: No BACKUP entries are created when a load module is deleted by the ++DELETE statement.

Target zone SYSMOD entries

For each SYSMOD processed, a SYSMOD entry is created in the target zone. 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, subentries for each of the elements included in the SYSMOD package, and indicators that are set when ++IF and ++JCLIN are present.

A SYSMOD is considered successfully processed when all its selected elements have been applied to the appropriate system libraries **and** all its requisites have been successfully processed. Because SMP/E processes any number of SYSMODs with elements in common, it is possible that some SYSMODs have elements that need not be installed in a target 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 one SYSMOD is superseded by another, SMP/E makes a record of this by adding the name of the superseding SYSMOD to the entry for the superseded SYSMOD.

- When there is only one superseding SYSMOD, its name is saved in the LASTSUP subentry.
- When there are several superseding SYSMODs, the name of the last one is saved in the LASTSUP subentry, and a complete list is saved in the SUPBY subentry.

If the superseded SYSMOD has not been previously applied, its target zone SYSMOD entry contains only the LASTSUP information. Such a SYSMOD entry is called a "dummy entry". Because the superseded SYSMOD had not been previously applied, SMP/E does not know what type of SYSMOD it was (function, PTF, APAR, or USERMOD).

Deleted SYSMODs

When one SYSMOD is deleted by another, SMP/E makes a record of this by adding the name of the deleting SYSMOD to the DELBY subentry of the deleted SYSMOD. If the deleted SYSMOD has not been previously applied, its target zone SYSMOD entry contains only the DELBY information. Such a SYSMOD entry is called a "dummy entry". Although the deleted SYSMOD had not been previously applied, SMP/E assumes that it was a function SYSMOD, because only function SYSMODs can be explicitly deleted.

Conditional requisite data

For each SYSMOD named as an FMID in a ++IF statement, a SYSMOD entry is created with CIFREQ subentries representing the conditional requisite requirements.

If the SYSMOD existed previously, the CIFREQ data is simply added to the existing entry; otherwise, a new SYSMOD entry is created to save the CIFREQ data for use if the FMID is installed later.

Regressed element subentries

The MODID verification checks described earlier may leave elements open to regression. When potential regression of an element is detected, a record for the SYSMOD that previously modified the element is kept by marking the element subentries in the SYSMOD entry as regressed.

Global zone SYSMOD entries

The target zone name is added as an APPID subentry in the global zone SYSMOD entry for each successfully processed SYSMOD. The APPID subentries in the global zone, therefore, reflect the target libraries to which each SYSMOD has been applied.

Note: The global zone SYSMOD entry is deleted when the SYSMOD is rejected.

Zone and data set sharing considerations

The following identifies the phases of APPLY processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

2. SYSMOD selection

   Cross-zones
   Read with shared enqueue.

   Global zone
   Read with shared enqueue.

   Target zone
   Update with exclusive enqueue.

   SMPPTS
   Read with shared enqueue.

3. Element selection

SMPPTS
Read with shared enqueue.

Target zone
Update with exclusive enqueue.

4. Load module build

DLIB zone
Read with shared enqueue.

Global zone
Read with no enqueue.

Target zone
Update with exclusive enqueue.

SMPPTS
Read with shared enqueue.

5. Utility calling

Target zone
Update with exclusive enqueue.

6. Cross-zone requisite reporting phase

Cross-zones
Read with shared enqueue.

Global zone
Read with shared enqueue.

Target zones
Update with exclusive enqueue.

7. Global zone update

Global zone
Update with exclusive enqueue.

Target zone
Update with exclusive enqueue.

SMPPTS
Update with exclusive enqueue.

8. Cross-zone processing

Cross-zones
Read with shared enqueue.

Cross-zones
Update with exclusive enqueue.

Global zone
Read with no enqueue.

9. Termination

All resources are freed.

Chapter 4. The BUILDMCS command

The BUILDMCS command provides a more automated and reliable method for copying products from one pair of target and distribution zones and their libraries into another pair of target and distribution zones and their associated libraries. The BUILDMCS command creates MCS and JCLIN needed as input to RECEIVE, APPLY, and ACCEPT processing for reinstallation of products in another SMP/E environment. Reinstallation allows for the requisite checking needed to ensure the environment into which the product is being installed is appropriate. The output of the BUILDMCS command is a superseding function SYSMOD for each base function specified and a superseding function SYSMOD for any dependent functions related to an FMID specified on the BUILDMCS FORFMID operand. These superseding functions include all maintenance and user modifications that have been installed in the zone specified for the BUILDMCS command.

Note: The BUILDMCS command does not create MCS for any FEATURE or PRODUCT data that may be associated with an FMID. If you wish to copy the FEATURE and PRODUCT data for an FMID, you can use the GZONEMERGE FORFMID command to merge all FEATURE and PRODUCT entries for the desired FMIDs from the source global zone to the target global zone. You must then delete any unwanted FEATURE, PRODUCT, SYSMOD, and HOLDDATA entries that were created by GZONEMERGE.

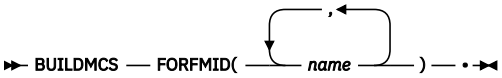
Zones for SET BOUNDARY

For the BUILDMCS command, the SET BOUNDARY command must specify either a target zone or distribution zone associated with the distribution libraries containing the elements for the FMIDs specified.

Note: For the BUILDMCS command, the SET BOUNDARY command should specify a target zone only if the target zone and its related distribution zone are at the same service level. If the target and distribution zone are not at the same service level, then the SET BOUNDARY command should specify the distribution zone.

Syntax

BUILDMCS Command



Operands

FORFMID

specifies the names of the FMIDs or FMIDSETs for which the MCS and JCLIN are to be created. This is a required operand.

The specified FMIDs (including the FMIDs obtained from the FMIDSET values) must be valid for the BUILDMCS command. An FMID is valid only if all of the following are true:

- the FMID exists in the zone specified on the SET command
- the FMID is a function
- the FMID is not deleted by another FMID
- the FMID is not superseded by another FMID
- the FMID is not in error

No MCS nor JCLIN is created for an invalid FMID. If all of the FMIDs are invalid, then no MCS nor JCLIN is created at all.

Data sets used

These data sets might be needed to run the BUILDMCS command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	*zone*
SMPLOG			

Note:

1. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
2. In addition to the data sets listed here, the BUILDMCS command also uses the DDDEFs for the distribution libraries, if they are present in the set-to zone. While the BUILDMCS command will function if these DDDEFs are not present in the set-to zone, the resulting output must be edited to add the information that would otherwise have been extracted from the DDDEF entries. IBM therefore recommends that you ensure that these DDDEFs are in the set-to zone before you use the BUILDMCS command.

Usage notes

Product intersections

The BUILDMCS command provides facilities to help copy a product from one SMP/E environment and install it into another. The BUILDMCS command is not intended to be used with all products. It is intended to be used for products that have no intersections with other products. These intersections come in two forms:

1. **Shared Load Modules**

   A shared load module is any load module that contains modules from more than one product. If the product to be copied supplies modules that reside in load modules along with modules from other products, then that product has shared load modules.

2. **Common Elements**

   A common element is any element with the same name and type that is supplied by more than one product. One product may take ownership of the element from another product using the VERSION operand.

If a product has either of these types of intersections with another product, the BUILDMCS command output for that product might be incorrect, because SMP/E does not have enough information in the zone entries to correctly create a corresponding superseding function SYSMOD.

Other considerations

Maintenance level
If running the BUILDMCS command from a target zone, you must ensure that the target and distribution zones are at the same maintenance level before issuing the command. This will ensure that the resulting MCS will be correct.

When deciding whether the BUILDMCS command is the appropriate method for copying a particular product, you should also consider these situations:

Element Versioning
If the original ++FUNCTION or any PTFs for the product to be copied supplied elements using the VERSION operand on the element MCS, the VERSION operand will not be included on the element MCS created by the BUILDMCS command. The version information is not saved in the zone entries during APPLY or ACCEPT processing and is therefore not available to the BUILDMCS command.

Macros causing assemblies
If the original ++FUNCTION or any PTFs for the product to be copied used the ASSEM or PREFIX operands on the ++MAC MCS to supply macro instructions, these ASSEM or PREFIX operands will not be included on the ++MAC MCS created by the BUILDMCS command. This information is not saved in the zone entries during APPLY or ACCEPT processing and is therefore not available to the BUILDMCS command.

Move, Rename, and Delete MCS
If the original ++FUNCTION or any PTFs for the product to be copied supplied ++MOVE, ++RENAME, or ++DELETE MCS, these MCS will not be included in the MCS created by the BUILDMCS command.

Load module definition
It is possible for more than one product to supply modules that reside in a single load module. SMP/E cannot determine from the zone entry information which product supplied the JCLIN link edit step to define the load module. Therefore, it is possible the BUILDMCS command will create a JCLIN link edit step to define a load module, even though the product to be copied did not originally supply the JCLIN or define the load module.

Target zones and LEPARM and DALIAS information
Target zones do not contain the LEPARM and DALIAS information for MOD entries. Therefore, when the BUILDMCS command is issued against a target zone, the generated ++MOD MCS will not contain the LEPARM and DALIAS information.

Target zone and distribution zone service levels
Ensure that the target and distribution zones into which SMP/E is currently installed are at the same service level. This can be determined by running either REPORT SYSMODS or LIST NOACCEPT/LIST NOAPPLY. If any differences are listed in the output, either ACCEPT or RESTORE the differences.

Output

Output from the BUILDMCS command includes reports, as well as output to SMPPUNCH.

Reports

The following reports are produced by the BUILDMCS command:

- BUILDMCS Entry Summary Report
- BUILDMCS Function Summary Report
- Dynamic Allocation Report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

SMPPUNCH output

The BUILDMCS command creates complete superseding function SYSMODs. The SYSMODs are written to the SMPPUNCH data set.

Example

Suppose that you want to propagate an FMID, such as SMP/E V3R6, to another system. You could do this with the BUILDMCS command by following these steps:

1. Ensure that the target and distribution zones into which SMP/E is currently installed are at the same service level. This can be determined by running either REPORT SYSMODS or LIST NOACCEPT/LIST NOAPPLY. If any differences are listed in the output, either ACCEPT or RESTORE the differences.

Note: If you ACCEPT the differences, you are raising the service level in the distribution zone. If you RESTORE the differences, you are lowering the service level in the target zone.

2. Use the distribution zone as the zone identified on the SET command. However, if you do not ACCEPT JCLIN (ACCJCLIN is not specified in the DLIBZONE entry), then you should use the target zone as the zone against which the BUILDMCS command is to be run. SMP/E needs the JCLIN to create entries such as load modules. The JCLIN exists in the distribution zone only if ACCJCLIN is specified in the DLIBZONE entry.
3. Run BUILDMCS against the zone identified in the previous step, as shown in this example:

```
SET BDY(FROMDLB)             /* DLIB zone where HMP1E00 is installed */.
BUILDMCS                     /* Build the MCS and JCLIN              */
FORFMID(HMP1E00)             /* for the FMID HMP1E00                 */.
```

4. Use the DDDEF entry information from the BUILDMCS Entry Summary Report to determine which DDDEFs must be defined in the new target and distribution zones and define them.
5. RECEIVE, APPLY, and ACCEPT the output in the SMPPUNCH data set that was generated by the BUILDMCS command.

Processing

FMID applicability

SMP/E first determines whether each specified FMID is valid, as previously defined under the description of the FORFMID operand. Once an FMID has been determined to be valid, SMP/E begins to collect the information needed to build the ++FUNCTION and ++VER statements for the FMID. For the ++FUNCTION MCS, SMP/E sets the SYSMOD ID to the specified FMID and REWORK operand to the current date. SMP/E creates the FESN operand only if the FESN subentry is specified in the SYSMOD entry.

SMP/E also begins building one ++VER statement for the entire FMID. The ++VER MCS information stored in the FMID's SYSMOD entry in the set-to zone is saved. However, all the SYSMODs associated with this FMID must first be determined before the entire ++VER statement can be built.

SMP/E then determines:

- All SRELs supported by the set-to zone
- All SYSMODs originally listed to be deleted upon installation of this FMID
- If this FMID is a dependent function, the FMID of the base function
- All functions originally listed that cannot exist in the same zone as this function
- All SYSMODs originally listed as prerequisites for this SYSMOD
- All SYSMODs originally listed to be superseded by this SYSMOD
- All functions previously listed on the VERSION operand of the specified function

Determine SYSMODs associated with FMIDs

For each FMID that is specified on the BUILDMCS command, SMP/E determines what other SYSMODs exist in the set-to zone that are associated with the FMID. An associated SYSMOD is one that has an FMID on the ++VER statement that matches a valid FMID specified on the BUILDMCS command. Each SYSMOD that is defined in the set-to zone is examined to determine if it is an associated SYSMOD. If it is an associated SYSMOD, then information from the SYSMOD entry is used to build the ++VER MCS statement.

The ++FUNCTION MCS that is being created is a superseding function. This means that this function completely includes and replaces all non-FUNCTION SYSMODs that are determined to be associated to it. Therefore, the ++VER MCS that is generated for this superseding function includes all the associated SYSMODs in the SUP operand. Also, all information supplied in the ++VER MCS statement specified for the associated SYSMOD is brought forward into this superseding function MCS. As an example, if ++PTF UR11111 has a SUP for APAR AR11111 in its ++VER MCS, then the ++VER MCS for the superseding

function contains both UR11111 and AR11111 on the SUP operand. If a SYSMOD is identified as associated to an FMID and has a status of ERROR, that PTF is still brought forward and included in the superseding function.

Once all associated SYSMODs are identified, the ++IF statements are created. All SYSMOD CIFREQ subentries in the set-to zone are examined. A single ++IF MCS is created for each SYSMOD CIFREQ subentry that was caused by the specified function or any of the associated SYSMODs to be superseded by the specified function. The CIFREQ subentries appear within the SYSMOD entry for a function SYSMOD named on the FMID operand of a ++IF. The subentry contains the required SYSMOD and the causer SYSMOD. The causer is the SYSMOD that originally supplied the ++IF MCS.

Determine elements for all associated SYSMODs

Once SMP/E has identified all of the SYSMODs associated with a function SYSMOD, it then identifies all of the elements associated with the specified function and other SYSMODs. The SYSMOD entries in target and distribution zones contain subentries for each element MCS supplied by that SYSMOD. SMP/E uses these subentries to identify the elements associated with the function and its associated SYSMODs. These associated elements are described by element MCS in the superseding ++FUNCTION that is being built by the BUILDMCS command.

SMP/E checks the zone entry for each SYSMOD, reads the subentries, and identifies candidate entries in the zone. For each candidate element, SMP/E then checks the element's zone entry. If the element entry exists in the zone, and its FMID matches the ++FUNCTION being built, SMP/E builds an MCS for the element specifying the information from the element's zone entry.

Note: SMP/E uses the MACUPD, SRCUPD, SZAP, and XZAP subentries only to identify candidate MAC, SRC, and MOD entries. SMP/E does not create any update MCS in the superseding ++FUNCTION.

SMP/E creates a FROMDS operand on the element MCS to specify the data set name associated with the element's DISTLIB value. SMP/E extracts the data set name from the DDDEF entry for the DISTLIB ddname. If a DDDEF entry is not found in the set-to zone for the DISTLIB ddname, then only the ddname is specified on the FROMDS operand. Processing will continue, but the superseding ++FUNCTION created will be incorrect, because the FROMDS operand does not specify the correct data set name. The report entry for this DDDEF in the BUILDMCS Entry Summary Report will indicate the needed entry was not found. If the FMID value found in the element's entry does not match the superseding function SYSMOD being built, the element was either not selected from the candidate SYSMODs when it was installed, or the FMID was subsequently changed by another SYSMOD using the VERSION operand or by the UCLIN command.

In either case, the MCS for the element will not be built because a different FMID now owns the element. If an element entry does not exist in the zone, then SMP/E will assume the element has been deleted. If a SYSMOD supplied an MCS with the DELETE operand to delete an element, the element entry will not exist in the set-to zone, and the SYSMOD entry in the set-to zone contains no indication of the deletion. An element subentry exists in the SYSMOD entry and cannot be distinguished from a replacement element subentry. Therefore, since SMP/E assumes the element has been deleted, an element MCS will not be built.

For module type elements, if a candidate module has cross-zone (XZLMOD) subentries, SMP/E does not carry forward any information in the MCS or JCLIN regarding the cross-zone load modules.

Determine LMODs for module elements

After identifying all of the associated module elements, SMP/E then determines the load modules associated with the identified modules. For each candidate module element, its zone entry is examined to obtain the module's LMOD subentries. Each load module named in the LMOD subentries is a candidate load module.

If the module's distribution library has been totally copied to a target library, then a load module with the same name as the module is also a candidate (a distribution library has been totally copied if there is a DLIB entry in the zone describing that distribution library).

If a module has no LMOD subentries and the module's distribution library is not a totally copied library, then SMP/E cannot determine the load modules into which the module should be copied or link edited. This means when the function SYSMOD is applied, SMP/E cannot determine into which target libraries this module should be installed.

SMP/E looks for the zone entry for each load module in the list of candidates. If the entry is found, the information from the entry is used when building the JCLIN to define the load module.

If a load module's LMOD entry contains a MODDEL subentry list, SMP/E cannot carry forward any information in the MCS or JCLIN about modules that were once part of this load module, but have since been deleted. This is because there is no information about the deleted module, other than its name, to carry forward.

If a load module contains cross-zone (XZMOD) subentries, SMP/E does not carry forward any information in the MCS or JCLIN regarding the cross-zone modules.

If the LMOD entry is not found, and the load module is part of a totally copied distribution library, then the target library information from the DLIB entry is saved for use when building the JCLIN copy steps to define the load module. If the LMOD entry is not found and the load module is not part of a totally copied distribution library, processing will continue, but the superseding ++FUNCTION created will be incorrect, because this load module will not be defined and some modules will not be installed in any target library.

Create JCLIN

Once all of the elements and load modules have been identified as previously described, JCLIN is built to define these entries:

- load modules, both copied and link edited
- in-line assembler source (ASSEM entries)
- totally copied distribution libraries (DLIB entries)

The intent is not to completely reproduce the JCLIN originally supplied by the subject FMID, but only to produce the JCLIN necessary to define those entries. For example, COPY steps will not be created for macros and source because the elements' MCS will contain all the information needed to properly install the elements; hence JCLIN is not necessary to define these elements. In addition, most DD statements found in JCLIN steps that identify target and distribution libraries and temporary work data sets are ignored by JCLIN processing. Therefore, unnecessary DD statements will not be created as part of the JCLIN for the BUILDMCS command output.

Load modules

For each candidate load module, SMP/E determines all of the modules that make up that load module. At least one of the component modules is associated with the superseding function (FMID), but others may not be. This is the case if a load module is composed of modules from multiple products, or FMIDs. A ++MOD statement will be built only for modules associated with the superseding function, but the load module is completely defined in the JCLIN with INCLUDE statements for all component modules. This is necessary because SMP/E cannot accurately determine from the zone information whether the superseding function or its associated SYSMODs supplied the JCLIN to fully define a load module, or only added modules to the load module using the LMOD operand on the ++MOD statement.

If a candidate load module's LMOD entry indicates the load module was copied, a JCLIN copy step to selectively copy the load module into the target libraries is created. Otherwise, a JCLIN link edit step is created. This JCLIN contains the link edit attributes, link edit control cards, and target library information from the LMOD entry. An INCLUDE statement is created for all modules defined as a component of the load module. The module's DISTLIB value is the ddname specified on the INCLUDE statement. The SYSLIB DD statement concatenation is also created if the LMOD entry contains a CALLLIBS subentry list.

ASSEM entries (inline assembler source)

ASSEM entries contain assembler source statements found inline when processing a JCLIN assembler step. During APPLY processing, an ASSEM entry can be assembled and link edited into a load module. This

occurs when SMP/E is building a load module and needs to include all modules that compose the load module, and the assembled ASSEM entry supplies the object deck for a needed module. The MOD entry created as a result of an assembled ASSEM entry has a DISTLIB of SYSPUNCH and no FMID values. Also, the SYSMOD that caused the assembly of an ASSEM entry contains no indicator that the ASSEM was used. Since SYSMOD entries will not identify candidate ASSEM entries, another method is used to determine the needed ASSEMs. When determining the modules that compose a load module, SMP/E will identify those modules with a DISTLIB of SYSPUNCH and no FMID values. If an ASSEM entry by the same name exists in the zone, then a JCLIN assembler step is created to identify the ASSEM entry. A corresponding INCLUDE statement with a ddname of SYSPUNCH is created in the link edit step for all load modules that contain the assembled ASSEM entry.

DLIB entries

For each element in the list of candidates, SMP/E determines if the element's distribution library has been totally copied to a target library. A distribution library has been totally copied if a DLIB entry describing the distribution library is found in the zone. The SYSLIB subentry of the DLIB entry identifies the target library. If any totally copied distribution libraries are found, a JCLIN copy step is created that contains one COPY statement for each DLIB entry.

Zone and data set sharing considerations

The following identifies the phases of BUILDMCS processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

2. BUILDMCS processing

   Target zone
   Update with shared enqueue.

   DLIB zone
   Update with shared enqueue.

   Note: The zones used depend on the zone type specified on the previous SET command.

3. Termination

   All resources are freed.

Chapter 5. The CLEANUP command

The CLEANUP command deletes entries from the SMPLTS, SMPMTS, SMPSTS, and SMPSCDS data sets. This is helpful for:

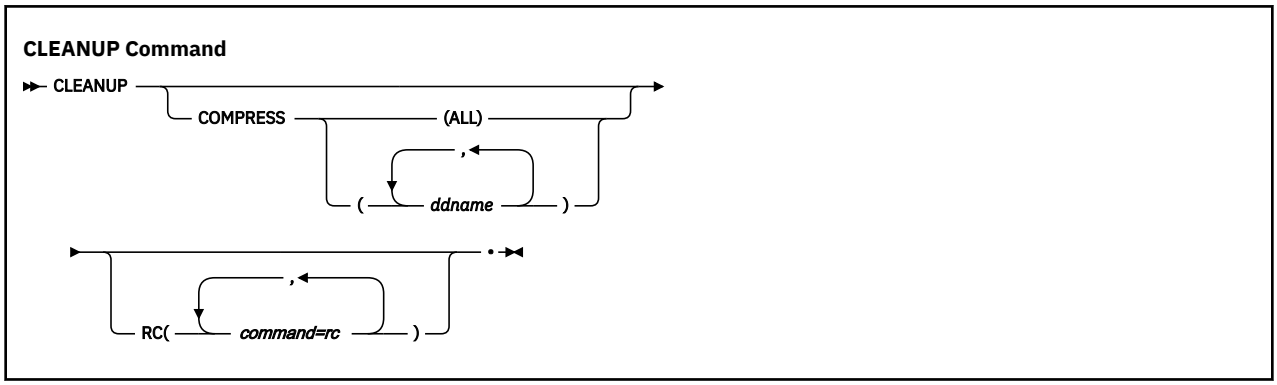
- APPLY followed by ACCEPT when several target libraries have been created from the same distribution library: When a SYSMOD is accepted into a distribution zone, the entries associated with the SYSMOD are automatically deleted from the SMPMTS, SMPSTS, and SMPSCDS for the related target zone. However, even if the SYSMOD was also applied to other target zones created from the same distribution zone, these data sets for the other target zones are not cleaned up.

  To delete the entries from these data sets, you can accept the SYSMOD and name these other target zones as the related zone. However, this updates the distribution library each time, which is time-consuming and can use up space in the distribution library data set. Instead, you can use the CLEANUP command, which deletes entries from the SMPMTS, SMPSTS, and SMPSCDS without updating the distribution library.
- ACCEPT followed by APPLY: When a SYSMOD is applied after it has been accepted, the entries associated with it are not deleted from the SMPMTS, SMPSTS, and SMPSCDS. To delete these entries, you can use the CLEANUP command.
- UPGRADE: When the UPGRADE command has been run to create an UPGLEVEL subentry for a zone, the SMPLTS data set for that zone does not need to retain a load module or program object unless it has both CALLLIBS and XZMOD subentries. You can use the CLEANUP command on the SMPLTS data set for a newly upgraded zone to delete from it any no longer needed load modules or program objects. You only need to run the CLEANUP command against an existing SMPLTS once, when its zone is first upgraded. Thereafter, SMP/E will no longer create unneeded load modules or program objects in it. You will never need to run the CLEANUP command (for this purpose) for an SMPLTS for a zone newly created with an UPGLEVEL subentry.

Zones for SET BOUNDARY

For the CLEANUP command, the SET BOUNDARY command must specify the target zone associated with the SMPMTS, SMPSTS, SMPSCDS, or SMPLTS data sets that should be cleaned up.

Syntax



Operands

COMPRESS
indicates the ddnames of the data sets to be compressed after the entries are deleted. The valid data sets are SMPMTS, SMPSCDS, SMPSTS, and (if the set-to zone has an UPGLEVEL subentry) SMPLTS.

- If you specify ALL, all the valid data sets are compressed.

- If you specify one or more specific ddnames, only the data sets they apply to are compressed.

Note: COMPRESS can also be specified as C.

RC

changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the CLEANUP command.

Before SMP/E processes the CLEANUP command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the CLEANUP command. Otherwise, the CLEANUP command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the CLEANUP command. Therefore, you must specify every command whose return code you want SMP/E to check.

Data sets used

These data sets might be needed to run the CLEANUP command. They can be defined by DD statements or, ordinarily, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLTS	SMPSCDS	SYSUT1
SMPCSI	SMPMTS	SMPSNAP	SYSUT2
SMPLOG	SMPOUT	SMPSTS	SYSUT3
SMPLOGA	SMPRPT	SYSPRINT	*zone*

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

Two reports are produced during CLEANUP processing:

- CLEANUP Summary report
- File Allocation report

See Chapter 34, "SMP/E reports," on page 457 for descriptions of these reports.

Example: Using CLEANUP with the COMPRESS operand

Assume that you want to clean up the data sets for target zone MVSTGT and at the same time compress the SMPMTS data set. Before cleanup, the data sets contain the entries shown in Table 6 on page 116.

Table 6. CLEANUP example: data set members before CLEANUP processing

Data set	Members
SMPSCDS	UZ00010 UZ00020

Table 6. CLEANUP example: data set members before CLEANUP processing (continued)	
Data set	**Members**
SMPMTS	MAC01 MAC02 MAC03 MAC04
SMPSTS	SRC11 SRC12 SRC13

Table 7 on page 117 shows the information contained in the MVSTGT zone about the MAC and SRC entries.

Table 7. CLEANUP example: service levels of elements			
Element	**FMID**	**RMID**	**UMID**
MAC01	JXY1234	UZ00010	AZ00100
MAC02	JXY1234	UZ00020	
MAC03	JXY1234	UZ00030	
MAC04	JXY2300	JXY2300	
SRC11	JXY1234	UZ00025	
SRC12	JXY1234	UZ00025	
SRC13	JXY2300	JXY2300	

Table 8 on page 117 shows the information contained in the related distribution zone about the SYSMODs associated with those MAC and SRC entries.

Table 8. CLEANUP example: status of SYSMODs		
SYSMOD	**Accepted**	**Superseded By**
AZ00100		
JXY1234	Yes	
JXY2300		
UZ00010	Yes	
UZ00020	Yes	
UZ00025		UZ00030
UZ00030	Yes	

You can use the following command to clean up and compress the data sets:

```
 SET BDY(MVSTGT) /* Process MVSTGT tgt zone
CLEANUP COMPRESS(SMPMTS)  /* Compress MTS at cleanup. */.
```

SMP/E deletes the entries shown in Table 9 on page 118 from the data sets:

Table 9. CLEANUP example: entries deleted by CLEANUP processing	
Data set	**Entries deleted**
SMPSCDS	UZ00010 UZ00020
SMPMTS	MAC02 MAC03
SMPSTS	SRC11 SRC12

SMP/E then compresses the SMPMTS and writes a CLEANUP Summary report.

Processing

The CLEANUP command deletes entries from the SMPMTS, SMPSTS, and SMPSCDS data sets. If the set-to zone has an UPGLEVEL subentry, the CLEANUP command deletes unneeded load modules or program objects from the SMPLTS.

SMP/E opens the specified target zone and its related distribution zone for read access. It also opens the SMPMTS, SMPSTS, SMPSCDS, and (if the set-to zone has an UPGLEVEL subentry) SMPLTS for update processing.

- For the SMPMTS, SMP/E checks which macros have MTSMAC entries in that data set and, for each entry, checks which SYSMODs are specified as the FMID, UMID, and RMID. It deletes the entries (and associated aliases) whose associated SYSMODs have been accepted into the distribution library or have been superseded by an accepted SYSMOD.
- For the SMPSTS, SMP/E checks which source elements have STSSRC entries in that data set and, for each entry, checks which SYSMODs are specified as the FMID, UMID, and RMID. It deletes the entries whose associated SYSMODs have been accepted into the distribution library or have been superseded by an accepted SYSMOD.
- For the SMPSCDS, SMP/E checks which SYSMODs have BACKUP entries. It deletes the entries for each SYSMOD that has been accepted into the distribution library or that has been superseded by an accepted SYSMOD.
- For the SMPLTS, if the set-to zone has an UPGLEVEL subentry, SMP/E checks if a load module or program object contains both CALLLIBS and XZMOD subentries. If the load module or program object contains CALLLIBS, but no XZMOD subentries, it deletes the base version of the load module or program object from the SMPLTS data set. If the set-to zone does not have an UPGLEVEL subentry, SMP/E will not delete load modules or program objects from SMPLTS.

SMP/E then generates a report listing all the deleted entries and writes it to the SMPRPT data set. It compresses any data sets, if requested, and then closes them.

If there are no entries in any of the data sets, SMP/E does no cleanup, but does compress any data sets if requested. It then closes the data sets and writes a report to SMPRPT.

If SMP/E could not open a data set, or if the target zone has no record of a particular entry in one of the data sets, SMP/E issues an error message, and CLEANUP processing fails.

Zone and data set sharing considerations

The following identifies the phases of CLEANUP processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

Global zone
Read without enqueue.

Target zone
Read without enqueue.

DLIB zone
Read without enqueue.

2. Processing

Target zone
Update with an exclusive enqueue.

DLIB zone
Read without enqueue.

3. Termination

All resources are freed.

Chapter 6. The DEBUG command

With the DEBUG command, you can request diagnostic processing from SMP/E—for example:

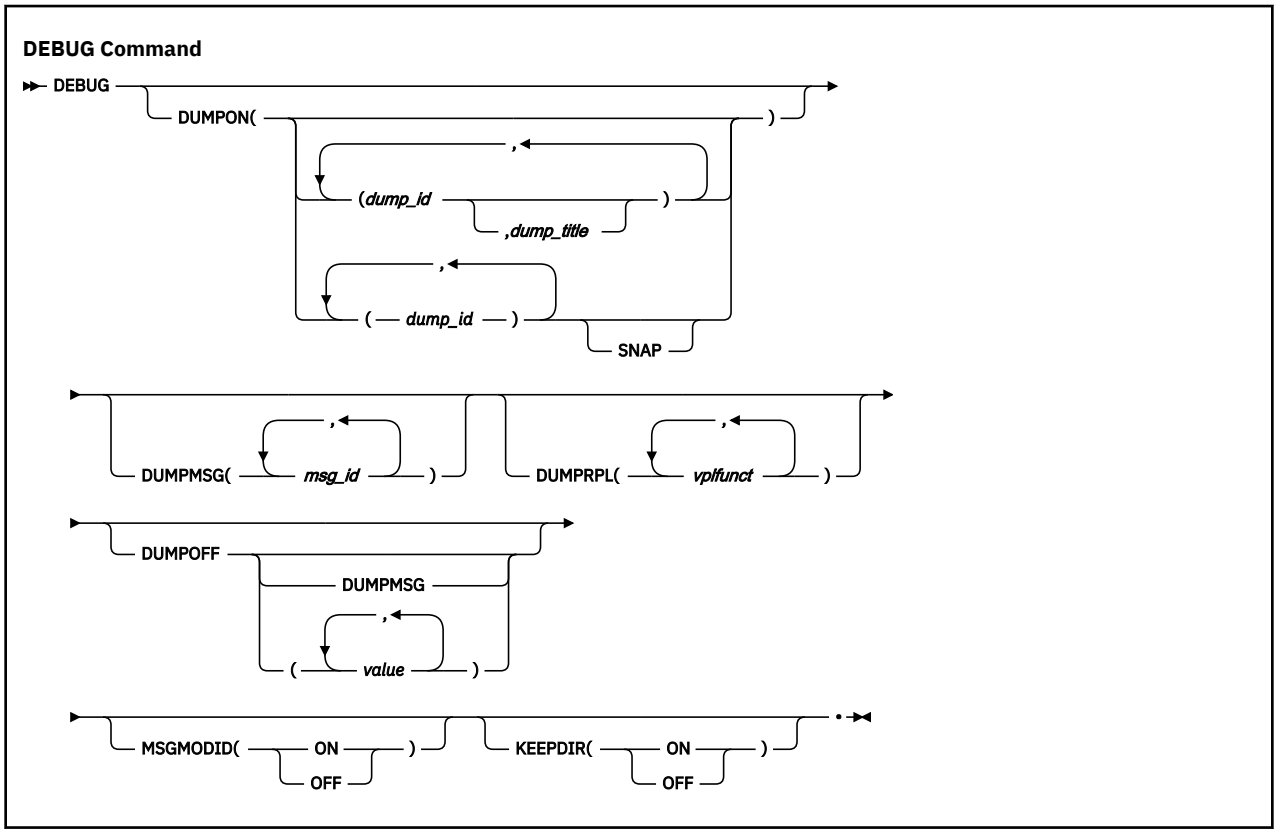
- Tracking the source of all SMP/E messages
- Dumping SMP/E control blocks and storage areas
- Dumping VSAM RPL control blocks
- Dumping SMP/E storage whenever specified messages are issued

You can use this command to provide additional documentation when reporting an SMP/E problem or when working with IBM to solve an SMP/E problem.

Zones for SET BOUNDARY

The DEBUG command is used to collect diagnostic information for problems with other SMP/E commands. Therefore, you should use the same SET BOUNDARY command for DEBUG as for the other commands you are debugging.

Syntax



Operands

Note: The DEBUG dump operands are for use when working with IBM to solve an SMP/E problem, not before you report a problem. Therefore, some of the information you need to specify on those operands is provided through IBM and is not included in this book.

DUMPON
specifies one or more dump points for which associated control blocks and storage areas are to be dumped. Unless SNAP is specified, the dump is formatted and written to SMPDEBUG. These are the values you can specify on the DUMPON operand:

dump-id
is a predefined dump point, whose name is provided by IBM. You must specify at least one dump point.

dump-title
is an optional, user-written title for SMP/E to include on the header page of all formatted dumps requested by the DUMPON operand and written to SMPDEBUG. The dump title may be up to 100 characters. If it contains parentheses, right and left parentheses must be in matched pairs. If SNAP is also specified, the dump title is not printed.

SNAP
indicates that the dump is to be written unformatted to the SMPSNAP data set. If SNAP is specified, any dump title specified is not printed.

DUMPOFF
specifies one or more dump points for which dumps are to be stopped. These are the values you can specify on the DUMPOFF operand:

blank
stops dumping for all dump points, including DUMPMSG.

DUMPMSG
stops dumping for all messages.

value
stops dumping for the specified dump point or VPLFUNCT value, which was provided by IBM. This dump point must have been activated by a previous DEBUG DUMPON or DEBUG DUMPRPL command.

Note: You can combine dump points and VPLFUNCT values on the same DEBUG DUMPOFF command.

DUMPMSG
indicates that a SNAP dump is to be taken of SMP/E storage when the specified SMP/E messages are issued. *msg-id* is the message ID without the severity level, such as GIM44301. You must specify at least one message ID.

DUMPRPL
indicates that a dump of the VSAM RPL control block and additional RPL information is to be taken. *vplfunct* is a VPLFUNCT value supplied by IBM. You must specify at least one VPLFUNCT value. The RPL dump is written to SMPDEBUG. The heading for the RPL dump contains the VPL function being performed when the dump was taken. This can be used to separate the dumps if you specify more than one VPLFUNCT value on the DEBUG command.

KEEPDIR
indicates whether SMP/E should keep temporary work directories created in a UNIX file system. Typically, these directories are deleted at command completion.

ON
keep temporary work directories at the completion of APPLY and ACCEPT commands.

OFF
do not keep temporary work directories at the completion of APPLY and ACCEPT commands (delete the directories). This is default processing.

MSGMODID
indicates whether to start or stop tracking which modules are issuing SMP/E messages.

Note: `MSGMODID` can also be specified as `M`.

ON
starts message tracking. Each SMP/E message is preceded by the name of the issuing module and the offset where the message was issued.

OFF
stops message tracking.

Syntax notes

- You must specify at least one operand.
- If you specify DUMPON or DUMPRPL and DUMPOFF on the same DEBUG command, these operands are processed in the order they occur. If you specify the same dump point on both operands, the last specification is used.
- DEBUG commands are generally used in pairs. The first one starts DEBUG processing and the second stops it. However, the second DEBUG command is not required if SMP/E is to do the same DEBUG processing for all the commands following the DEBUG command.

Data sets used

These data sets might be needed to run the DEBUG command. You can define them by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPDEBUG	SMPLOGA	SMPSNAP
SMPCSI	SMPLOG	SMPOUT	

Output

The File Allocation report is produced during DEBUG processing. For a description of this report, see Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the DEBUG command.

Example 1: Tracing SMP/E messages

Suppose that a problem occurred when you ran the following SMP/E commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    S(USR0001)         /* Apply user modification.   */.
```

Because the problem appeared to be in SMP/E and not in the USERMOD, you decided to report it to IBM. To help IBM determine the cause of the problem, you should also rerun the job with the message trace on. You can use the following commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
DEBUG    MSGMODID(ON)       /* Start message trace.       */.
APPLY    S(USR0001)         /* Apply user modification.   */.
DEBUG    MSGMODID(OFF)      /* Stop message trace.        */.
```

When you run this job, SMP/E precedes all messages for the APPLY command with the name and offset of the issuing module. This stops when the second DEBUG command is processed.

Note: The second DEBUG command is not required if no further SMP/E commands are to be traced. It is used in this example only to show that the trace can be turned on and off.

Example 2: Dumping control blocks and storage areas

Suppose that SYSMOD UZ12345 should have been selected when you ran the following SMP/E commands, but was not:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    PTFS               /* Apply PTFs for HBT1201.    */
         FORFMID(HBT1201)   /*                            */
         GROUP              /*                            */.
```

After you report the problem to IBM, you may be asked to rerun the job with certain dump points enabled, for example, dump point 1. This information helps IBM determine the cause of the problem. You may also want to give the dump a title that describes the problem. You can use the following commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
DEBUG    DUMPON ((1,SYSMOD  /* Specify debug dump point   */
         UZ12345 NOT        /* and dump title.            */
         SELECTED FOR       /*                            */
         APPLY))            /*                            */.
APPLY    PTFS               /* Apply PTFs for HBT1201.    */
         FORFMID(HBT1201)   /*                            */
         GROUP              /*                            */.
DEBUG    DUMPOFF(1)         /* Stop debug dump.           */.
```

When you run this job, SMP/E formats and dumps the control blocks and storage areas associated with dump point 1, then writes the dump to SMPDEBUG.

Example 3: Dumping a VSAM RPL control block

Suppose that when you ran the following SMP/E commands, SMP/E issued message GIM27901S on a VSAM error:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    PTFS               /* Apply PTFs for HBT1201.    */
         FORFMID(HBT1201)   /*                            */
         GROUP              /*                            */.
```

After you report the problem to IBM, you may be asked to rerun the job and request a dump of the VSAM RPL control block. You can use the following commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
DEBUG    DUMPRPL(VPLEXT)    /* Debug dump for VPLEXT.     */.
APPLY    PTFS               /* Apply PTFs for HBT1201.    */
         FORFMID(HBT1201)   /*                            */
         GROUP              /*                            */.
DEBUG    DUMPOFF(VPLEXT)    /* Debug dump off.            */.
```

When you run this job, a dump of the VSAM RPL control block, plus additional RPL information, is written to SMPDEBUG. The heading for the RPL dump contains the VPL function being performed when the dump is taken (in this case, VPLEXT).

Example 4: Dumping SMP/E storage when messages are issued

Suppose that when you ran the following SMP/E commands, SMP/E issued message GIM38201E, and you need more information about the problem:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    PTFS               /* Apply PTFs for HBT1201.    */
         FORFMID(HBT1201)   /*                            */
         GROUP              /*                            */.
```

After you report the problem to IBM, you may be asked to rerun the job and have SMP/E dump its storage whenever message GIM38201E is issued. You can use the following commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
DEBUG    DUMPMSG(GIM38201)  /* Debug dump for GIM38201.   */.
APPLY    PTFS               /* Apply PTFs for HBT1201.    */
```

```
          FORFMID(HBT1201)   /*                          */
          GROUP              /*                          */.
DEBUG     DUMPOFF(DUMPMSG)   /* Debug dump off.          */.
```

When you run this job, SMP/E dumps its storage and work areas to the SMPSNAP data set.

Processing

When SMP/E encounters the DEBUG command, it first checks whether the SMPDEBUG and SMPSNAP data sets exist. These are opened when the dump is about to be written. It then scans the command for valid operands.

- If you specified DUMPON, SMP/E checks whether the dump points are valid. Unless SNAP is specified, the control blocks and storage areas are formatted and written to SMPDEBUG. Otherwise, they are written unformatted to SMPSNAP.
- If you specified DUMPRPL, a dump of the VSAM RPL control block plus additional RPL information is written to SMPDEBUG when the specified VPL function is performed.
- If you specified DUMPMSG, a SNAP dump of SMP/E storage is written when the specified messages are issued.
- If you specified DUMPOFF, dumps are stopped for the specified dump points, or for all dump points if none are specified.
- If you specified MSGMODID(ON), all messages are prefixed with the following string:

  ```
  @module+X'offset'
  ```

  where:

  module
  is the name of the SMP/E module (without the GIM prefix) that issued the message.

  offset
  is the hexadecimal offset into the module where the message was issued.
- If you specified MSGMODID(OFF), messages are no longer prefixed with the module name and offset.

When SMP/E finishes processing the command following DEBUG, the SMPDEBUG and SMPSNAP data sets are closed.

DEBUG processing fails if a DUMPON dump point is incorrect, a required DD statement is missing, an incorrect VPLFUNCT value is specified with DUMPRPL, or a DUMPOFF dump point was not already active.

Chapter 7. The GENERATE command

With the GENERATE command, you can create a job stream that builds a set of target libraries from a set of distribution libraries. In that way, it is similar to system generation. However, the GENERATE command offers several advantages over system generation:

- GENERATE helps you reinstall products without SYSGEN support.

  System generation creates jobs to install only products that are included by the system generation macros. Products without this SYSGEN support are not included. As a result, when SYSGEN is used to create or replace a system, a number of products have to be reinstalled outside the SYSGEN process.

  GENERATE can create jobs to install **all** the products defined in a target zone, regardless of whether the products have SYSGEN support. As a result, when GENERATE is used to create or replace a system, no products have to be reinstalled outside the generation process.
- GENERATE creates job streams that are processed more efficiently.

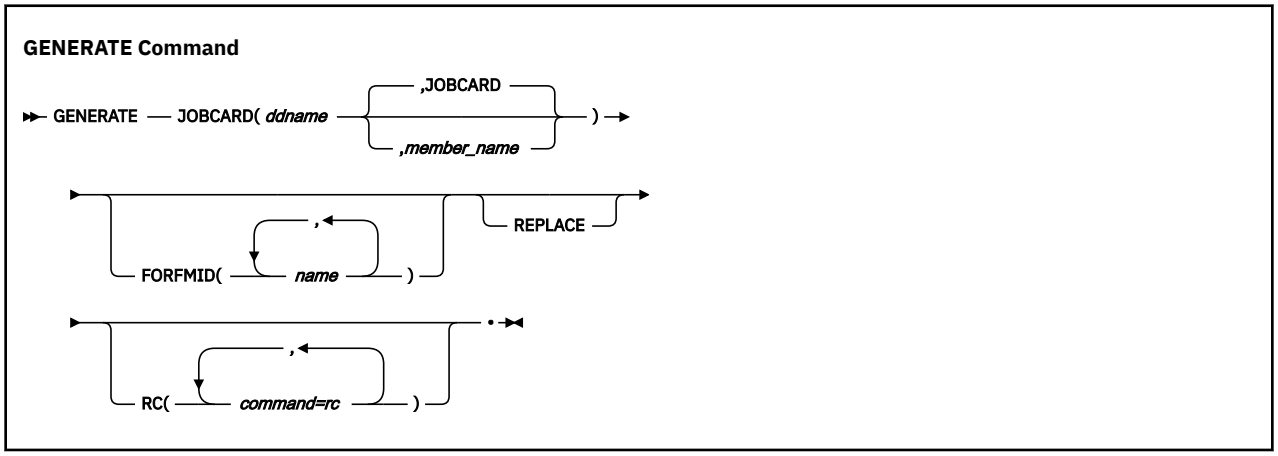
  The format of the system generation job stream depends on how the system generation macros are coded. For example, the number of products being installed and any changes in the system generation process may cause utilities to be called inefficiently.

  The format of the GENERATE job stream is based on an analysis of the target zone. One job is created for each target library. This reduces the number of utility calls for each data set and improves SMP/E performance by allowing the various utilities to run concurrently.

Zones for SET BOUNDARY

For the GENERATE command, the SET BOUNDARY command must specify the target zone containing the entries used to create the job stream.

Syntax



Operands

FORFMID
specifies the names of the FMIDs or FMIDSETs for which a job stream is to be generated. FORFMID should be used only if you want to create a job stream for specific products. To create a job stream describing all the products defined in the target zone, do not specify FORFMID.

Note: The FORFMID operand may include entries that do not have an FMID, such as the ASSEM entry. For more information, see “Determining whether a module must be assembled” on page 133.

The products you can specify on the FORFMID operand depend on why you are using the GENERATE command:

- To initialize a new target zone from an old target zone by having the JCLIN command process the GENERATE output

  In this case, you should specify only products without SYSGEN support for which JCLIN was not processed at ACCEPT time.

  Do not specify any products for which you have done a stage 1 system generation. The stage 1 output should be processed by the JCLIN command to initialize the target zone.

  Note: You do not need to specify any products for which you have processed inline JCLIN at ACCEPT time. Instead of using the GENERATE command to initialize the new target zone for these products, you should use the ZONECOPY or ZONEMERGE command to copy the distribution zone into the new target zone.

- To create installation job streams

  In this case, you can specify any of the products defined in the target zone. This includes products with SYSGEN support, products without SYSGEN support, and products for which you have processed inline JCLIN at ACCEPT time.

JOBCARD
indicates where SMP/E is to get the job card for the generated jobs. *ddname* is the ddname for the partitioned data set containing the job card member. *member-name* is the name of the member within that data set containing the job card. If *member-name* is not specified, the default is JOBCARD.

The JOBCARD operand is required.

REPLACE
indicates that the JCL created should allow:

- Existing members in a data set to be replaced when the copy utility is invoked
- Existing load modules to be replaced when the link-edit utility is invoked

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the GENERATE command.

Before SMP/E processes the GENERATE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the GENERATE command. Otherwise, the GENERATE command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the GENERATE command. Therefore, you must specify every command whose return code you want SMP/E to check.

Data sets used

These data sets might be needed to run the GENERATE command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPUNCH	*zone*
SMPCSI	SMPOBJ	SMPRPT	
SMPLOG	SMPOUT	SMPSNAP	

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the

ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Besides checking the element entries in the CSI data set, SMP/E also uses information from the sources shown in Table 10 on page 129 to create the output JCL for the GENERATE command. This information must, therefore, be defined before the GENERATE command is run.

Table 10. Sources of information for GENERATE output JCL

Source of input	**JCL it is used for**
JOBCARD member specified on the GENERATE command	JOB statement
UTILITY entries in the global zone (pointed to by the OPTIONS entry in effect for the target zone)	• EXEC statements for utilities • ddnames for print output data sets
DDDEF entries in the target zone	DD statements for: • Distribution libraries (DLIBs) • Target libraries (SYSLIBs) • Print output data sets • SYSUT1—4 data sets • Side deck libraries • Utility input libraries • Jave Home Directory (SMPJHOME)
SMPPARM member GIMDDALC	DD statements for: • Print output data sets • SYSUT1—4 data sets

Note: SMPJHOME is an optional DDDEF entry used to specify the path needed to find the Java runtime. SMP/E uses the SMPJHOME DDDEF entry to create an SMPJHOME DD statement in the HFSINST job, if a UNIX shell script is invoked during the installation of a file system element. When processing the HFSINST job, the GIMIAP service uses the SMPJHOME DD statement, if the UNIX shell script invokes a Java command.

Usage notes

Before using the GENERATE command, you must:

- Accept or restore all SYSMODs that have been successfully applied.
- Initialize the new target zone with the appropriate JCLIN.
- Define a target zone DDDEF entry for each distribution library and target library.
- Define the SMPPUNCH, SYSOUT, and temporary data sets required. You can use either DDDEF entries or GIMDDALC control statements in SMPPARM member GIMDDALC.
- Define SMPOUT and SMPRPT. You either can use DDDEF entries or DD statements. If you have only SMPOUT defined, the messages and reports are mixed together and are hard to read.

Output

GENERATE output is written to the SMPPUNCH data set.

Two reports are produced during GENERATE processing:

- File Allocation report
- GENERATE Summary report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

Examples

The following examples are provided to help you use the GENERATE command.

Example 1: Using GENERATE to install new products

Suppose that you want to install a new product into an existing set of distribution libraries, and then create a new set of target libraries to be controlled from target zone NEWMVS. You should follow these steps:

1. Install dummy function SYSMODs, if required. See the program directory for the product for additional information.
2. Receive and accept the product and any cumulative service. You can simplify the installation of SYSMODs without SYSGEN support if they contain inline JCLIN by processing that JCLIN when you accept the SYSMODs. To do this, the ACCJCLIN indicator must be set in the DLIBZONE entry before the SYSMODs are accepted. For more information about setting the ACCJCLIN indicator, see “Inline JCLIN” on page 36.
3. Allocate new target libraries.
4. Do a stage 1 generation to get the JCLIN for the SYSGEN-supported products. The JCL produced describes the structure of these products.
5. Run GENERATE against the old target zone, using the FORFMID operand to specify all products not included in the system generation. The JCL produced describes the structure of these products.

   If inline JCLIN was processed for any of these products at ACCEPT time, do not specify them on the GENERATE command.
6. Allocate a new target zone.
7. Copy the distribution zone into the target zone.

   If you processed inline JCLIN for SYSMODs without SYSGEN support at ACCEPT time, the target zone now describes the structure of those products.
8. Add any necessary entries to the new target zone and to the global zone—for example, DDDEF entries in the new target zone and OPTIONS entries in the global zone.
9. Run JCLIN against the new target zone, using the stage 1 generation output JCL from step 4 as input. This updates the target zone with information on the structure of all products included by SYSGEN.

   Note: If there are intersections between SYSGEN-supported products and products without SYSGEN support, process step 9 after step 10.
10. Run JCLIN against the new target zone, using the GENERATE output as input. This updates the target zone with information on the structure of the specified products without SYSGEN support.
11. Create a JOBCARD member (in this example, MYJOB). When running the GENERATE command, include a DD statement pointing to the data set containing that member (in this example, JOB).
12. Run GENERATE against the new target zone without the FORFMID operand, as in this example:

```
SET      BDY(NEWMVS)         /* Process NEWMVS tgt zone.   */.
GENERATE                     /* Generate JCL with          */
         JOBCARD(JOB,MYJOB) /* JOBCARD from data set.      */.
```

    This creates jobs to install all the products defined in the target zone.
13. Submit the jobs created by the GENERATE command.

Example 2: Reinstalling products not included by SYSGEN

A target zone may contain products that are installed by a system generation procedure, as well as products that are not. When a system is created or replaced by use of a SYSGEN, products in the target libraries that were not included in the SYSGEN must be reinstalled. The GENERATE command can help you reinstall them.

Note: If any elements (such as macros or modules) are common to a product with SYSGEN support and a product without it, you must make sure the proper version of the element is used. To do this, you should examine all SYSGEN and GENERATE output and edit it as necessary. You may also need to process the steps in an order different from that described in this section.

Here is a procedure you can follow to avoid having to reinstall products when you do a SYSGEN:

1. Accept or restore all applied SYSMODs. This is to make sure that the distribution libraries used by SYSGEN are at the same service level as the current target zone.
2. Receive and accept the SYSMODs to be installed. You can simplify the installation of SYSMODs without SYSGEN support, if they contain inline JCLIN, by processing that JCLIN when you accept the SYSMODs. To do this, set the ACCJCLIN indicator in the DLIBZONE entry before accepting the SYSMODs.
3. Allocate new target libraries.
4. Do a stage 1 generation to get the JCLIN for the SYSGEN-supported products. The JCL produced describes the structure of these products.
5. Run GENERATE against the old target zone, using the FORFMID operand to specify all products that are not included in the system generation. The JCL produced describes the structure of these products.

   If inline JCLIN was processed for any of these products at ACCEPT time, do not specify them on the GENERATE command.
6. Allocate a new target zone.
7. Copy the distribution zone into the new target zone. If you processed inline JCLIN for SYSMODs without SYSGEN support at ACCEPT time, the target zone now describes the structure of those products.
8. Add any necessary entries to the new target zone and to the global zone—for example, DDDEF entries in the new target zone and OPTIONS entries in the global zone.
9. Run JCLIN against the new target zone, using the stage 1 generation output JCL from step 4 as input. This updates the target zone with information on the structure of all products included by SYSGEN.

   Note: If there are intersections between SYSGEN-supported products and products without SYSGEN support, process step 9 after step 10.
10. Run JCLIN against the new target zone, using the GENERATE output as input. This updates the target zone with information on the structure of the specified products without SYSGEN support.
11. Create a JOBCARD member (in this example, MYJOB). When running the GENERATE command, include a DD statement pointing to the data set containing that member (in this example, JOB).
12. Run GENERATE against the new target zone without the FORFMID operand, as in this example:

```
SET      BDY(NEWMVS)        /* Process NEWMVS tgt zone.   */.
GENERATE                    /* Generate JCL with          */
         JOBCARD(JOB,MYJOB) /* JOBCARD from data set.      */.
```

    This creates jobs to install all the products defined in the target zone.
13. Run the GENERATE jobs to build the system.

Processing

The GENERATE command has three processing phases:

1. **Target zone analysis:** SMP/E analyzes the target zone to determine which modules, macros, source, load modules, data elements, hierarchical file system elements, and program elements must be created.
2. **JCL creation:** SMP/E constructs the JCL statements used to create the jobs to build the target libraries.
3. **Job generation:** SMP/E constructs the jobs necessary to create the target libraries.

Target zone analysis

Before building the installation job stream, SMP/E first determines:

- Which elements are defined as part of the system
- Which target libraries the elements should be installed in
- Which utilities to use to install the elements

This information is in the target zone element, LMOD, ASSEM, and DLIB entries.

Note: SMP/E checks all the entries, even if FORFMID was specified on the GENERATE command. As a result, SMP/E may issue error messages for elements for which no install job is created. These can be ignored.

Also note that specifying the FORFMID command does **not** reduce the amount of storage required for GENERATE command processing.

Determining which macros to install

To determine which macros to install and where to install them, SMP/E checks the MAC and DLIB entries. A macro should be installed if it meets all the following conditions:

- The macro has a DISTLIB subentry. DISTLIB indicates the distribution library containing the macro.
- The macro has a SYSLIB subentry, or the macro distribution library was totally copied during initial installation. If the library was copied, SMP/E looks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the macro is installed.

  Note:

  1. Because of how SMP/E determines SYSLIB values, if the DLIB entry specifies more than one SYSLIB subentry, the value used for the macro might not be for the correct target library.
  2. Macros not residing in any SYSLIB are stored in the SMPMTS during APPLY and are deleted during ACCEPT. However, GENERATE does not create any steps to copy these macros to the SMPMTS, because when GENERATE is used, SYSMODs are not applied and then accepted. Rather, as in system generation, the SYSMODs are accepted, and then the elements are installed in the target libraries. As a result, the SMPMTS has no members.
- The FMID that owns the macro matches an FMID specified on the FORFMID operand. This is not checked if FORFMID was not specified.

  Note: If FORFMID was not specified and SMP/E cannot determine which FMID owns the macro, the macro is still selected. If the macro is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the macros from the distribution library into the target library.

Determining which source to install

To determine which source to install and where to install it, SMP/E checks the SRC and DLIB entries. A source should be installed if it meets all the following conditions:

- The source has a DISTLIB subentry. DISTLIB indicates the distribution library containing the source.
- The source has a SYSLIB subentry, or the source distribution library was totally copied during initial installation. If the library was copied, SMP/E checks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the source is installed.

Note:

1. Because of how SMP/E determines SYSLIB values, if the DLIB entry specifies more than one SYSLIB subentry, the value used for the source might not be for the correct target library.
2. Source not residing in any SYSLIB is stored in the SMPSTS during APPLY and deleted during ACCEPT. However, GENERATE does not create any steps to copy this source to the SMPSTS, because when GENERATE is used, SYSMODs are not applied and then accepted. Rather, as in system generation, the SYSMODs are accepted, and then the elements are installed in the target libraries. As a result, the SMPSTS has no members.

- The FMID that owns the source matches an FMID specified on the FORFMID operand. This is not checked if FORFMID was not specified.

  Note: If FORFMID was not specified and SMP/E cannot determine which FMID owns the element, the source is still selected. If the source is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the source from the distribution library into the target library.

Note: For more information about processing assembled modules, see the next section.

Determining which modules to install

To determine which modules to install and where to install them, SMP/E checks the MOD, LMOD, and DLIB entries. A module should be installed if it meets all the following conditions:

- The module has a DISTLIB subentry. DISTLIB indicates the distribution library or other data set containing the module.
- The module has at least one LMOD subentry, or the module distribution library was totally copied during initial installation. If the module contains an LMOD subentry, SMP/E checks for a SYSLIB entry in the corresponding LMOD entry. If the module is part of a totally copied library, SMP/E checks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the module should be installed.
- The FMID that owns the module matches an FMID specified on the FORFMID operand. This is not checked if FORFMID was not specified.

  Note:

  1. If the module does not have an FMID subentry, it may still be eligible if it is assembled by use of a macro owned by an FMID that was specified on the FORFMID operand. This is described in “Determining whether a module must be assembled” on page 133.
  2. If FORFMID was not specified and SMP/E cannot determine which FMID owns the element, the module is still selected. If the module is not found in the distribution library, however, an error may result. This is not an error if one product has linked a module from another product that resides in a different target zone.

Besides checking which modules to install and which libraries to install them in, SMP/E must also determine these situations:

- Whether the module must be assembled before it is installed
- What load module to install the module into
- Whether to copy or link-edit the module

Determining whether a module must be assembled

To determine whether a module must be assembled, SMP/E checks whether the DISTLIB subentry is SYSPUNCH. If so, the module does not reside in a distribution library; it must then be assembled from some other source before it is link-edited. SMP/E must, therefore, determine how to assemble the module:

- If there is an ASSEM entry with the same name as the module, the statements in the ASSEM entry can be used to assemble the module.

- If there is no ASSEM entry, SMP/E checks for an SRC entry with the same name as the module. If one exists, the source can be used to assemble the module.
- If there are no ASSEM or SRC entries for the module, it cannot be assembled, nor can it be link-edited into any load modules.

If FORFMID was specified and the module being assembled does not have an FMID, it may still be eligible. If an ASSEM entry can be used to assemble the module, SMP/E checks whether that assembly uses macros owned by any of the specified FMIDs. If so, the module is included in the installation job stream. Otherwise, SMP/E assumes that the assembly is not needed for any of the selected FMIDs, and the module is not included in the installation job stream.

Note: To determine whether the ASSEM uses any selected macros, SMP/E checks all the MAC entries to see if any of them specify that ASSEM entry in the MAC GENASM list.

Because SYSPUNCH is used as a DISTLIB value for modules in the SMPOBJ data set, SMP/E also checks for a DD statement or DDDEF entry for SMPOBJ. This data set contains preassembled modules that can be used to avoid re-assembling the modules. If SMPOBJ is found, SMP/E checks whether it contains a member with the same name as the module that must be assembled. If so, the SMPOBJ version of the module can be used with no need to reassemble the module. Otherwise, the module must be assembled.

Determining where and how a module should be installed

To determine where and how the module should be installed, SMP/E checks the MOD entry, the LMOD entry, and the DLIB entry:

- If there is an LMOD subentry in the MOD entry, it indicates the load module that the module is part of. SMP/E checks the link-edit attributes in the corresponding LMOD entry to determine how to install the module.

  If the attributes indicate that the module has been copied, SMP/E writes JCL to selectively copy the module into the target library. Otherwise, SMP/E uses the specified attributes and link-edit control statements in the entry to link-edit the load module into the target library.
- If there is no LMOD subentry in the MOD entry, the module is part of a totally copied library; therefore, there is no LMOD entry to check. In this case, SMP/E writes JCL to copy the entire distribution library into the target library. In addition, a SELECT statement is generated for each module.

When determining which load modules to link and how to link them, SMP/E checks the LMOD subentries in the MOD entries to ensure that each LMOD entry is referred to by at least one MOD entry. If any LMOD is found without a reference from a MOD, the load module is not selected for installation.

SMP/E also checks which modules are included in each load module. If a load module is to be link-edited, and not all of the modules in that load module were selected, SMP/E writes an INCLUDE card for the old version of the load module from the SYSLMOD data set. If all the modules for the load module are selected, the old version of the load module is not included. This enables the FORFMID operand to generate JCL that adds modules to a product that already exists in the target libraries.

Note:

1. If the module has cross-zone (XZLMOD) subentries, SMP/E does not try to create the associated cross-zone load modules. The GENERATE command creates JCL only for load modules in the set-to zone. You can use the LINK MODULE command to create these load modules.
2. If a load module contains cross-zone (XZMOD) subentries, SMP/E does not try to include the associated cross-zone modules. However, it does issue warning messages indicating which cross-zone modules were left out.

Link-editing when a load module has a CALLLIBS subentry list

The link-edit steps for a load module with a CALLLIBS subentry list and its base version in the SMPLTS library are created as follows:

1. Building the base version of the load module in the SMPLTS library (this is done only if the load module also has an XZMOD subentry or if the set-to zone has no UPGLEVEL subentry)

- INCLUDE statements are built for all modules in the distribution libraries that are explicitly included in the load module.
- Of all the other link-edit control statements defined for the load module, the only ones specified are CHANGE and REPLACE. No other link-edit control statements are processed.
- All link-edit options defined in the LMOD entry are specified, except for ALIASES(ALL) and DYNAM(DLL). NCAL is always passed.

2. Building the executable load module in the target libraries
   - INCLUDE statements are built for all modules in the distribution libraries that are explicitly included in the load module. INCLUDE statements are built for the utility input members included in the load module.
   - All other link-edit control statements in the LMOD entry are specified. CALL is always passed.
   - All link-edit options defined in the LMOD entry are specified.
   - A SYSLIB allocation, as defined by the CALLLIBS subentry list, is specified.
   - A SYSDEFSD allocation is specified as defined by the SIDEDECKLIB subentry. If the SIDEDECKLIB subentry is SMPDUMMY, the SYSDEFSD allocation will be specified as a DUMMY data set.

Determining which other elements to install

SMP/E also determines which of the following elements to install:

- data elements
- hierarchical file system elements
- program elements

The way in which SMP/E makes this determination is similar for all of these elements.

To determine which elements to install and where to install them, SMP/E checks the element and DLIB entries. An element should be installed if it meets all the following conditions:

- The element entry has a DISTLIB subentry. DISTLIB indicates the distribution library containing the element.
- The element entry has a SYSLIB subentry, or the distribution library was totally copied during initial installation. In that case, SMP/E looks for a SYSLIB subentry in the DLIB entry for the distribution library. SYSLIB indicates the target library where the element is installed.

  Note: Because of how SMP/E determines SYSLIB values, if the DLIB entry specifies more than one SYSLIB subentry, the value used for the element might not be for the correct target library.
- The FMID that owns the element matches an FMID specified on the FORFMID operand. This is not checked if FORFMID was not specified.

  Note: If FORFMID was not specified and SMP/E cannot determine which FMID owns the element, the element is still selected. If the element is not found in the distribution library, however, an error may result.

SMP/E creates JCL to copy the elements from the distribution library into the target library.

JCL creation

After analyzing the target zone, SMP/E builds the following JCL statements for generating the jobs to install the elements:

- JOB card
- EXEC statement
- Distribution library and target library DD statements
- Utility work file DD statements
- Utility print output DD statements

- Other DD statements

JOB card

The JOB card is obtained from the library and member specified on the JOBCARD operand of the GENERATE command. If the member name is not specified, “JOBCARD” is taken as the default member name. If the job card member is not found, message GIM64001E is issued.

The job card member can contain any number of records. However, the first record in the member must be the job card, and there must be room for an 8-character job name. This is because SMP/E does not check the format of the job card when generating jobs. The generated job name is stored in columns 3 through 10 of the first card read from the job card member.

The following example shows the expected format of the job card:

```
//xxxxxxxx JOB 'accounting info',
//             …
//             …
```

EXEC statement

The GENERATE command creates assembly, copy, and link-edit steps. The information used to construct the EXEC statement for each of these steps is obtained from the OPTIONS entry for the target zone. The OPTIONS entry points to a UTILITY entry for each utility that SMP/E can call. Each UTILITY entry contains:

- The name of the program to call.
- The parameters to pass that program. (Based on information in the target zone, SMP/E may add more parameters.)

  Although you can specify up to 100 characters of data in the PARM field of a UTILITY entry, JCL restricts the length of the PARM field to a total of 100 characters. Therefore, only the first 50 characters of the UTILITY PARM field are used for the GENERATE command. The other 50 characters are reserved for SMP/E-generated parameters.

  Note:

  1. The limitation on the length of the EXEC PARM field does not apply to invocations of the hierarchical file system copy utility. The parameters for this utility are specified on the EPARM operand of the generated SELECT statement instead of on the EXEC statement. However, the maximum total length of the parameters to be passed on the EPARM operand is X'FFFF' bytes. If the length exceeds this number, SMP/E truncates the parameters at the limit and passes this value to the hierarchical file system copy utility.
  2. If the DFP level of the driving system on which SMP/E is running supports the binder, SMP/E supports PARM options exceeding 100 characters. In this case, SMP/E uses OPTIONS instead of PARM on the EXEC statement. GENOPTS is specified as the ddname, and a DD statement is created for GENOPTS. SMP/E then adds all other options after the GENOPTS DD statement. In this way, the PARM string limit of 100 characters can be exceeded for the binder link-edit step. SMP/E does not verify whether the DFP level of the system on which the GENERATE output is executed supports the binder.
  3. Before using the GENERATE command to create JCL that will update files in a UNIX file system, you may want to add `UID(0)` to the UTILITY entry PARM value so that the JCL created by GENERATE will include UID(0) in the execution parameter string for the link-edit utility. Consider specifying UID(0) if all the following are true:

     a. UID 0 authority is needed to update files in a UNIX file system.

     b. Your UID is not 0 but you are authorized to the BPX.SUPERUSER facility class profile. The UID(0) option, in this case, causes the binder to set an effective UID of 0 for its execution.

     c. The binder to be invoked by the generated JCL is at the proper level to understand the UID option.

     If you do specify UID(0) for GENERATE, you must remove it after GENERATE has run, so that it is not used for other SMP/E commands (such as APPLY, which handles setting the effective UID itself prior to invoking the binder).

If no OPTIONS entry is available, or if no UTILITY entries have been defined, the SMP/E default values are used. For a summary of these default values, see the UTILITY Entry (Global Zone) section in *z/OS SMP/E Reference*.

Distribution library and target library DD statements

The information used to generate the DD statements for the target libraries and distribution libraries is obtained from the DDDEF entries in the target zone. Generally, the name of the DDDEF entry matches the name of the DD statement generated. Sometimes, however, SMP/E uses information in a DDDEF entry to generate a DD statement with a name different from the DDDEF entry name. For example, when calling the link-edit utility to link a load module to LINKLIB, SMP/E uses the LINKLIB DDDEF entry to generate the SYSLMOD DD statement.

Utility work file DD statements

The information used to generate the utility work files (SYSUT1 through SYSUT4) may be obtained from DDDEF entries or from GIMDDALC control statements specified in SMPPARM member GIMDDALC. If there is no DDDEF entry for a work file, there must be a GIMDDALC control statement in SMPPARM member GIMDDALC. For more information about specifying GIMDDALC control statement, see the GIMDDALC section in *z/OS SMP/E Reference*.

When SMP/E generates assembly steps, it needs a temporary library to store the resulting object module. To generate the GIMDDALC statement for this work file, SMP/E looks for a SYSPUNCH DDDEF entry or for a SYSPUNCH GIMDDALC control statement in SMPPARM member GIMDDALC.

Utility print output DD statement

The information used to generate the SYSOUT files (SYSPRINT) is obtained from DDDEF entries or from GIMDDALC control statements specified in SMPPARM member GIMDDALC. If there is no DDDEF entry for a SYSPRINT data set, and you do not want to use the SMP/E SYSOUT defaults, there must be a GIMDDALC control statement in SMPPARM member GIMDDALC. For more information about specifying GIMDDALC control statement, see the GIMDDALC section in *z/OS SMP/E Reference*.

All other DD statements

SMP/E directly generates other DD statements for the input files (such as SYSIN and SYSLIN) for the various utilities.

Job generation

The final phase of GENERATE processing is the actual building of the installation jobs. These jobs are written to the SMPPUNCH data set. Jobs are created for these candidates:

- Elements to be copied
- Target libraries for link-edited load modules
- Load modules installed in libraries specified in a SYSLIB allocation
- Elements to be installed in a UNIX file system
- Totally copied libraries

Elements to be copied

Two jobs, COPYJOB and DEIINST, are produced for all copied elements. COPYJOB handles all element types, except for data elements, which are handled by DEIINST.

Within COPYJOB, there is a separate COPY statement (or COPYMOD for program elements or copied modules) for each unique combination of distribution library, target library, and element type. The COPY (or COPYMOD) and SELECT statements generated in this step are arranged by output library ddname, and by distribution library module name under each output library ddname.

Within DEIINST, one job step is generated for each target library. The name of each job step matches the ddname of the associated target library. Each job step installs multiple data elements from multiple distribution libraries into a single target library. This is done by invoking SMP/E program GIMIAP, which calls the appropriate copy utility to do the actual installation. COPY and SELECT statements are arranged by distribution library ddname and by element name under each distribution library ddname. For more information about the control statements passed to GIMIAP, see the GIMIAP section in *z/OS SMP/E Reference*.

Note:

1. If you specified REPLACE on the GENERATE command, each applicable statement (COPYMOD for program elements or copied modules, INVOKE for data elements, or COPY for other element types) indicates that replacement is allowed.
2. GENERATE determines whether to list the names of copied members based on the value of the LIST subentry in the copy UTILITY entry in effect for the set-to target zone. For more information about the LIST subentry, see the UTILITY Entry (Global Zone) section in *z/OS SMP/E Reference*.

Target libraries for link-edited load modules

One job is produced for each target library into which load modules must be link-edited. The name of the job matches the ddname of the target library; for example, the job to link-edit modules into SYS1.LINKLIB is LINKLIB. These jobs can contain multiple steps.

- The first steps are assemblies for any modules that are link-edited into this target library. These steps are named ASSM0001 through ASSM9999.
- Following the assembly steps are one or more link-edit steps for the target library. One link-edit step is generated for each unique set of link-edit attributes. These steps are named LINK0001 through LINK9999.

Note: If REPLACE was specified on the GENERATE command, the NAME statement for each load module indicates that replacement is allowed.

During any job step generation, if SMP/E cannot generate a required JCL statement, it issues message GIM64601E to identify the JCL statement that could not be generated, as well as the current job and step name. A single JCL comment is generated in place of the statement. The comment has the following format:

```
//*xxxxxxxx *** ERROR *** UNABLE TO GENERATE JCL STATEMENT
                          FOR yyyyyyyy
```

where:

xxxxxxxx
is the name that would have been generated on the JCL statement.

yyyyyyyy
is the name of the DDDEF entry, SYSOUT table entry, or temporary table entry that should have been used to generate this JCL statement.

If SMP/E determines that no elements should be installed, it issues message GIM64901E or GIM64902E.

Load modules installed in libraries specified in a SYSLIB allocation

When a load module specifies a SYSLIB allocation, the LKSYSLIB job is generated. In some cases, the SMPLTS job is also generated:

- If a load module has an XZMOD subentry or if the set-to zone does not have an UPGLEVEL subentry, an SMPLTS job is generated to link-edit the base version of the load module into the SMPLTS library before the creation of the final link-edit job. This job can also include load modules that are installed in target libraries that are part of another load module's SYSLIB allocation. SMP/E inserts a JCL comment statement (//*SMPLTS) immediately after the SMPLTS job card to ensure that when the JCLIN command processes the GENERATE output, SMP/E can detect the SMPLTS job and skip over it.

Note: When SMP/E detects the //*SMPLTS comment during JCLIN processing, it skips all the steps in the SMPLTS job. For this reason, the //*SMPLTS comment should never be modified.

- LKSYSLIB is the final link-edit job. It is generated to link-edit the executable version of the load module into the target libraries. SMP/E inserts a JCL comment statement (//*CALLLIBS=YES) immediately after the LKSYSLIB job to ensure that when the JCLIN command processes the GENERATE output, the SYSLIB DD statements in the output are processed and not ignored.

If there are no link-edits for load modules specifying a SYSLIB allocation, SMP/E does not create either of these jobs.

Elements to be Installed in a UNIX file system

When elements need to be installed in a UNIX file system, the HFSINST job is generated to invoke the hierarchical file system copy utility for those elements:

- One job step is generated per target library within a UNIX file system.
- The name of each job step matches the ddname for the associated target library.
- Each job step installs multiple hierarchical file system elements from multiple distribution libraries into a single target library. This is done by invoking the SMP/E program GIMIAP, which calls the hierarchical file system copy utility to do the actual installation. For more information about the control statements passed to GIMIAP, see the GIMIAP section in *z/OS SMP/E Reference*.
- The DD statement for the target library specifies the pathname, not a data set name.
- COPY and SELECT statements are arranged by distribution library ddname and by element name under each distribution library ddname.
- The first two bytes after EPARM(on the SELECT control statement may appear as blanks or odd characters. However, they are valid data and should not be changed or deleted.

If no PATH value was specified in the DDDEF entry for a target library DD statement in the HFSINST job, SMP/E issues an error message. Instead of generating the required JCL statement, SMP/E produces a comment with the following format:

```
//*xxxxxxxx *** ERROR *** INFORMATION MISSING FOR THIS DDNAME
```

where:

xxxxxxxx
is the name of the DDDEF entry that should contain a PATH subentry for the hierarchical file system element's target library.

Totally copied libraries

The final job produced is a copy step for all totally copied distribution libraries. It contains one IEBCOPY COPY statement for each DLIB entry in the target zone. This step is not meant to be executed, because all the elements are selectively copied. It is, therefore, preceded by two // statements, which cause the reader or interpreter to ignore it. This step is generated so SMP/E can reconstruct the DLIB entries if the GENERATE output is processed by the JCLIN command.

Using the output from GENERATE

This section describes the following considerations for using the output job streams from the GENERATE command:

- Multiprogramming considerations for running the jobs concurrently
- Considerations for using the jobs to reinstall products

Multiprogramming considerations

The GENERATE command produces several jobs. The first, COPYJOB, does all the copies. You should run COPYJOB first so the subsequent jobs can run concurrently. This not only constructs all the macro and

source libraries the other steps use, but prevents the subsequent jobs from using the data sets when COPYJOB needs them.

Note: If an LKSYSLIB job is generated, the link-edit jobs created by GENERATE might not be able to run concurrently. This can occur when the SYSLMOD data set specified in one of the steps in the LKSYSLIB job is the same as a SYSLMOD data set specified in another link-edit job.

Reinstalling products using GENERATE

A target zone can contain products that are installed by a system generation procedure, as well as products that are not. Typically, when you run a generation procedure, such as SYSGEN, the products included by the generation macro are installed in a new set of target libraries. Any products not included by the generation macro must be reinstalled in those new libraries to make the system complete.

With the GENERATE command, you can avoid having to separately reinstall products not included in a generation procedure. To do this, first make sure the target zone for the new target libraries is updated with JCLIN that describes all the products that will be installed in those libraries.

- The JCLIN for the products included by the generation procedure is in the stage 1 generation output.
- The JCLIN for the products not included can be obtained by running the GENERATE command against the target zone for the current target libraries, and by specifying the desired products on the FORFMID operand.

  If you processed inline JCLIN at ACCEPT time for any of these products, you can update the target zone with that JCLIN by copying the distribution zone into the target zone.

Once the new target zone describes all these products, you can run the GENERATE command without the FORFMID operand to create jobs that install all the products defined in that zone. You can then run these jobs to create the system, without having to separately reinstall any products. This procedure is described in more detail in “Example 2: Reinstalling products not included by SYSGEN” on page 131.

Zone and data set sharing considerations

The following are the phases of GENERATE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

2. Processing

   Target zone
   Update with exclusive enqueue.

3. Termination

   All resources are freed.

Chapter 8. The GZONEMERGE command

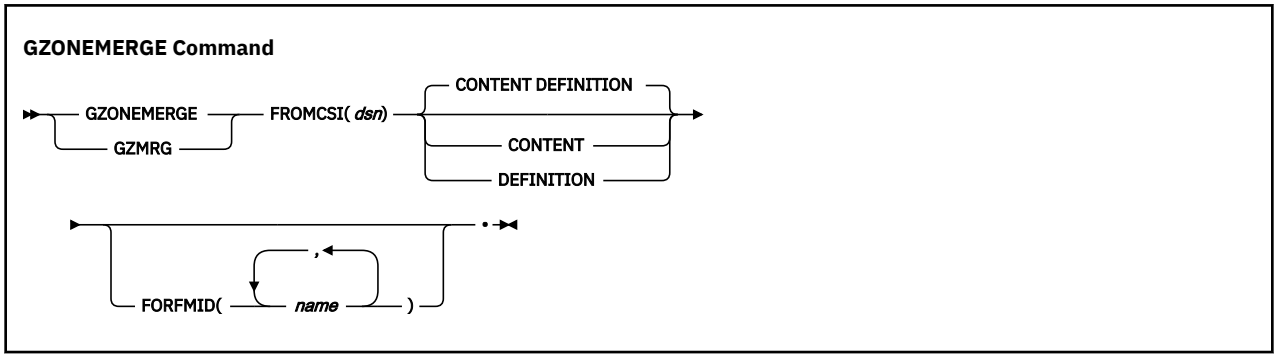
The GZONEMERGE command copies information from one SMP/E global zone to another. Possible uses of the GZONEMERGE command include:

- Copying data for specific FMIDs from one existing global zone to another existing global zone.
- Priming a new global zone with data from an existing global zone before you build a new system.

Zones for SET BOUNDARY

For the GZONEMERGE command, the SET BOUNDARY command must specify the global zone into which the data is to be merged.

Syntax



Operands

FROMCSI
specifies the global zone CSI data set name from which the data to be merged is to be obtained (the *originating* zone). The data set name (*dsn*) must be from 1 to 44 characters long, and cannot be split across input lines. This operand is required.

CONTENT
indicates that SYSMOD and HOLDDATA entries should be merged. Associated will also be copied from the *originating* SMPPTS to the *destination* SMPPTS data set. This operand is optional.

Note: CONTENT can also be specified as CON.

DEFINITION
indicates that definition entries from the originating global zone should be merged. The following entries are considered definition entries in the originating global zone:

- DDDEF
- FMIDSET
- GLOBALZONE
- OPTIONS
- ORDER
- UTILITY
- ZONESET

This operand is optional.

Note:

1. DEFINITION can also be specified as DEF.
2. The DDDEFs for the SMPPTS and its spill data sets are not merged during GZONEMERGE processing.

FORFMID
indicates that FEATURE, HOLDDATA, PRODUCT, and SYSMOD entries for the specified FMIDs or FMIDSETs should be merged. Any FMIDSETs specified must already be defined in the originating global zone. Associated will also be copied from the *originating* SMPPTS to the *destination* SMPPTS data set.

FORFMID is used to identify specific content entries. Therefore, if FORFMID is specified, CONTENT is assumed for only those entries associated with the FORFMID operand. In effect, the FORFMID operand requests content entries for only those FMIDs specified on the FORFMID operand. This operand is optional.

If none of the optional operands are specified (CONTENT, DEFINITION, or FORFMID), the GZONEMERGE command will attempt to merge the entire contents of the originating global zone into the destination global zone.

Syntax notes:

- If CONTENT or FORFMID is specified and DEFINITION is not, then only content entries are processed.
- If DEFINITION is specified and CONTENT and FORFMID are not, then only definition entries are processed.
- If both CONTENT and FORFMID are specified, only the FMIDS specified in FORFMID are merged.

The original SMPPTS must be defined by a DDDEF in the global zone of the original CSI data set. The destination SMPPTS must be defined either by a DDDEF in the global zone of the destination CSI data set or by a DD statement. If you are merging into a new empty CSI data set, you must allocate the new CSI data set and initialize it with a GIMZPOOL record before you run the GZONEMERGE command.

Data sets used

These data sets might be needed to run the GZONEMERGE command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

fromcsi	SMPLOG	SMPRPT	SYSUT2
frompts	SMPLOGA	SMPSNAP	SYSUT3
SMPCNTL	SMPOUT	SYSPRINT	SYSUT4
SMPCSI	SMPPTS	SYSUT1	

Note:

1. All of these data sets are for the destination global zone.
2. *fromcsi* is the data set name specified in the FROMCSI operand. The data set pointed to by this operand is the *originating* global zone CSI data set name.
3. *frompts* is the SMPPTS data set associated with the *originating* global zone CSI data set name.

Usage notes

The GZONEMERGE command is intended to simplify the process of migrating information from one global zone into another. It does not eliminate the need for you to decide how to configure the SMP/E environment to support the products to be used.

If either the CONTENT or FORFMID operand is specified or assumed by default, SMPPTS data sets are required for both the originating global zone and the destination global zone. These two SMPPTS data sets must have different names.

The originating SMPCSI data set is always required.

Before merging one global zone into another, it is recommended that you first clean up the originating global zone and its related SMPPTS to reduce the amount of data to be merged. You should do this by deleting all unneeded SYSMOD and HOLDDATA entries from the global zone, and deleting unneeded from the SMPPTS. You can use the REJECT NOFMID command to delete those entries related to FMIDs that are no longer needed. Also, rather than manually determine which FMIDs to delete, you can use the sample programs GIMCRSAM and GIMPRSAM (provided in SYS1.SAMPLIB) to help you create a REJECT NOFMID command for the FMIDs that are superseded or deleted in the DLIB zones you specify.

Output

Two reports are produced during GZONEMERGE processing:

- File Allocation report
- GZONEMERGE report

These reports are described in "GZONEMERGE report" on page 485

Examples

The following examples are provided to help you use the GZONEMERGE command:

- "Example 1: Merge definition entries" on page 143
- "Example 2: Merge content entries" on page 143

Example 1: Merge definition entries

You can use the GZONEMERGE command to merge the entries from an existing global zone that do not reflect system content. This can be done by using the DEFINITION operand of the GZONEMERGE command. The entries that are merged in this case are:

- DDDEF
- FMIDSET
- GLOBALZONE
- OPTIONS
- ORDER
- UTILITY
- ZONESET

The following SMP/E command merges the DEFINITION entries:

```
SET      BDY(GLOBAL)                      /* Set to destination global zone. */.
GZONEMERGE                                /* Merge global zones              */
         FROMCSI(SMPE.ZOSR1.GLOBAL.CSI) /* From this GLOBAL CSI data set   */
         DEFINITION                       /* Definition entries only.        */.
```

Figure 5. Example of GZONEMERGE of definition entries

Example 2: Merge content entries

You can use the GZONEMERGE command to merge the entries associated with specific FMIDs that reflect system content. This can be done by using the FORFMID operand of the GZONEMERGE command. The entries that are merged in this case are the SYSMOD and HOLDDATA entries associated with the FMIDs or FMIDSETs specified on the FORFMID operand.

Note: The associated are copied from the originating global zone's SMPPTS into the destination global zone's SMPPTS data set.

The following SMP/E command merges the content entries:

```
SET      BDY(GLOBAL)                       /* Set to destination global zone. */.
GZONEMERGE                                 /* Merge global zones.             */
         FROMCSI(SMPE.ZOSR1.GLOBAL.CSI)    /* From this GLOBAL CSI data set.  */
         FORFMID(MSB0001, MSB0002)         /* Merge SYSMODs and HOLDDATA for  */
                                           /* these FMIDs or FMIDSETs.        */.
```

Figure 6. Example of GZONEMERGE of content entries

Processing

With the GZONEMERGE command, you can merge a specified global zone into another specified global zone. After the merge operation is complete, the originating global zone still exists in the CSI data set.

Note: Trying to merge a target zone or a distribution zone with the GZONEMERGE command causes an error. Use the ZONEMERGE command to merge target zones or distribution zones.

The syntax refers to two types of zone entries: CONTENT and DEFINITION. CONTENT entries are created by SMP/E; DEFINITION entries are set up by the user. If neither CONTENT nor DEFINITION is specified, the default is to merge both the CONTENT and the DEFINITION entries. The following lists show how the various entries are categorized:

- CONTENT type entries
  - FEATURE entries
  - HOLDDATA entries
  - PRODUCT entries
  - SYSMOD entries
- DEFINITION type entries
  - DDDEF entries
  - FMIDSET entries
  - GLOBALZONE entries
  - OPTIONS entries
  - ORDER entries
  - UTILITY entries
  - ZONESET entries

The GZONEMERGE command operates on the two types of entries as follows:

- For content entries, SMP/E will:
  - Merge the entries from the *originating* global zone into the *destination* global zone.
  - Copy the from the *originating* SMPPTS to the *destination* SMPPTS.
- For definition entries, SMP/E will:
  - Merge the entries from the *originating* global zone into the *destination* global zone.

The GZONEMERGE command does not create a new global zone. Both global zones must have been previously defined.

FMID applicability

SMP/E determines whether each value specified on the FORFMID operand is an FMID or FMIDSET, and expands the FMIDSETs into FMIDs. SMP/E then ensures that each of these FMIDs is listed in the FMID subentry of the GLOBALZONE entry of the originating global zone.

If a FUNCTION SYSMOD name specified on the FORFMID operand is valid, but the actual SYSMOD is not received in the global zone, then SMP/E goes on to determine if there are any associated FEATURE, HOLDDATA, PRODUCT, or SYSMOD entries for that FMID.

Determine SYSMODs associated with FMIDs

For each FMID specified on the FORFMID operand, SMP/E determines what other SYSMODs in the originating global zone are associated with it. An associated SYSMOD is one that has an FMID on the ++VER statement that matches a validly specified FMID on the FORFMID operand. This is in addition to the SYSMOD whose name matches an FMID name specified on the FORFMID operand. Associated SYSMODs are candidates for the merge operation when the FORFMID operand is specified.

For each base function, all its dependent functions are merged. Also, the service and HOLDDATA for both the base function and its dependent functions are merged.

Determine HOLDDATA entries associated with FMIDs

For each FMID that is specified on the FORFMID operand, SMP/E determines what HOLDDATA entries in the originating global zone are associated with it. An associated HOLDDATA entry is one that has an FMID subentry that matches a valid FMID on the FORFMID operand of the GZONEMERGE command. Associated HOLDDATA entries are a candidate for the merge operation when the FORFMID operand is specified.

If an FMID specified on the FORFMID operand is a base function, and some dependent functions and their service are being merged according to the rules laid out for determining SYSMODs associated with an FMID, then SMP/E merges the HOLDDATA associated with the dependent functions, even though the dependent functions were not explicitly specified by the user.

Determine FEATURE entries associated with FMIDs

For each FMID that is specified on the FORFMID operand, SMP/E determines what FEATURE entries in the originating global zone are associated with it. An associated FEATURE entry is one that has at least one FMID value in its FMID subentry list that matches an FMID specified on the FORFMID operand or matches the FMID of a SYSMOD entry that is selected for merging.

Determine PRODUCT entries associated with FMIDs

For each FMID that is specified on the FORFMID operand, SMP/E determines what PRODUCT entries in the originating global zone are associated with it. An associated PRODUCT entry is one whose name (in the form *prodid,vv.rr.mm*) matches the PRODUCT subentry value of a FEATURE entry that has been selected for merging.

GLOBALZONE entry updates for content entries

When merging content entries, whether for specific FMIDs or for the entire global zone, SMP/E updates the destination global zone's GZONE entry's SREL and FMID subentries if the SRELs and FMIDs being merged are not present in the destination global zone. SMP/E always does this, whether DEFINITION is specified or not. If a GZONE SREL subentry for an FMID that is being merged exists only in the originating global zone, then that GZONE SREL subentry is merged into the destination global zone. If a GZONE SREL subentry for an FMID that is being merged exists in both the originating and destination global zones, then that GZONE SREL subentry is not merged. If a GZONE FMID subentry for an FMID that is being merged exists only in the originating global zone, then that GZONE FMID subentry is merged into the destination global zone. If a GZONE FMID subentry for an FMID that is being merged exists in both the originating and destination global zones, then that GZONE FMID subentry is not merged.

The following table lists possible combinations of optional operands that can be specified on the GZONEMERGE command and the resulting merge processing.

Table 11. GZONEMERGE merge processing options

FORFMID specified	CONTENT specified	DEFINITION specified	Content processing	Definition processing
Yes	Yes	Yes	Only content entries for the specified FMIDs are merge candidates	All definition entries are merge candidates

Table 11. GZONEMERGE merge processing options (continued)

FORFMID specified	CONTENT specified	DEFINITION specified	Content processing	Definition processing
Yes	Yes	No	Only content entries for the specified FMIDs are merge candidates	Only the GZONE SREL and FMID subentries are updated based on the FMIDs specified
Yes	No	No	Only content entries for the specified FMIDs are merge candidates	Only the GZONE SREL and FMID subentries are updated based on the FMIDs specified
Yes	No	Yes	Only content entries for the specified FMIDs are merge candidates	All definition entries are merge candidates
No	No	No	All content entries are merge candidates	All definition entries are merge candidates
No	Yes	Yes	All content entries are merge candidates	All definition entries are merge candidates
No	No	Yes	No content entries are merge candidates	All definition entries are merge candidates
No	Yes	No	All content entries are merge candidates	Only the GZONE SREL and FMID subentries are updated

When either CONTENT or FORFMID is specified, SMP/E may update the FMID and SREL subentries in the GZONE entry for the destination global zone:

- If CONTENT is specified, then:
  - SMP/E copies each FMID and SREL subentry in the original global zone's GZONE entry to the destination global zone's GZONE entry, unless that subentry already exists in the destination GZONE entry.
- If FORFMID is specified, then:
  - For each selected FMID that has an FMID subentry in the GZONE entry for the original global zone, SMP/E copies that FMID subentry to the GZONE entry for the destination global zone, unless that subentry already exists in the destination GZONE entry.
  - For each SREL assigned to the selected FMIDs, SMP/E copies its SREL subentry in the original global zone's GZONE entry to the destination global zone's GZONE entry, unless that subentry already exists in the destination GZONE entry.

Merging SYSMOD entries

SYSMOD entries are merged when either CONTENT or FORFMID is specified on the GZONEMERGE command. If CONTENT is specified, SMP/E will consider all SYSMOD entries in the original global zone as candidates for merging. If FORFMID is specified, SMP/E will consider only the SYSMOD entries associated with the selected FMIDs for merging.

SMP/E reads through the original zone, gathering data on the candidate SYSMOD and HOLDDATA entries. For each candidate entry in the originating zone, SMP/E checks to see whether that entry exists in the destination zone.

- If a SYSMOD entry is not found, then the entry from the originating global zone is stored in the destination global zone.
- If a SYSMOD entry is found, processing continues to determine the higher level SYSMOD. If the originating global zone contains a higher or equal level of the SYSMOD than the destination global zone, then the entry from the originating global zone replaces the existing SYSMOD entry in the destination global zone. Any existing HOLDFIXCAT (or HOLDERR, HOLDSYS, and HOLDUSER) subentries from the destination global zone will be retained. If the destination global zone contains a higher level of the SYSMOD than the originating global zone, then processing continues to the next entry.

  SMP/E uses the REWORK subentry of the global zone SYSMOD entry to determine if a SYSMOD in the originating global zone is at a higher level than the same-named SYSMOD in the destination zone. If the REWORK date of the SYSMOD in the originating global zone is more recent than the REWORK date of the

SYSMOD in the destination global zone, then that SYSMOD is considered to be at a higher level in the originating global zone and would be merged.

- If a SYSMOD entry in the originating global zone has a SOURCEID subentry that matches the name of the ORDER entry from which it originated, and that ORDER entry is being renamed because a like-named entry already exists in the destination global zone, then SMP/E ensures that the SOURCEID value matches the new name for the ORDER entry. To accomplish this it is necessary to process DEFINITION entries before CONTENT entries.

Merging HOLDDATA entries

HOLDDATA entries are merged when either CONTENT or FORFMID is specified on the GZONEMERGE command. If CONTENT is specified, SMP/E considers all HOLDDATA entries in the original global zone as candidates for merging. If FORFMID is specified, SMP/E considers only the HOLDDATA entries associated with the selected FMIDs for merging. The new FIXCAT type HOLDDATA entries may be considered candidates for merging as well.

For each candidate HOLDDATA entry in the originating zone, SMP/E checks to see whether that HOLDDATA entry exists in the destination zone.

- If a HOLDDATA entry is not found, the entry from the originating global zone is stored in the destination global zone. In this case, the GZONEMERGE report indicates that the entry was 'MERGED'.
- If a HOLDDATA entry is found in the destination zone, processing continues to the next entry. The GZONEMERGE report indicates that the entry was 'NOT MERGED' because there was a 'DUPLICATE HOLDDATA ENTRY IN THE GLOBAL ZONE'.

To uniquely identify each HOLDDATA entry, SMP/E uses the combination of the hold category, SYSMOD ID, and reason ID that are associated with that HOLDDATA entry. A HOLDDATA entry that exists in both global zones with the same hold category, SYSMOD ID, and reason ID is considered the same HOLDDATA entry and is not merged. All other HOLDDATA entries are merged.

When SMP/E copies a HOLDDATA entry from the originating global zone into the destination global zone, it also creates the corresponding HOLDDATA subentry in the SYSMOD entry for the held SYSMOD. This is true even if the SYSMOD entry is not present in the originating or destination global zones, a situation that occurs when the HOLDDATA has been received in the originating global zone, but its SYSMOD has not.

In addition, for FIXCAT type HOLDDATA entries, if the RESOLVER operand was specified on the ++HOLD FIXCAT statement, SMP/E determines if the SYSMOD named on the RESOLVER operand exists in the destination global zone. If a SYSMOD entry is found for the resolving SYSMOD, then SMP/E stores the fix category values specified on the CATEGORY operand as SOURCEID subentries in the SYSMOD entry for that resolving SYSMOD. If the SYSMOD entry is not found for the resolving SYSMOD, then SMP/E assigns no SOURCEID value.

Merging FEATURE entries

FEATURE entries are merged when CONTENT is specified or implied. If FORFMID is not specified, SMP/E will consider all FEATURE entries in the original global zone as candidates for merging. If FORFMID is specified, SMP/E will consider a FEATURE entry a candidate for merging if at least one FMID value in the FMID subentry list for the FEATURE entry matches an FMID specified on the FORFMID operand of the GZONEMERGE command or the FMID of a SYSMOD entry that is selected for merging.

SMP/E reads through the original zone, gathering data on the candidate FEATURE entries. For each candidate entry in the originating zone, SMP/E checks to see whether that entry exists in the destination zone.

- If a FEATURE entry is not found, then the entry from the originating zone is stored in the destination zone.
- If a FEATURE entry is found, processing continues to determine the higher level of the FEATURE entry. If the originating global zone contains a higher or equal level of the FEATURE entry, then the entry from the originating global zone is stored in the destination global zone, overlaying the existing entry. If the

destination global zone contains a higher level of the FEATURE entry, then processing continues to the next entry.

SMP/E uses the REWORK subentry of the global zone FEATURE entry to determine if the FEATURE entry in the originating global zone is at a higher level than the same FEATURE entry in the destination zone. If the rework date of the FEATURE entry in the originating global zone is more recent than the REWORK date of the FEATURE entry in the destination global zone, then that FEATURE entry is considered to be at a higher level in the originating global zone and would be merged.

If a FEATURE entry is to be merged into the destination global zone, SMP/E will ensure the SYSMODs identified in the FEATURE entry's FMID subentry are associated to the FEATURE in the destination global zone. Specifically, for each SYSMOD identified by a FEATURE:

- If a SYSMOD entry exists in the destination global zone, then SMP/E will simply update the SYSMOD entry to add a FEATURE subentry for the FEATURE being merged.
- If a SYSMOD entry does not exist in the destination global zone, then SMP/E will create an entry for it, containing only a FEATURE subentry for the FEATURE being merged.

Merging PRODUCT entries

PRODUCT entries are merged when CONTENT is specified or implied. If FORFMID is not specified, SMP/E will consider all PRODUCT entries in the original global zone as candidates for merging. If FORFMID is specified, SMP/E will consider a PRODUCT entry a candidate for merging if:

- A FEATURE entry that has been selected for processing contains a PRODUCT subentry value that matches the name on the PRODUCT entry. The PRODUCT subentry value is the combined *prodid* and *vv.rr.mm* values in the form *prodid,vv.rr.mm*.

SMP/E reads through the original zone, gathering data on the candidate PRODUCT entries. For each candidate entry in the originating zone, SMP/E checks to see whether that entry exists in the destination zone.

- If a PRODUCT entry is not found, then the entry from the originating zone is stored in the destination zone.
- If a PRODUCT entry is found, processing continues to determine the higher level of the PRODUCT entry. If the originating global zone contains a higher or equal level of the PRODUCT entry, then the entry from the originating global zone is stored in the destination global zone, overlaying the existing entry. If the destination global zone contains a higher level of the PRODUCT entry, then processing continues to the next entry.

  SMP/E uses the REWORK subentry of the global zone PRODUCT entry to determine if the PRODUCT entry in the originating global zone is at a higher level than the same PRODUCT entry in the destination zone. If the rework date of the PRODUCT entry in the originating global zone is more recent than the REWORK date of the PRODUCT entry in the destination global zone, then that PRODUCT entry is considered to be at a higher level in the originating global zone and would be merged.

Merging DEFINITION entries

This processing occurs when the DEFINITION operand is specified on the GZONEMERGE command.

For each definition entry in the originating zone, SMP/E checks to see whether that entry exists in the destination zone.

- If the same-named definition entry is not found, the entry from the originating global zone is stored in the destination global zone.
- If the same-named definition entry is found, then processing continues based on the type of definition entry.
- For DDDEF, FMIDSET, OPTIONS, ORDER, UTILITY, and ZONESET entries, if the originating global zone contains an entry that does not appear in the destination global zone, then the entry from the originating global zone is stored in the destination global zone. (The DDDEF entries for the SMPPTS and its spill data sets are not merged by GZONEMERGE processing.)

- For the GLOBALZONE entry, processing continues to determine which subentries within the GLOBALZONE entry to merge.

Determining GLOBALZONE subentries to merge

SMP/E builds a new GLOBALZONE entry if the destination global zone does not contain one. The specific subentries that appear in the GLOBALZONE entry depends on what combination of operands were specified on the GZONEMERGE command.

SMP/E will modify an existing GLOBALZONE entry if the destination global zone already contains subentries. The specific subentries that are modified in the GLOBALZONE entry depends on what combination of operands were specified on the GZONEMERGE command.

Merging the GZONE SREL and FMID subentries

The GZONE SREL and FMID subentries are merged as follows:

When the user specifies the FORFMID operand in combination with the DEFINITION operand, processing for the SREL and FMID subentries are as described in "GLOBALZONE entry updates for content entries" on page 145.

When the user specifies the CONTENT operand in combination with the DEFINITION operand, processing for the SREL and FMID subentries are as described in "GLOBALZONE entry updates for content entries" on page 145.

When the user specifies only the DEFINITION operand, the SREL and FMID subentries in the originating global zone are compared to the SREL and FMID subentries in the destination zone. Any SRELs or FMIDs that exist in the originating global zone's SREL or FMID subentry that are not listed in the destination global zone's SREL or FMID subentry are added to the destination global zone.

Merging the GZONE OPTIONS subentry

The GZONE OPTIONS subentry is merged as follows:

If no OPTIONS subentry exists in the destination global zone, then the OPTIONS subentry from the originating global zone is added to the destination global zone. If an OPTIONS subentry exists in the destination global zone, then the OPTIONS subentry from the originating global zone is not merged.

Merging the GZONE ZONEDESCRIPTION subentry

The GZONE ZONEDESCRIPTION subentry is merged as follows:

If no ZONEDESCRIPTION subentry exists in the destination global zone, then the ZONEDESCRIPTION subentry from the originating global zone is added to the destination global zone. If an ZONEDESCRIPTION subentry exists in the destination global zone, then the ZONEDESCRIPTION subentry from the originating global zone is not merged.

Merging the GZONE ZONEINDEX subentry

The GZONE ZONEINDEX subentry consists of three fields for each zone index:

- *name* is the name of the zone.
- *dsn* is the fully qualified name of the data set in which the zone resides.
- *type* is the zone type, either TARGET or DLIB.

SMP/E uses the name field of each zone index to determine whether or not to merge the particular component of the subentry.

If the same zone name exists in both the originating and destination global zones, then that zone index is not merged.

If the zone name only exists in the originating global zone, then that zone index is merged. When a zone index is merged, SMP/E copies all three fields (name,dsn,type) associated with that zone index to the destination global zone.

The GZONEMERGE command does not overlay any existing zone indices in the destination global zone.

If no ZONEINDEX subentry exists in the destination global zone, then the ZONEINDEX subentry from the originating global zone is added to the destination global zone.

Merging ORDER entries

The ORDER entry is merged as follows:

The ORDERID subentry value of an ORDER entry is a unique identifier for an order assigned by the server. If the ORDERID subentry values of two ORDER entries are the same, then the ORDER entries represent the same order on the server. Therefore, if any ORDER entry in the destination global zone has an ORDERID subentry value that matches the ORDERID subentry value of the ORDER entry from the originating global zone, the entry will not be copied during processing for this GZONEMERGE command (most likely such an ORDER entry was copied during a previous GZONEMERGE command).

If an ORDER entry from the originating global zone has a unique ORDERID subentry value, then SMP/E will check to see if an ORDER entry with the same name already exists in the destination zone. If not, then the ORDER entry from the originating zone is stored in the destination zone. If a same-named entry does exist, SMP/E will attempt to generate a new entry name as is done when a new ORDER entry is created. If an ORDER entry name can be generated which is unique in the destination global zone, then SMP/E copies the entry to the destination zone using the new entry name. If a unique ORDER entry name cannot be generated, the entry is not copied to the destination zone.

Compaction of inline data

The GZONEMERGE command automatically compacts inline data within SYSMODs when copying members to an SMPPTS data set whenever the COMPACT subentry in the active OPTIONS entry indicates compaction is to be performed (this is the default) and the driving system supports compression and expansion services. Otherwise, the GZONEMERGE command does not compact SMPPTS members.

When SMPPTS members are compacted, the modification control statements, inline JCLIN data, and inline ZAP data remain in their original, uncompacted state. All other inline data associated with each of the following modification control statements are compacted before being written to the SMPPTS data set member:

All data elements
All hierarchical file system elements
++JAR
++JARUPD
++MAC
++MACUPD
++MOD
++PROGRAM
++SRC
++SRCUPD

Zone and data set sharing considerations

The following identifies the phases of GZONEMERGE processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Destination Global zone
   Read without enqueue.

   Originating Global zone
   Read without enqueue.

2. Processing

   Destination Global zone
   Update with exclusive enqueue.

   Destination SMPPTS
   Update with exclusive enqueue.

   Originating Global zone
   Read with shared enqueue.

   Originating SMPPTS
   Read with shared enqueue.

3. Termination

   All resources are freed.

Chapter 9. The JCLIN command

JCLIN processing initializes entries in the target or distribution zone with the data required to install elements into the target libraries. SMP/E derives this data by scanning a job stream containing assembly, copy, and link-edit job steps. JCLIN processing is called by either the JCLIN command or as part of installing a SYSMOD containing ++JCLIN statements.

Note that SMP/E does **not** execute the JCLIN. It does not call the utilities specified in the job stream, and it does not update, modify, or look at any of the target libraries. Rather, JCLIN is a vehicle to describe the structure of the target libraries (and, ultimately, the structure of the load modules in those libraries). It is element statements (such as ++MOD) that cause SMP/E to actually invoke the utilities.

The purpose of this chapter is to describe JCLIN processing and the requirements SMP/E has on the input, so that you can better understand how to construct JCLIN for your own use.

Note:

1. JCLIN processes only information about elements such as macros, modules, and source. It does not process information about data elements, except for totally copied libraries. Information about data elements is added to the zone when the data elements are installed in the libraries or when the distribution zone is copied into the target zone.
2. JCLIN processing does **not** cause SMP/E to update the target or distribution libraries; only the entries in the target and distribution zones are updated. These libraries are updated when SMP/E processes the elements in a SYSMOD. The element statements in a SYSMOD determine which elements should be installed.

The following terms are used in this chapter:

JCLIN input
A job stream of assembly, link-edit, and copy job steps. This input is pointed to by the SMPJCLIN DD statement.

Inline JCLIN
JCLIN data pointed to by a ++JCLIN MCS. The actual JCLIN data can be packaged inline, following the ++JCLIN MCS, or it can be packaged in a RELFILE or TXLIB data set.

The description of the JCLIN command used here applies to processing both the JCLIN command and to inline JCLIN processed at APPLY or ACCEPT time. The utility and OPCODE operands that can be specified on the JCLIN command are also valid as operands on the ++JCLIN statement.

Reminder

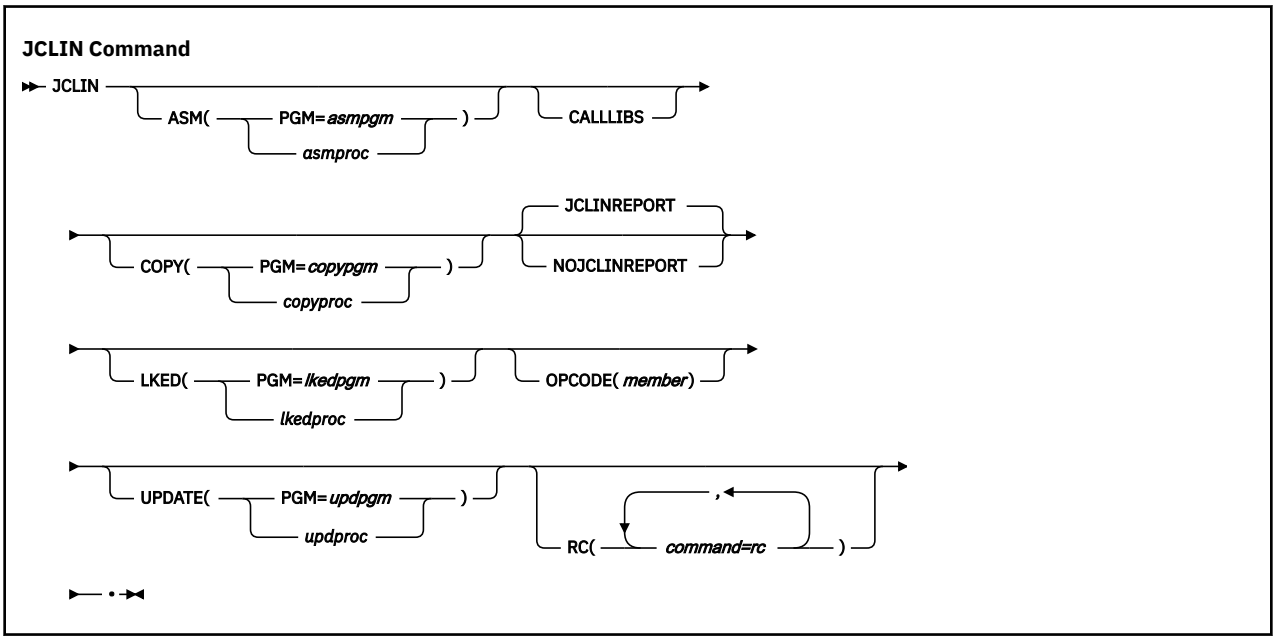
The input for JCLIN must be free of JCL errors and must conform to the SMP/E restrictions. Incorrect input can cause unpredictable errors, or even cause the installation of SYSMODs that depend on the JCLIN to fail.

Zone for SET BOUNDARY

The zone specified on the SET BOUNDARY command depends on the type of JCLIN being processed.

- For the JCLIN command, the SET BOUNDARY command must specify the name of the target zone to be updated.
- For inline JCLIN processed by the APPLY command, the SET BOUNDARY command must specify the name of the target zone to be updated.
- For inline JCLIN processed by the ACCEPT command, the SET BOUNDARY command must specify the name of the distribution zone to be updated, and the ACCJCLIN indicator must be set in the DLIBZONE entry. For more information, see “Inline JCLIN” on page 36.

Syntax



Operands

ASM
specifies an assembler program, *asmpgm*, or procedure, *asmproc*, in addition to those recognized by SMP/E. SMP/E recognizes programs ASMBLR, ASMA90, IEUASM, IEV90, and IFOX00, as well as procedure ASMS.

CALLLIBS
specifies that SMP/E is to process SYSLIB DD statements in JCLIN link-edit steps. If the CALLLIBS operand is not specified on either the JCLIN command or the ++JCLIN statement, SMP/E JCLIN processing ignores any SYSLIB DD statements it encounters.

COPY
specifies a copy program, *copypgm*, or procedure, *copyproc*, in addition to those recognized by SMP/E. SMP/E recognizes only the IEBCOPY program.

JCLINREPORT
specifies that SMP/E is to write the JCLIN reports. This is the default.

Note: `JCLINREPORT` can also be specified as `JCLR`.

LKED
specifies a link-edit utility program, *lkedpgm*, or procedure, *lkedproc*, in addition to those recognized by SMP/E. SMP/E recognizes programs IEWBLINK, IEWL, HEWL, HEWLH096, HEWLKED, and LINKEDIT, as well as procedure LINKS.

NOJCLINREPORT
specifies that SMP/E is not to write any JCLIN reports.

Note: `NOJCLINREPORT` can also be specified as `NOJCLR`.

OPCODE
identifies an OPCODE member (which must be defined in the SMPPARM data set) containing control cards identifying character strings to be treated as operation codes or macros.

SMP/E contains a default set of OPCODE definitions for your use. If you do not want to use this default set, you can define your own by using the sample GIMOPCDE member supplied to you.

For more information about OPCODE members and how SMP/E determines whether an assembler instruction is an OPCODE or a macro, see "Building MAC entries" on page 173 and the OPCODE control statements section in *z/OS SMP/E Reference*.

PGM
specifies a program name (instead of a procedure name) for a utility.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the JCLIN command.

Before SMP/E processes the JCLIN command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the JCLIN command. Otherwise, the JCLIN command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the JCLIN command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

UPDATE
specifies an update program, *updpgm*, or procedure, *updproc*, in addition to those recognized by SMP/E. SMP/E recognizes only program IEBUPDTE.

Note: Although JCLIN does not actually derive any target zone 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.

Data sets used

These data sets might be needed to run the JCLIN command. You can define them by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPOUT	SMPCSI	SMPPARM
SMPJCLIN	SMPSNAP	SMPLOGA	*zone*
SMPLOG			

Note:

1. The SMPPARM DD statement is required only if the JCLIN input contains assembly steps and if you want to create OPCODE members to override or amend the default set of OPCODE definitions.
2. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are allocated dynamically by use of the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

This section provides usage notes for the JCLIN command.

JCLIN input

SMP/E comment statements in JCLIN input

SMP/E comment statements are special control statements that are coded as JCL comments. These comment statements are used to convey information to SMP/E during JCLIN processing.

The types of comment statements are:

1. CALLLIBS

SMP/E inserts a CALLLIBS comment statement (//*CALLLIBS=YES) immediately after the LKSYSLIB job to ensure that when the JCLIN command processes the GENERATE output, the SYSLIB DD statements in the output are processed and not ignored. See "Load modules installed in libraries specified in a SYSLIB allocation" on page 138 for information on when the LKSYSLIB job is generated.

2. Conditional JCLIN

   Conditional JCLIN comment statements are used to cause SMP/E to skip parts of the JCLIN input stream. See "Conditional JCLIN comment statements" on page 156 for a description of conditional JCLIN comment statements.

3. LIBRARYDD

   The LIBRARYDD comment statement is used to associate a ddname with a UNIX pathname. See "Example 7: JCLIN for load modules residing in a UNIX file system" on page 167 for an example of how the LIBRARYDD comment statement is used.

4. SMPLTS

   SMP/E inserts an SMPLTS comment statement (//*SMPLTS) immediately after the SMPLTS job card to ensure that when the JCLIN command processes the GENERATE output, SMP/E can detect the SMPLTS job and skip over it. See "Load modules installed in libraries specified in a SYSLIB allocation" on page 138 for information on when the SMPLTS job is generated.

Conditional JCLIN comment statements

SMP/E allows a packager to provide multiple structural definitions within a single JCLIN input stream. SMP/E can then skip parts of the JCLIN input stream, based on control statements placed in the stream by the packager. The parts of the JCLIN input that are skipped are ignored by SMP/E and therefore do not contribute to the structure information derived from JCLIN processing. One use of this function is to ensure that load module information altered by a feature of a product is not downleveled by JCLIN input provided by base product service.

The function described in this section applies to processing by the JCLIN command, which processes JCLIN input from the SMPJCLIN DD statement, and to the processing of JCLIN input within a SYSMOD by the APPLY and ACCEPT commands.

The conditional JCLIN comment statements are:

1. `//*SMPE-IF SYSMOD(`*sysmod_id*`) THEN DO`
2. `//*SMPE-ELSE DO`
3. `//*SMPE-END`

The `//*SMPE-IF SYSMOD(`*sysmod_id*`) THEN DO` (or `SMPE-IF` for short) comment statement specifies a *sysmod_id* that is to have checks performed against it and a `DO` that starts a `DO/END` grouping of records within the JCLIN input stream. When SMP/E JCLIN processing encounters the `SMPE-IF` control statement, the *sysmod_id* value is saved for later checking. Additionally, SMP/E notes that a `DO` group has been initiated so it can match the current `DO` to its `SMPE-END` control statement.

When the `SMPE-IF` comment statement is within a section of the JCLIN input stream that is being fully processed (that is, not being skipped), some checks are performed on the *sysmod_id* specified by the `SYSMOD` keyword to determine if subsequent records in the JCLIN input stream should be fully processed or should be skipped until the `SMPE-END` for the `DO/END` group is found. The checks on the specified *sysmod_id* value are not performed when the `SMPE-IF` comment statement is encountered as part of the JCLIN input stream that is being skipped.

- If the specified *sysmod_id* is
  - a superseded only SYSMOD in the current target or distribution zone, or
  - a SYSMOD that is in process and is not currently marked NOGO (during APPLY and ACCEPT processing), or
  - a real SYSMOD entry with the ERROR indicator OFF in the current target or distribution zone,

SMP/E reads and fully processes subsequent records until the SMPE-END comment statement that matches the current DO (that is, the one on this SMPE-IF comment statement) is found.

- If the specified *sysmod_id*
  - is a SYSMOD that is in process that has been terminated (that is, it is currently marked NOGO) during APPLY and ACCEPT processing and either
    - has no real SYSMOD entry in the current target or distribution zone -OR-
    - has a real SYSMOD entry in the current target or distribution zone with the ERROR indicator ON

    or
  - is not in process and has a real SYSMOD entry with the ERROR indicator ON in the current target or distribution zone,

  SMP/E terminates JCLIN processing (along with the containing SYSMOD for APPLY and ACCEPT processing), because having the specified *sysmod_id* in error leaves the content of the current zone in question.
- If the specified *sysmod_id*
  - is being deleted during the current APPLY or ACCEPT command and is not in ERROR, or
  - exists as a DELETED SYSMOD entry in the current target or distribution zone, or
  - does not exist at all in the current target or distribution zone or as a SYSMOD currently being installed.

  SMP/E skips any records in the JCLIN input stream that are contained in the DO/END group of the current SMPE-IF comment statement.

The SMPE-IF comment statement must adhere to the following syntax rules:

- The IF must immediately follow the //*SMPE- on the JCL comment with no intervening blanks.
- One or more blanks must separate the IF from the SYSMOD(*sysmod_id*) specification.
- The SYSMOD(*sysmod_id*) specification
  - may have one or more blanks between SYSMOD and the opening parenthesis (), although none are required,
  - may have one or more blanks between the opening parenthesis () and the *sysmod_id*, although none are required,
  - may have one or more blanks between the *sysmod_id* and the closing parenthesis ()), although none are required,

  The *sysmod_id* must contain seven uppercase alphanumeric characters.
- One or more blanks must separate the SYSMOD(*sysmod_id*) specification from the THEN.
- One or more blanks must separate the THEN from the DO.
- The whole SMPE-IF comment statement must be on a single record and cannot extend past column 71.
- Conditional JCLIN comment statements cannot appear within the utility control card section of a JCLIN job step.

The SMPE-ELSE DO clause of an SMPE-IF control statement is optional. It is used to group together a portion of the JCLIN input stream that is to be fully processed only if the checks against the *sysmod_id* specified on the associated SMPE-IF clause cause the DO/END group of the SMPE-IF to be skipped. Conversely, the DO/END group of the SMPE-ELSE clause is skipped if the DO/END group of the associated SMPE-IF clause is fully processed.

The SMPE-ELSE comment statement must immediately follow the SMPE-END comment statement that ends the DO/END group of the associated SMPE-IF clause. If an SMPE-ELSE comment statement is encountered that does not immediately follow the SMPE-END comment statement that ends the DO/END group of the associated SMPE-IF clause, SMP/E terminates JCLIN processing (along with the containing SYSMOD for APPLY and ACCEPT processing).

The ELSE part of this comment statement must immediately follow the //*SMPE- on the JCL comment with no intervening blanks. One or more blanks must separate the DO portion of this comment statement from the ELSE. The whole SMPE-ELSE comment statement must be on a single record and cannot extend past column 71.

The SMPE-END comment statement ends a DO group started on either an SMPE-IF comment statement or an SMPE-ELSE comment statement. The SMPE-END comment statement is not optional; it must be used to end the DO group.

The END portion of this control statement must immediately follow the //*SMPE- on the JCL comment with no intervening blanks.

If an SMPE-END comment statement is encountered that does not close an open DO from an SMPE-IF or SMPE-ELSE comment statement, SMP/E terminates JCLIN processing (along with the containing SYSMOD for APPLY and ACCEPT processing).

Note that a DO/END group may be empty. This means that there might be no records in the JCLIN input stream between the SMPE-IF or SMPE-ELSE comment statement that starts the DO/END group and the SMPE-END comment statement that ends the DO/END group.

Once SMP/E has parsed one of the conditional JCLIN comment statements, SMP/E ignores the remainder of the record that contains the JCLIN comment statement. This means that users can follow the JCLIN comment statement with one or more blanks and then use the remainder of the record for their own comments.

JCLIN processing works its way through the JCLIN input stream from the first record in the stream to the last record in the stream. Once a record has been processed, it cannot be processed again.

While skipping records in the JCLIN input stream, JCLIN processing still syntax checks the conditional JCLIN comment statements that may exist in the skipped section of the JCLIN input stream. Additionally, DO/END groups and IF THEN/ELSE clauses are matched up in the skipped section of the JCLIN input stream. If an error is detected when doing this syntax checking and matching, SMP/E terminates JCLIN processing. Once a record being skipped is found not to contain one of the SMP/E comment statements, the record is not processed any further.

If SMP/E does not find an SMPE-END comment statement to close an open DO/END group before end-of-file is reached on the JCLIN input stream, SMP/E terminates JCLIN processing.

Example of using conditional JCLIN comment statements

Given these facts:

- A base release of the BCP is shipped and has FMID HBB2900.
- The inline JCLIN for HBB2900 defines a load module, LMOD001, as follows:

```
//LINK1 EXEC PGM=IEWBLINK,PARM='LIST,RENT'
//*
//* HBB2900 DEFINITION OF LMOD001
//*
//DLIB1 DD DSN=SYS1.DLIB1,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
   ORDER PART001,PART002,PART003
   INCLUDE DLIB1(PART001)
   INCLUDE DLIB1(PART002)
   INCLUDE DLIB1(PART003)
   ENTRY PART001
   NAME LMOD001(R)
```

- A point release for this level of the BCP is shipped and has FMID JBB2901. This feature adds one new module to LMOD001 (PART004) that makes the load module serially reusable, instead of reentrant, as it was before. The other modules in LMOD001 (PART001, PART002, and PART003) are still owned by HBB2900. JBB2901's JCLIN for LMOD001 is as follows:

```
//LINK1 EXEC PGM=IEWBLINK,PARM='LIST,REUS'
//*
//* JBB2901 DEFINITION OF LMOD001
```

```
//*
//DLIB1 DD DSN=SYS1.DLIB1,DISP=SHR
//DLIB2 DD DSN=SYS1.DLIB2,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
   ORDER PART001,PART002,PART003,PART004
   INCLUDE DLIB1(PART001)
   INCLUDE DLIB1(PART002)
   INCLUDE DLIB1(PART003)
   INCLUDE DLIB2(PART004)
   ENTRY PART001
   NAME LMOD001(R)
```

- After JBB2901 has been released to the field, service is shipped that affects module PART002 such that the module has to be split into two modules, PART002 and PART002B. A PTF against the base, HBB2900, must be shipped that includes these two modules and a redefinition of LMOD001.

  Using conditional JCLIN comment statements, the definition of LMOD001 can be altered to take into account the splitting of module PART002 such that both definitions (base and feature) can be packaged in the same base PTF. The MCS and inline JCLIN for LMOD001 in the base PTF would look something like this:

```
++PTF(PTFB001).
++VER(Z038) FMID(HBB2900) SUP(…) PRE(…).
++IF FMID(JBB2901) THEN REQ(PTFF001).
++MOD(PART002) DISTLIB(DLIB1).
  -- inline object for PART002 --
++MOD(PART002B) DISTLIB(DLIB1).
  -- inline object for PART002B --
++JCLIN.
//ANYNAME  JOB
//*SMPE-IF SYSMOD(JBB2901) THEN DO  -- JBB2901 FLAVOR OF LMOD001
//LINK1 EXEC PGM=IEWBLINK,PARM='LIST,REUS'
//*
//* JBB2901 LEVEL FOR BIND OF LMOD001
//*
//DLIB1 DD DSN=SYS1.DLIB1,DISP=SHR
//DLIB2 DD DSN=SYS1.DLIB2,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
   ORDER PART001,PART002,PART002B,PART003,PART004
   INCLUDE DLIB1(PART001)
   INCLUDE DLIB1(PART002)
   INCLUDE DLIB1(PART002B)
   INCLUDE DLIB1(PART003)
   INCLUDE DLIB2(PART004)
   ENTRY PART001
   NAME LMOD001(R)
//*SMPE-END
//*SMPE-ELSE DO                     -- HBB2900 FLAVOR OF LMOD001
//LINK1 EXEC PGM=IEWBLINK,PARM='LIST,RENT'
//*
//* HBB2900 LEVEL FOR BIND OF LMOD001
//*
//DLIB1 DD DSN=SYS1.DLIB1,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
   ORDER PART001,PART002,PART002B,PART003
   INCLUDE DLIB1(PART001)
   INCLUDE DLIB1(PART002)
   INCLUDE DLIB1(PART002B)
   INCLUDE DLIB1(PART003)
   ENTRY PART001
   NAME LMOD001(R)
//*SMPE-END
```

  SMP/E is able to process this JCLIN input for the environment that has just HBB2900 installed and for the environment that has both HBB2900 and JBB2901 installed such that the definition for load module LMOD001 is appropriate for each environment.

  When the base PTF, PTFB001, is shipped, a corresponding PTF must be shipped for the feature that contains the updated JCLIN for the feature's definition of LMOD001. The feature PTF is needed so that, if the feature is installed after the base PTF has been installed, the definition of LMOD001 is not downleveled by the installation of the feature, because the feature PTF will be required to be installed along with the feature.

The requisite PTF for the feature, PTFF001, would look something like the following example::

```
++PTF(PTFF001).
++VER(Z038) FMID(JBB2901) SUP(…) PRE(PTFB001, … ).
++JCLIN.
//ANYNAME  JOB
//LINK1 EXEC PGM=IEWBLINK,PARM='LIST,REUS'
//*
//* JBB2901 LEVEL FOR BIND OF LMOD001
//*
//DLIB1 DD DSN=SYS1.DLIB1,DISP=SHR
//DLIB2 DD DSN=SYS1.DLIB2,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
   ORDER PART001,PART002,PART002B,PART003,PART004
   INCLUDE DLIB1(PART001)
   INCLUDE DLIB1(PART002)
   INCLUDE DLIB1(PART002B)
   INCLUDE DLIB1(PART003)
   INCLUDE DLIB2(PART004)
   ENTRY PART001
   NAME LMOD001(R)
```

Packaging JCLIN input

Once JCLIN has been processed in the target zone by the SMP/E JCLIN command, it cannot be removed. Therefore, if there is any chance that you will want to remove the JCLIN changes, you should not use the JCLIN command in processing the input. Instead, you should package the JCLIN input as part of a SYSMOD, and then receive and apply the SYSMOD. The following is an example of such a SYSMOD:

```
++USERMOD(USR0001).
++VER(Z038) FMID(USRFUNC).
++JCLIN.
//…
//…
//… JCLIN input
//…
//…
```

These are the commands to install it:

```
SET      BDY(GLOBAL)        /* Set to global zone.        */.
RECEIVE  SYSMODS            /* Receive the SYSMOD.        */
         S(USR0001)         /*                            */.
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
APPLY    S(USR0001)         /* Apply it.                  */.
```

When you use this method, any target zone entries that are changed as a result of the new JCLIN are first saved on the SMPSCDS data set. The SYSMOD, USR0001, can then be restored if you want to remove the JCLIN.

Note: After doing a full system generation, you must:

- Delete your old target zone and define a new one.
- Use ZONECOPY to copy the distribution zone to the new target zone.
- Scratch and reallocate your SMPMTS and SMPSTS data sets.

If you do not start your new system with an empty SMPMTS and SMPSTS, you may inadvertently regress your new system when service is applied.

SYSMODs with inline JCLIN

A SYSMOD can be constructed with the JCLIN necessary for it to be installed. In this case, the SYSMOD contains a ++JCLIN statement. When processing such a SYSMOD, SMP/E processes the JCLIN as part of the installation of the SYSMOD. During APPLY processing, and before changing any target zone entry affected by the JCLIN, a copy of that entry is stored on the SMPSCDS data set. This data is used by SMP/E in case the SYSMOD has to be restored. The target zone can then be brought back to the level it was at before the apply was run.

When applying multiple SYSMODs, each containing JCLIN, the JCLIN from any superseded SYSMODs is not processed.

Note: Each target zone must have its own SMPSCDS data set, because the information in the SMPSCDS data set is unique to that target zone. This SMPSCDS data set must also be used with the related distribution zone.

OPCODE members to identify opcodes in assembler text

SMP/E uses information stored in an OPCODE definition default set to differentiate macro invocations from assembler operation codes. If you wish to change the default set, the SMPPARM DD statement is required when SMP/E processes JCLIN with assembly steps.

Licensed programs or user products that use special assemblers recognizing additional operation codes must provide an OPCODE member identifying those codes, and the name of that OPCODE member must be specified in the OPCODE operand. For more information about defining OPCODE members for JCLIN processing, see the "SMP/E OPCODE Member Control Statements" topic in *z/OS SMP/E Reference*.

Processing after system generation

After a complete system generation, you must copy the distribution zone, using an SMP/E zone utility command (such as ZONECOPY, ZONEEXPORT and ZONEIMPORT, or ZONEMERGE), to the new target zone before doing any JCLIN processing. This makes sure the initial target zone entries match those of the distribution zone and contain entries not created by JCLIN processing.

After a partial system generation (that is, a device generation), the output of stage 1 must be used as input to JCLIN processing to make sure that:

- Module, macro, and load module entries in the target zone are updated.
- New assembler entries are stored with the new assembler input in the target zone.
- Link-edit utility control statements for load module entries are automatically replaced, except for link-edit utility CHANGE and REPLACE control statements.

  CHANGE and REPLACE control statements are associated with the DLIB module name found on the next INCLUDE statement in the link-edit job.

  – If the same INCLUDE statement is contained in the stage 1 output, the CHANGE and REPLACE control statements are replaced.
  – If that INCLUDE statement is **not** contained in the stage 1 output, the CHANGE and REPLACE control statements are carried over to the updated entry.

Cross-zone relationships

If SMP/E creates an LMOD entry during JCLIN processing and there is already a copy of that entry containing only cross-zone subentries (a *stub* entry), SMP/E issues messages indicating that the cross-zone relationship defined by cross-zone subentries might no longer be valid. SMP/E then deletes the cross-zone subentries from the LMOD entry. If there is no longer a TIEDTO relationship between the cross-zone and the set-to zone, SMP/E also deletes the TIEDTO value for the cross-zone from the zone definition entry for the set-to zone. It does not, however, update related cross-zone modules to indicate that they are no longer part of the load module in the set-to zone or the cross-zone's TIEDTO subentry. You need to determine whether the cross-zone relationship is still valid. If it is valid, use the LINK MODULE command to reestablish it. For more information, see Chapter 11, “The LINK MODULE command,” on page 197 .

Output

The following reports are produced during JCLIN processing:

- File Allocation report
- JCLIN Cross-Reference report

- JCLIN Summary report

For descriptions of these reports, see Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the JCLIN command.

Example 1: JCLIN for products with special utilities

Given the following JCLIN input:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=MYCOPY
//XYZDLIB  DD DSN=SYS1.XYZDLIB,DISP=SHR
//XYZLOAD  DD DSN=SYS1.XYZLOAD,DISP=SHR
//SYSIN    DD *
 COPY INDD=XYZDLIB,OUTDD=XYZLOAD
/*
```

SMP/E processes it but does not recognize MYCOPY as a valid program name; therefore, it creates no entries.

But given the following JCLIN command:

```
SET      BDY(XYZTST1)       /* Process zone XYZTGT1.    */.
JCLIN    COPY(PGM=MYCOPY)   /* Process JCLIN.           */.
```

SMP/E now recognizes the step as a copy step and builds a DLIB entry for XYZDLIB.

Example 2: JCLIN for products with special assembler opcodes

Assume that you have a special version of the assembler supporting two additional operation codes, LOOP and ENDLOOP, and that the following JCLIN is to be processed:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP     EXEC PGM=MYASM
//SYSPUNCH DD DSN=&&PUNCH(NEWMOD),;
//         SPACE=(TRK,(1,1,1)),DISP=(,PASS)
//SYSIN    DD *
NEWMOD     CSECT
           .
           .
LOOPSTRT   LOOP
           .
           .
LOOPEND    ENDLOOP
           .
           .
           END NEWMOD
```

To process this correctly and have SMP/E recognize that LOOP and ENDLOOP are opcodes, you should set up the following OPCODE member (in this case, named SMPPRM01):

```
KEY=LOOP    TYPE=OPCODE.
KEY=ENDLOOP TYPE=OPCODE.
```

The command to execute the JCLIN should then be as follows:

```
SET      BDY(XYZTST1)       /* Process zone XYZTST1.    */.
JCLIN    ASM(MYASM)         /* Special assembler.       */
         OPCODE(SMPPRM01)   /* Special opcode list.     */.
```

When SMP/E processes this input, it recognizes that MYASM is an assembler program, and that LOOP and ENDLOOP are operation codes.

Example 3: JCLIN for MOD entries

Given the job steps shown in this example, SMP/E creates target zone entries as follows:

```
//C1       EXEC PGM=IEBCOPY
//AMACLIB  DD DSN=SYS1.AMACLIB,DISP=SHR
//MACLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//SYSIN DD *
 COPY INDD=AMACLIB,OUTDD=MACLIB   TYPE=MAC
/*
//C2       EXEC PGM=IEBCOPY
//AOS14    DD DSN=SYS1.AOS14,DISP=SHR
//LINKLIB  DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSIN DD *
 COPY INDD=AOS14,OUTDD=LINKLIB  TYPE=MOD
 SELECT MEMBER=((LOADMODC,,R))
/*
```

```
//DEFINE   JOB 'accounting info',MSGLEVEL=(1,1)
//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) RC=4
/*
```

```
//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)
/*
```

```
//L3       EXEC PGM=IEWBLINK,PARM='OL,AMODE=31,OPTIONS(OPTNAME)'
//SYSLMOD  DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//OPTNAME  DD *
 FETCHOPT(PACK,PRIME),RMODE=24
 MAXBLK(256)
/*
//SYSLIN DD *
 INCLUDE AOS12(GIMMPDRV,GIMMPDR1)
 ENTRY GIMMPDRV
 SETCODE AC(1)
 NAME GIMMPP(R)
/*
```

```
//L4       EXEC PGM=IEWBLINK,PARM='CALL,RENT,REUS'
//SYSLMOD  DD DSN=SYS1.APPLOAD,DISP=SHR
//AOS12    DD DSN=SYS1.AOS12,DISP=SHR
//SYSLIB   DD DSN=SYS1.V2R2M0.PLIBASE,DISP=SHR
//         DD DSN=SYS1.V2R2M0.APPBASE,DISP=SHR
//SYSLIN DD *
 INCLUDE AOS12(MOD00004,MOD00005)
 ENTRY MOD00004
 SETCODE AC(1)
 NAME LMOD04(R)
/*
```

Copy step C1

This copy step informs SMP/E that an entire distribution library has been copied to an operating system library. From the INDD operand, SMP/E determines the ddname of the distribution library and creates a target zone DLIB entry named AMACLIB. From the OUTDD operand, SMP/E determines the ddname of the operating system library and adds a SYSLIB subentry, MACLIB, to the DLIB entry. SMP/E uses this entry to determine the operating system library for subsequent modifications that 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/E assumes that the elements involved are **modules** and creates a target zone MOD entry for each of them. 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 step C2, a MOD entry named LOADMODC is created; the distribution library is determined to be AOS14, and the LMOD is LOADMODC.

Finally, a target zone LMOD entry is created to represent the operating system library to which the module is copied. The load module name (and the target zone LMOD entry name) is the module name, LOADMODC. The operating system library (saved as a SYSLIB subentry) is determined from the operand of the copy OUTDD operand—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/E that the link-edit attributes must be obtained by examining the operating system library when the module must first be link-edited.

Assembly step A1

SMP/E creates an ASSEM entry with the name TESTMOD1, derived from the member name on the SYSPUNCH DD statement. This entry contains the assembler SYSIN data set.

A MAC entry named TESTMAC1 is created, because SMP/E detects the invocation of the macro in the assembler SYSIN. The macro entry created contains the name of the assembly as a GENASM subentry (for use by SMP/E in determining 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/E derives the data required to create a target zone MOD entry representing the module, TESTMOD1. The module name is determined from the member name operand on the INCLUDE statement, 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 target zone LMOD entry is created from the link-edit NAME statement. It 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, the load module's highest acceptable link edit return code value is set to 4, and the system library (saved as a SYSLIB subentry) 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 and LIST are not saved.

Link-edit step L2

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

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 target zone MOD entry is created for each of these modules with a LMOD subentry naming the load module, LOADMODX, and a DLIB subentry, AOS12.

The operating system load module, LOADMODX, is represented by a target zone LMOD entry. This LMOD entry, as created, contains the operating system library, LINKLIB, as a SYSLIB subentry. No link-edit **attributes** are specified in this step; therefore, the STD indicator is set in the LMOD entry.

Link-edit step L3

The third link-edit step shows an example of using the OPTIONS option. The OPTNAME DD statement allows SMP/E to process the PARM string, even though the options exceed the 100-character limit.

Link-edit step L4

The fourth link-edit step shows an example containing a SYSLIB allocation. If CALLLIBS was specified on the JCLIN command or the ++JCLIN MCS, the low-level qualifiers of the data sets named in the SYSLIB allocation, PLIBASE and APPBASE, are saved as part of the CALLLIBS subentry list in LMOD entry LMOD04.

Example 4: JCLIN for MAC and SRC entries

Given the following JCLIN input:

```
//JOB      JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IEBCOPY
//MACDLIB  DD DSN=SYS1.MACDLIB,DISP=SHR
//MACTGT   DD DSN=SYS1.MACTGT,DISP=SHR
//SRCDLIB  DD DSN=SYS1.SRCDLIB,DISP=SHR
//SRCTGT   DD DSN=SYS1.SRCTGT,DISP=SHR
//SYSIN    DD *
 COPY INDD=MACDLIB,OUTDD=MACTGT    TYPE=MAC
 S    M=(MAC01,MAC02,MAC03)
 S    M=(MAC04,MAC05)
 COPY INDD=SRCDLIB,OUTDD=SRCTGT    TYPE=SRC
 S    M=(SRC01,SRC02,SRC03)
 S    M=(SRC04,SRC05)
/*
```

SMP/E creates the entries shown in Table 12 on page 165:

Table 12. MAC and SRC entries created by JCLIN

Entry type	Entry name	DISTLIB	SYSLIB
MAC	MAC01	MACDLIB	MACTGT
MAC	MAC02	MACDLIB	MACTGT
MAC	MAC03	MACDLIB	MACTGT
MAC	MAC04	MACDLIB	MACTGT
MAC	MAC05	MACDLIB	MACTGT
SRC	SRC01	SRCDLIB	SRCTGT
SRC	SRC02	SRCDLIB	SRCTGT
SRC	SRC03	SRCDLIB	SRCTGT
SRC	SRC04	SRCDLIB	SRCTGT
SRC	SRC05	SRCDLIB	SRCTGT

Example 5: JCLIN for an assembler step to create a SRC entry

Given the following JCLIN input:

```
//JOB       JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1     EXEC PGM=IEUASM
//SYSLIB    DD   DSN=SYS1.MACLIB,DISP=SHR
```

```
//SYSPUNCH  DD   DSN=&&PUNCH(SRCA),DISP=SHR;
//SYSIN     DD   DSN=SYS1.ASRCLIB(SRCA),DISP=SHR
```

SMP/E creates a SRC entry named SRCA whose DISTLIB is ASRCLIB.

Example 6: JCLIN for using the link-edit automatic library call function

SMP/E provides support for load modules that need to use the link-edit automatic library call function, which enables the load modules to contain modules from multiple products without explicitly specifying those modules on INCLUDE statements in link-edit steps. SMP/E's support for load modules that use the link-edit automatic library call function is called *CALLLIBS support*.

Overview of CALLLIBS support

SMP/E's CALLLIBS support uses the link-edit CALL parameter and a SYSLIB allocation when invoking the link-edit utility to resolve external references in load modules. CALLLIBS support can be useful for a variety of products, including those that:

- Are written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) that are owned by a different product
- Make use of a callable-services interface provided by another product
- Need to include stub routines or interface modules from different products that may reside in other zones

To package a load module that needs to use the automatic library call function, follow these steps:

1. Specify the CALLLIBS operand on the ++JCLIN MCS. CALLLIBS tells SMP/E to:
   - Save the SYSLIB allocation defined by the JCLIN link-edit step in the LMOD entry for the load module. This information is recorded in the CALLLIBS subentry list.
   - Pass the SYSLIB allocation and the CALL parameter to the link-edit utility for linking the load module.

   Here is an example of the ++JCLIN MCS:

   ```
   ++JCLIN … CALLLIBS.
   ```

   Note: If CALLLIBS is not specified, the SYSLIB allocation in the link-edit step is ignored and the NCAL parameter is used when invoking the link-edit utility.
2. Provide link-edit JCLIN that defines the SYSLIB allocation for the libraries containing the modules to be implicitly included by the link-edit automatic library call function.

   SMP/E will save the low-level qualifiers of the data sets in the SYSLIB allocation as a CALLLIBS subentry list in the LMOD entry for the load module.

   Here is an example of link-edit JCLIN that defines a SYSLIB allocation for a load module that needs to use the link-edit automatic library call function.

   ```
   //STEP1 EXEC PGM=IEWBLINK,PARM='RENT,REUS'
   //SYSLMOD DD DSN=SYS1.APPLOAD,DISP=OLD
   //AOS12   DD DSN=SYS1.AOS12,DISP=SHR
   //SYSLIB  DD DSN=SYS1.V2R2M0.PLIBASE,DISP=SHR
   //        DD DSN=SYS1.V2R2M0.APPBASE,DISP=SHR
   //SYSLIN  DD *
      INCLUDE AOS12(MOD00001,MOD00002)
      ENTRY MOD00001
      SETCODE AC(1)
      NAME LMOD01(R)
   /*
   ```
3. Inform your users of special requirements for installing the SYSMOD.
   - To install the SYSMOD, users must run SMP/E.
   - Before installing the SYSMOD, users must define DDDEF entries in the target zone for each of the data sets in the load module's SYSLIB allocation. If the load module has an XZMOD subentry or if the

set-to zone does not have an UPGLEVEL subentry, a DDDEF entry is also required in the target zone for the SMPLTS data set.

Example of a SYSMOD that implements CALLLIBS support

The following is a part of a sample function SYSMOD with a load module that needs to use the link-edit automatic library call function. The numbers associate items in the SYSMOD with the steps listed in "Overview of CALLLIBS support" on page 166.

```
    ++FUNCTION(HXY1100) FILES(3).
    ++VER(Z038).
 1  ++JCLIN CALLLIBS.
    …
    //STEP1 EXEC PGM=IEWBLINK,PARM='RENT,REUS'
    //SYSLMOD DD DSN=SYS1.APPLOAD,DISP=OLD
    //AOS12   DD DSN=SYS1.AOS12,DISP=SHR
 2  //SYSLIB  DD DSN=SYS1.V2R2M0.PLIBASE,DISP=SHR
    //        DD DSN=SYS1.V2R2M0.APPBASE,DISP=SHR
    //SYSLIN  DD *
       INCLUDE AOS12(MOD00001,MOD00002)
       ENTRY MOD00001
       SETCODE AC(1)
       NAME LMOD01(R)
    /*
    …
    ++MOD(MOD00001) RELFILE(2) DISTLIB(AOS12).
    ++MOD(MOD00002) RELFILE(2) DISTLIB(AOS12).
    …
```

The user needs to define DDDEF entries for the data sets specified in the SYSLIB allocation (PLIBASE and APPBASE) and the SMPLTS data set, which SMP/E will use to link-edit the load module. (For details on the SMPLTS data set, see "Use of the SMPLTS library" on page 69 and the SMPLTS section in *z/OS SMP/E Reference*.) Here are examples of defining the DDDEF entries, assuming that the function will be applied to target zone TGT1.

```
 3  SET BDY(TGT1).               /* Set to target zone.    */
    UCLIN.                       /*                        */
    ADD DDDEF(PLIBASE)           /* Define PLIBASE.        */
        DA(SYS1.V2R2M0.PLIBASE)  /* Data set is cataloged. */
        SHR.                     /* SHR for read.          */
    ADD DDDEF(APPBASE)           /* Define APPBASE.        */
        DA(SYS1.V2R2M0.APPBASE)  /* Data set is cataloged. */
        SHR.                     /* SHR for read.          */
    ADD DDDEF(SMPLTS)            /* Define SMPLTS.         */
        DA(SYS1.SMPLTS)          /* Data set is cataloged. */
        SHR.                     /* SHR for read.          */
    ENDUCL.
```

Restrictions in CALLLIBS support

CALLLIBS support puts restrictions on the use of the CALL and NCAL parameters. Processing of the CALL and NCAL parameters in releases of SMP/E that support CALLLIBS (which includes all currently supported releases of SMP/E) is different from processing of those parameters in previous SMP/E releases that did not support CALLLIBS.

Before CALLLIBS support, NCAL was a default parameter passed to the link-edit utility. However, you could use the link-edit UTILITY entry to pass the CALL parameter instead.

With CALLLIBS support, there is no way to directly tell SMP/E to pass the NCAL or CALL parameter. SMP/E ignores any specification of NCAL or CALL, and instead checks for the CALLLIBS subentry in the load module's LMOD entry to determine which parameter to pass to the link-edit utility when linking the load module.

Example 7: JCLIN for load modules residing in a UNIX file system

A load module can reside in a UNIX file system. To determine where the load module resides, SMP/E uses the following information, in addition to the usual JCL statements needed for load modules:

- The PATH operand on the SYSLIB or SYSLMOD statement associated with the load module. The PATH operand alerts SMP/E to the fact that the load module resides in a UNIX file system; however, the PATH value specified is ignored.
- The LIBRARYDD comment statement immediately following the statement with the PATH operand. This comment statement specifies the ddname to be associated with the PATH value on the previous DD statement.
- The user-provided DDDEF entry whose name matches the ddname on the LIBRARYDD comment statement. The DDDEF entry specifies the directory portion of the pathname identified by the ddname. SMP/E uses the PATH value specified in the DDDEF entry to allocate the pathname, and does not check whether this value matches the PATH value specified on the SYSLIB or SYSLMOD DD statement associated with the LIBRARYDD comment statement.

Following are examples of job steps containing SYSLMOD and SYSLIB DD statements that use the PATH operand.

```
   //STEP1 EXEC PGM=IEWBLINK,PARM='RENT,REUS'
 1 //SYSLMOD DD PATH='/path_name1/'
 2 //*LIBRARYDD=BPXLOAD1
   //AOS12   DD DSN=SYS1.AOS12,DISP=SHR
   //SYSLIN  DD *
      INCLUDE AOS12(MOD00001)
      INCLUDE AOS12(MOD00002)
      ENTRY MOD00001
      NAME LMOD01(R)
   /*
   //STEP2 EXEC PGM=IEWBLINK,PARM='CALL,RENT,REUS'
   //SYSLMOD DD PATH=SYS1.LINKLIB,DISP=OLD
   //AOS12   DD DSN=SYS1.AOS12,DISP=SHR
 3 //SYSLIB  DD PATH='/path_calllib3/'
 4 //*LIBRARYDD=BPXCALL3
 3 //        DD PATH='/path_calllib4/'
 4 //*LIBRARYDD=BPXCALL4
   //SYSLIN  DD *
      INCLUDE AOS12(MOD00005)
      INCLUDE AOS12(MOD00006)
      ENTRY MOD00005
      NAME LMOD03(R)
   /*
```

1

Because the SYSLMOD statement specifies a PATH operand, SMP/E expects the next statement to be a LIBRARYDD comment statement.

2

Using the ddname on the LIBRARYDD comment statement, SMP/E updates the LMOD entry for LMOD01 to specify a SYSLIB value of BPXLOAD1. The user needs to provide a DDDEF entry for BPXLOAD1, specifying the appropriate pathname.

3

The SYSLIB DD statement is a concatenation of two DD statements that specify the PATH operand.

4

Using the ddnames on the LIBRARYDD comment statements and the low-level qualifier of the data set specified on the DSN operand, SMP/E updates the LMOD entry for LMOD03 to specify a CALLLIBS subentry list with the values BPXCALL3, and BPXCALL4. The user needs to provide DDDEF entries for BPXCALL3 and BPXCALL4, specifying the appropriate pathnames.

Example 8: JCLIN for SIDEDECKLIB subentries

DD statements in link-edit steps are processed during JCLIN processing to obtain the system library and CALLLIBS libraries for a load module. In addition, the SYSDEFSD DD statement will be processed to obtain the SIDEDECKLIB subentry for a load module. The SIDEDECKLIB subentry identifies the library used to contain a definition side deck of IMPORT control statements. The IMPORT control statements are created by the link edit utility when the load module is link-edited. If the library identifies a partitioned data set (PDS or PDSE), the definition side deck created is a member in that data set. If the library identifies a directory in a UNIX file system, the definition side deck created is a file in that directory.

The following is an example JCLIN link-edit step to define load modules using a definition side deck library:

```
//LINK1    EXEC PGM=IEWBLINK,PARM='CALL,DYNAM(DLL)'
//SYSLMOD  DD DSN=GOS.LOADLIB,DISP=SHR
//SYSLIB   DD DSN=SYS1.SCEELOAD,DISP=SHR
//SYSDEFSD DD DSN=GOS.SGOSSD,DISP=SHR
//SYSLIN   DD *
  INCLUDE AGOSDLIB(GOSMOD1)
  NAME GOSLMOD1
  INCLUDE AGOSDLIB(GOSMOD2)
  NAME GOSLMOD2
/*
```

Note: The SYSDEFSD DD statement may be specified as SMPDUMMY, allowing it to be allocated as a DUMMY data set.

Example 9: JCLIN for UTIN subentries

INCLUDE control statements are used to identify utility input to be included when link-editing a load module. This utility input may be a definition side deck file containing IMPORT control statements, or any other file to be included during the link-edit. A comment on the control statement indicates to SMP/E the INCLUDE statement identifies a utility input file, not a module.

The following is an example JCLIN link-edit step to define load modules including utility input files:

```
//LINK2    EXEC PGM=IEWBLINK,PARM='CALL,RMODE=SPLIT,DYNAM(DLL)'
//SYSLMOD  DD DSN=APPL.LOADLIB,DISP=SHR
//SYSLIB   DD DSN=SYS1.SCEELOAD,DISP=SHR
//SYSLIN   DD *
  INCLUDE APPLDLIB(APPLMOD1)
  INCLUDE APPLDLIB(APPLMOD2)
  INCLUDE SGOSSD(GOSLMOD1)    TYPE=UTIN
  INCLUDE SGOSSD(GOSLMOD2)    TYPE=UTIN
  INCLUDE SGOSDR(IBM/gos/goslmod3_shipped_as_element_MCS) TYPE=UTIN
  ENTRY APPLMOD1
  NAME APPLLMOD
/*
```

Note:

1. SGOSSD would be associated with either a partitioned data set or a directory in the UNIX file system.
2. SGOSDR would be associated with a directory in the UNIX file system.

Processing

This section discusses these topics:

- Summary of JCLIN processing
- General JCLIN coding conventions
- Processing assembler steps
- Processing copy steps
-
- Processing update steps
- Processing other utility steps

Summary

When the JCLIN function of SMP/E is called, SMP/E begins to read the JCLIN input. It scans the JCL for job steps (that is, EXEC PGM=*xxx* or EXEC *procname*) containing information that SMP/E may need. The target or distribution zone is updated in the order in which the job steps are found in the JCLIN input.

Reminder

The input for JCLIN must be free from all JCL errors and must conform to the SMP/E restrictions. Incorrect input can cause unpredictable errors.

SMP/E looks for certain program and procedure names that it recognizes as valid assembler, copy, link-edit, and update steps. It issues a warning message for other program and procedure names if it is not sure how to process them. For more information about programs and procedures not processed by JCLIN, see "Processing other utility steps" on page 189.

If the JCLIN input contains program or procedure names that SMP/E does not recognize, these names can be passed to SMP/E by use of operands on the JCLIN control statement or ++JCLIN 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: Only one additional program or procedure name can be specified for each utility. These names are in addition to the standard names built into SMP/E, and are lower in the search order than the built-in names. Therefore, if you specify `JCLIN UPDATE(PGM=ASMBLR)`, SMP/E still treats a job step specifying `PGM=ASMBLR` as an assembler.

The following sections describe JCLIN processing and coding conventions for each of the job steps SMP/E treats as significant. Remember that:

- Inline JCLIN processed at ACCEPT time updates the distribution zone.
- Inline JCLIN processed at APPLY time updates the target zone.
- The JCLIN command updates the target zone.

General JCLIN coding conventions

SMP/E is not intended to support the wide variety of options and facilities supported in job control language (JCL); therefore, SMP/E has some rules as to how the JCL must be coded in order for SMP/E to process it correctly. If these conventions are not followed, the results are unpredictable: **JCLIN processing may not run successfully, and follow-on processing may be affected.** Certain conventions must be followed:

- Specify all information in uppercase characters (verbs as well as values). This is necessary to avoid syntax errors or incorrect results during SMP/E processing.

  Note: This convention does not apply to values on the ALIAS statement. These values must be specified in the desired case (uppercase or mixed-case), because they are used as is.
- SMP/E does not recognize in-stream procedures. When SMP/E examines the JCLIN input, it looks only at JCL statements provided inline. That is, if you invoke any cataloged procedures in the JCLIN input, SMP/E does not expand those procedures in order to access any required DD statements. If you use cataloged procedures, they must follow the same conventions as the IBM-supplied system generation cataloged procedures (ASMS for assemblies and LINKS for link-edits). The conventions are described in later sections.
- The ddname for a distribution library or target library should match the lowest-level qualifier of the data set name. This is not a requirement, but rather a recommendation to help you specify the data sets used by SMP/E. During processing, a data set is referred to by its ddname, not the complete data set name. If the ddname and the lowest-level qualifier DSN are kept the same, correlating these names may be easier for you. For example, the DD statement for SYS1.LINKLIB can be specified as:

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

This convention for correlating ddnames and data set names is used by IBM when it distributes functions or PTFs. Therefore, you should avoid changing the ddnames of libraries used for elements provided by IBM. If you change these ddnames in the element entries, the ddnames for these elements in subsequent SYSMODs may not match the ddnames in the entries, and SMP/E may not be able to process the SYSMODs.

- Concatenated data sets must not be used for input. For example, a link-edit step has an INCLUDE statement 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/E is able to process the JCL, but the updates to the zone being processed are not sufficient to enable service to be installed. There are two problems with this example:

– The ddnames are not the same as the lowest-level qualifier of the DSN.

– The data sets are concatenated.

The following examples show the correct format for the INCLUDE statements and DD statements:

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

and

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

- A continued utility control statement should be used only when absolutely necessary. Although SMP/E attempts to support all existing formats of the utility control statements, it cannot completely duplicate the syntax checking of the utility. The safe method is to use the simplest format of the utility control statement.

For example, rather than code:

```
 INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2,            X
               GIMMPDC1,GIMMPDC2)
```

a better (and safer) method is:

```
 INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2)
 INCLUDE AOS12(GIMMPDC1,GIMMPDC2)
```

The link-edit utility accepts both formats, but the second example is the safe format.

Note: Generally, SMP/E does **not** require a continuation character in column 72. However, if a link-edit control statement is to be continued to the next statement, a nonblank continuation character **must** be placed in column 72. This is necessary to avoid syntax errors or incorrect results during SMP/E processing.

- The DD statement identifying the input control statements for a utility **must be** 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(GIMMPDRV,GIMMPDR1,GIMMPDR2,…)
  ENTRY GIMMPDRV
```

```
  SETCODE AC(1)
  NAME GIMMPP(R)
/*
```

The preceding example is valid, but the following is **not** valid.

```
//STEP1   EXEC PGM=IEWL
//SYSLIN  DD *
  INCLUDE AOS12(GIMMPDRV,GIMMPDR1,GIMMPDR2,…)
  ENTRY GIMMPDRV
  SETCODE AC(1)
  NAME GIMSMP(R)
/*
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//AOS12   DD DSN=SYS1.AOS12.DISP=SHR
```

For each step encountered, SMP/E saves all the JCLIN records in a work area until the first non-JCL (that is, first statement after the SYSIN DD statement or the SYSLIN DD statement) is found. The saved records are then 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 not be in the work area when the search performed, and SMP/E might not be able to capture the necessary data.

- Conditional JCLIN comment statements cannot appear within the utility control card section of a JCLIN job step. See "Conditional JCLIN comment statements" on page 156 for information on conditional JCLIN comment statements.

Note: SMP/E has no column limitations for operands beyond the normal JCL rules.

Processing assembler steps

Assembler steps are identified by one of these EXECs:

- EXEC PGM=IFOX00
- EXEC PGM=IEUASM
- EXEC PGM=IEV90
- EXEC PGM=ASMBLR
- EXEC PGM=ASMA90
- EXEC ASMS
- EXEC PGM=*asmpgm* [if the JCLIN specified ASM(PGM=*asmpgm*)]
- EXEC *asmproc* [if the JCLIN specified ASM(*asmproc*)]

Either ASSEM or SRC entries are built, depending on where the assembler input is obtained.

- ASSEM entries are built when the assembler input is inline (the SYSIN DD statement does not point to another data set).
- SRC entries are built when the SYSIN DD statement points to a member of a partitioned data set.

Building ASSEM entries

As the assembler step is processed, every assembler control statement [that is, every card from the SYSIN DD * statement until the end of input (/* or //)] is written as an ASSEM entry.

The name of the ASSEM entry is determined by looking at the JCL for the assembly step. If the step is of the EXEC PGM=*name* type, SMP/E looks for any of the following statements to find the member name for the DSN or DSNAME library:

```
//SYSPUNCH DD DSN=…
```

or

```
//SYSGO    DD DSN=…
```

or

```
//SYSLIN   DD DSN=…
```

These are examples of the statements SMP/E looks for:

```
//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 *asmproc* type, SMP/E looks for the MOD=*name* operand of the PROC statement. If it is found, that name is used as the ASSEM entry name. If not, one of three DD statements (SYSPUNCH, SYSGO, or SYSLIN DD) must be present and the ASSEM entry name is obtained from the DD statement. If any ASSEM entry previously existed, the current set of assembler input completely replaces the previously saved assembler input.

Building SRC entries

If the SYSIN DD statement points to a member of a partitioned data set, SMP/E assumes that the member is a source. It then creates a SRC entry whose name is the same as the member name on the SYSIN DD statement. The name of the DISTLIB for the SRC entry is the low-level identifier of the data set name on the SYSIN DD statement.

Building MAC entries

In the process of reading and writing inline assembler input and building an ASSEM entry, the operation field (operation codes) of each assembler statement is scanned. If column 1 of the assembler statement is blank, SMP/E considers the first character string found to be the operation code. If column 1 is not blank, the second character string found is the operation code.

For each operation code found, SMP/E determines whether it is a macro invocation or an assembler instruction. It does so by using its default set of OPCODE definitions. SMP/E's default set of OPCODE definitions identifies all the operation codes in *ESA/390 Principles of Operation, SA22-7201*, as well as all the assembler instructions supported by the High Level Assembler (ASMA90).

You may optionally provide your own OPCODE member to override SMP/E's default set of OPCODE definitions. The user-provided OPCODE member is a text member stored in a user-allocated PDS named SMPPARM. You are not required to allocate the SMPPARM data set, unless you want to supply your own user-defined member. The operation fields (operation codes) of the assembler input are scanned during JCLIN processing and are compared to the OPCODE definitions (either user-defined or default).

SMP/E uses the following method to determine whether an assembler instruction is an OPCODE or a macro:

- SMP/E looks for a user-allocated SMPPARM data set.
- If SMPPARM is **not** found, SMP/E uses the default set of OPCODE definitions.

  If SMPPARM is found and there is a user-defined OPCODE member specified on the JCLIN statement or in the ++JCLIN MCS, then SMP/E searches for the specified member in SMPPARM. If it finds that member, it will look first in that member for a definition of the character string.
- If the user-defined OPCODE member specified is not found, SMP/E searches SMPPARM for the GIMOPCDE member. If it finds the GIMOPCDE member, it will look **only** in that member for a definition of the character string. (The GIMOPCDE member, if it exists in SMPPARM, will completely override SMP/E's default set of OPCODE definitions.)
- If the GIMOPCODE member is not found, SMP/E uses the default set of OPCODE definitions.

- If the character string is not defined in either the SMPPARM data set or the default set, SMP/E considers it to be a macro.

For a description of the format of the OPCODE member control cards, see the "SMP/E OPCODE Member Control Statements" section in *z/OS SMP/E Reference*.

For each macro found in the assembly input, SMP/E takes these actions:

- Locates the MAC entry in the zone being processed. If the entry is found, it is modified. If the entry is not found, a new entry is created.
- Updates the MAC entry by adding the name of the assembly module (as previously described) as a GENASM subentry. This now means that each macro used during the assembly includes a reference to the target zone ASSEM entry. Therefore, when a SYSMOD is processed that modifies that macro, SMP/E knows what target zone ASSEM entries should be reassembled in order to include the new macro.

Note: After JCLIN processing that creates a new MAC entry, the only data present in the MAC entry is the set of GENASM subentries. Additional data, such as the distribution library, is added to the MAC entry during the installation of the SYSMOD supplying the actual macro.

Processing copy steps

Copy steps are identified by one of these EXECs:

- EXEC PGM=IEBCOPY
- EXEC PGM=*copypgm* [if the JCLIN specified COPY(PGM=*copypgm*)]
- EXEC *copyproc* [if the JCLIN specified COPY(*copyproc*)]

When a copy step is encountered, SMP/E searches through the JCL looking for the SYSIN DD * statement. All records from the SYSIN DD * statement to the end of input (/* or //) are assumed to be the copy input control statements.

Note:

1. Copy input must be **inline**, not pointing to another data set.
2. The only copy utility control statements allowed are COPY (or C), COPYMOD (or CM), and SELECT (or S).
3. The COPYMOD control statement is used when processing program elements or copied modules.

When scanning the copy input, SMP/E assumes the ddnames of the statement are the same as the lowest-level qualifier of the data set referred to. If the COPY or COPYMOD statement is set up without this convention, SMP/E generates incorrect DLIB and SYSLIB names in the MOD and LMOD entries.

By scanning the IEBCOPY control statements, SMP/E knows which members of the DLIB are being copied (from the IEBCOPY SELECT MEMBER statements), from which DLIB they are being copied (the IEBCOPY INDD ddname), and to which target libraries they are going (the IEBCOPY OUTDD ddname).

The members can be macros, modules, source elements, or data elements. SMP/E has no way of determining the type from the standard IEBCOPY control statements. To find this information, SMP/E looks for the TYPE comment on the COPY (or COPYMOD) INDD=*ddname*,OUTDD=*ddname* control statement. The format of the TYPE comment on the COPY or COPYMOD statement is:

```
copystmt INDD=ddname,OUTDD=ddname   TYPE=type
```

where:

copystmt
is COPYMOD (for copied modules or program elements) or COPY (for all other elements)

type
indicates the type of element in the distribution library. It can be specified as:

DATA
for data elements. If DATA is specified, SMP/E skips to the next COPY or COPYMOD control statement without creating any entries for the data elements.

MAC
for macros.

MOD
for modules.

PROGRAM
for program elements.

SRC
for source elements.

If the TYPE=*type* parameter is not specified, the default is TYPE=MOD.

There must be at least one blank between the COPY or COPYMOD statement and the comment. TYPE=*type* must be the first character string in the comment, and no blanks must precede or follow the "=" sign. This information is used only if a SELECT statement follows the COPY or COPYMOD statement.

The SELECT statement can name either the member to be copied or an alias name for a member. The format of these SELECT statements is:

```
SELECT MEMBER=member
SELECT MEMBER=alias             ALIAS OF member
```

The first SELECT statement identifies a member to be copied. The second identifies an alias and names the associated member in the comment portion of the statement.

Note: A SELECT statement identifying an alias can specify only one name on the MEMBER operand and cannot specify RENAME.

Building DLIB entries

If a COPY or COPYMOD statement is followed either by another COPY or COPYMOD statement or by an EXCLUDE statement, SMP/E assumes that the INDD library is totally copied to the OUTDD library. Because SMP/E does not look at either the DLIB or target library during JCLIN processing, it has no way of knowing what elements are contained in the INDD library. Therefore, a DLIB entry with the same name as the INDD ddname is created. This indicates that the entire library has been copied to the library specified by the OUTDD ddname. If the INDD ddname DLIB entry already exists, SMP/E adds the OUTDD ddname to the DLIB entry. SMP/E records these ddnames in alphabetical order. A DLIB entry can have at most two SYSLIB subentries. If the DLIB entry already has two SYSLIB subentries, SMP/E replaces the second SYSLIB ddname with the ddname specified on the OUTDD operand of the COPY or COPYMOD statement.

Later, whenever a SYSMOD is processed during APPLY or ACCEPT, SMP/E can check the DISTLIB ddname for each of the SYSMOD's elements to determine whether that element belonged to a totally copied distribution library. This is how DLIB entries are used during APPLY and ACCEPT processing:

- For ++MOD statements, SMP/E can construct a MOD entry indicating that the module has been copied and that the load module name is the same as the module name. In addition, an LMOD entry can be built, indicating a copy with a SYSLIB equal to the SYSLIB value in the DLIB entry.
- For ++MAC, ++MACUPD, ++SRC, ++SRCUPD, and data element message control statements, SMP/E can determine that the element is part of a totally copied library, and can then fill in the SYSLIB field in the appropriate element entry with the SYSLIB ddname saved in the DLIB entry. If there are two SYSLIB ddnames in the DLIB entry, SMP/E uses the second value.

Note: SMP/E treats copies with exclude as total copies. It does not retain information about which elements were excluded during the creation of the initial library.

Building MOD and LMOD entries

If the COPY (or COPYMOD) statement is followed by a SELECT statement and either specifies TYPE=MOD or lets TYPE default to MOD, SMP/E builds MOD and LMOD entries for the elements specified in the SELECT statement.

The COPY (or COPYMOD) and SELECT statements specify the members of the DLIB that are being copied (the IEBCOPY select member statements), aliases for these members (the IEBCOPY select member statements with alias comments), the DLIB they are being copied from (the IEBCOPY INDD ddname), the 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 statement list, SMP/E builds:

- A MOD entry with the same name as the selected element, with a DISTLIB value equal to the INDD operand value and an LMOD subentry with the same name as the copied element (unless the rename function of the copy select statement was used, in which case the LMOD subentry name is as specified in the RENAME operand).

  If a SELECT statement names an alias as the member, that name is added as a TALIAS value in the MOD entry for the module named in the SELECT statement comment.
- An LMOD entry with the same name as the selected element (unless the rename function of the copy select statement was used, in which case the LMOD name is as specified in the RENAME operand) is created, indicating that the load module has been copied and that its SYSLIB is equal to the ddname specified in the OUTDD COPY or COPYMOD statement operand.

  An LMOD entry can have at most two SYSLIB subentries. If the LMOD entry already has two SYSLIB subentries, SMP/E replaces the second SYSLIB ddname with the ddname specified on the OUTDD COPY or COPYMOD statement operand.

Note: Because SMP/E only looks at the JCLIN input and has no knowledge about the DLIBs or target libraries built into it, SMP/E cannot determine what types of libraries are being copied. SMP/E assumes 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/E **does not** build MAC entries and SRC entries, but builds extraneous MOD and LMOD entries.

Building MAC entries

If the COPY statement is followed by a SELECT statement and specifies TYPE=MAC, SMP/E builds MAC entries for the elements specified in the SELECT statement. SMP/E determines the macro name from the SELECT statement, the macro DISTLIB from the INDD=*ddname* parameter on the COPY statement, and the macro SYSLIB from the OUTDD=*ddname* parameter on the COPY statement.

If no entry exists for a macro, SMP/E creates one using the DISTLIB and SYSLIB information obtained from the COPY statements. If an entry already exists for a macro, SMP/E compares the DISTLIB and SYSLIB subentries in the existing entry to those obtained from the COPY statements. If they are different, SMP/E updates the macro entry with the values from the COPY statement.

If a SELECT statement names an alias as the member, that name is added as a MALIAS value in the MAC entry for the macro named in the SELECT statement comment.

SMP/E does not support the RENAME option of the COPY statement.

Building SRC entries

If the COPY statement is followed by a SELECT statement and specifies TYPE=SRC, SMP/E builds SRC entries for the elements specified in the SELECT statement. SMP/E determines the source name from the SELECT statement, the source DISTLIB from the INDD=*ddname* parameter on the COPY statement, and the source SYSLIB from the OUTDD=*ddname* parameter on the COPY statement.

If no entry exists for the source, SMP/E creates one using the DISTLIB and SYSLIB information from the COPY statements. If a source subentry already exists, SMP/E compares the DISTLIB and SYSLIB

subentries in the existing entry to those obtained from the COPY statements. If they are different, SMP/E updates the source entry with the values from the COPY statement.

SMP/E does not support the RENAME option of the COPY statement.

Processing link-edit steps

Link-edit steps are identified by one of these EXECs:

- EXEC PGM=HEWL
- EXEC PGM=HEWLH096
- EXEC PGM=HEWLKED
- EXEC PGM=IEWBLINK
- EXEC PGM=IEWL
- EXEC PGM=LINKEDIT
- EXEC PGM=*lkedpgm* [if the JCLIN specified LKED(PGM=*lkedpgm*)]
- EXEC LINKS
- EXEC *lkedproc* [if the JCLIN specified LKED(*lkedproc*)]

When SMP/E encounters a link-edit step, it reads through the JCL control statements looking for the SYSLIN DD statement containing the link-edit control card input. All records from the SYSLIN DD statement to the end of input (/* or //) are assumed to be link-edit utility control statements.

- These records **must be** control statements, not object modules.
- All link-edit control statements must start in or after column 2.
- The SYSLIN input **must be** inline. It cannot point to a member of another data set, because SMP/E then could not analyze the data.

Note: When SMP/E detects the SMPLTS comment statement during JCLIN processing, it skips all the steps in the SMPLTS job.

Link-edit steps must not be sensitive to the order of execution of other link-edit steps either for the same FMID or for another, conditionally coexistent FMID. No elements to be included in a link-edit step should be derived from the output of another link-edit step. Also, if multiple load modules and target libraries are involved, SMP/E organizes the link-edit steps for the most efficient invocations of the link-edit utility (which might not be in the same order as in the JCLIN data).

For example, suppose a product consists of a base function and a dependent function. The dependent function conditionally coexists with the base function; it can be installed with the base function, but is not a prerequisite for the base function. The base function must have its own JCLIN data that completely describes the elements it contains, because a user may choose to install the functions together or separately. If the base function is installed separately, its JCLIN data cannot contain a link-edit step that includes elements from the dependent function, because those elements are not yet available.

When scanning the input, SMP/E looks for and performs special processing for selected DD statements and link-edit control statements.

- Some DD statements and link-edit statements are processed to create MOD and LMOD entries.
- Some link edit statements are not processed, but are saved as is in the LMOD entry so they can be passed to the link-edit utility when the load module needs to be relinked.

  Note: These statements may be processed by the link-edit utility after they are saved in the LMOD entry by the command being run. For example, if the JCLIN data is being processed by the APPLY command, these statements may be processed by the link-edit utility as part of installing the module. On the other hand, if the JCLIN data is being processed by the JCLIN command, these statements are saved in the LMOD entry and are passed to the link-edit utility only when the module is link-edited again at some future date.

For more information about link-edit utility rules, see *z/OS MVS Program Management: User's Guide and Reference*.

Table 13 on page 178 and Table 14 on page 178 show a summary of this information. They are followed by more detailed guides.

Reminder

JCLIN by itself does not force SMP/E to link-edit a load module. The JCLIN only causes SMP/E to update the CSI entries with the information in the JCLIN. To force a link-edit, you must also supply the appropriate element statements (++MOD).

For example, to add an alias to a load module, you must supply JCLIN and a ++MOD statement for a module contained in the load module. The JCLIN updates the LMOD entry with the new link-edit control statements, and the ++MOD statement tells SMP/E to perform the link-edit.

Table 13. Summary of how DD statements are processed as JCLIN data

Statement	Processed to create entries	Saved only as + +LMODIN data in the LMOD entry	Should not be specified in JCLIN data
SYSDEFSD	Yes		
SYSLIB	Yes (See note)		
SYSLMOD	Yes		

Note:

1. A SYSLIB DD statement is processed only if CALLLIBS was specified on the JCLIN command or the ++JCLIN MCS.
2. The SYSDEFSD DD statement may be specified as SMPDUMMY, allowing it to be allocated as a DUMMY data set.

Table 14. Summary of how link-edit control statements are processed as JCLIN data

Statement	Processed to create entries	Saved only as + +LMODIN data in the LMOD entry	Should not be specified in JCLIN data
ENTRY		Yes	
EXPAND			Do not specify in JCLIN data
IDENTIFY			Do not specify in JCLIN data
INCLUDE	Yes		
INSERT and OVERLAY		Yes	
LIBRARY			Do not specify in JCLIN data (See note)
NAME	Yes		
ORDER		Yes	
SETSSI			Do not specify in JCLIN data

Table 14. Summary of how link-edit control statements are processed as JCLIN data (continued)			
Statement	**Processed to create entries**	**Saved only as + +LMODIN data in the LMOD entry**	**Should not be specified in JCLIN data**
All other statements except comments		Yes	
Note: Do not use LIBRARY statements to specify additional automatic call libraries.			

ALIAS statement

To ensure that SMP/E can process your link-edit ALIAS control statements, you must address the following considerations:

- **General considerations**
  - An ALIAS control statement can span any number of 80-byte records.

    Note:

    1. If you assign a load module residing in a PDSE an alias value greater than eight characters, you cannot later use the ++DELETE statement to delete that alias value (and not the associated load module). To delete such an alias value without deleting the load module, you need to resupply JCLIN to define the load module without providing an ALIAS statement for the alias value to be deleted. Make sure to also include a ++MOD statement for a module in the load module to force SMP/E to relink the load module.
    2. If the set of alias values (or symbolic links) for an existing load module changes, SMP/E does not remove the original set of alias values (or symbolic links) from a PDS target library or a UNIX file system. If it is important that these original values be removed, you can delete them by either:
       a. Using the ++DELETE MCS to remove the specific alias value or symbolic link that is not needed, or
       b. Using the ++DELETE MCS to remove the entire load module, using JCLIN to redefine the load module with the correct set of alias values and symbolic links, and then rebuilding the load module by supplying all of its modules.
  - Column 1 of all 80-byte records composing an ALIAS control statement must contain a blank (X'40').
  - The data for the first 80-byte record of an ALIAS control statement must start in column 2 or later and end anywhere up to and including column 71.
  - The control statement type (ALIAS) must be followed by at least 1 blank (X'40').
  - The control statement type (ALIAS) must be in uppercase.
  - Columns 73 through 80 of an 80-byte record are ignored.
  - An alias value can be from one to 1023 characters.
  - An alias value can be composed of characters in the range X'41' through X'FE'.

    Note: Although the binder also accepts characters X'0E' (shift-out character) and X'0F' (shift-in character), SMP/E does not accept them.
  - An alias value can be enclosed in single apostrophes. It must be enclosed in single apostrophes when:
    - It contains a comma. For example:

      ```
      ALIAS 'If_the_alias_contains_a_comma,_enclose_it_in_apostrophes.'
      ```

    - It contains an apostrophe. For example:

```
ALIAS 'It''s_the_apostrophe_in_"it''s"_that_necessitates apostrophes.'
```

- It contains a left parenthesis. For example:

```
ALIAS 'A_left_parenthesis_-(-_means_enclose_it_in_apostrophes.'
```

- It contains a right parenthesis. For example:

```
ALIAS 'A_right_parenthesis_-)-_means_enclose_it_in_apostrophes.'
```

– If an apostrophe is part of the alias value (not a delimiter), two apostrophes need to be specified in the appropriate location in the alias value. These two apostrophes count as two characters in the 1023-character limit for an alias value. Here is an example:

```
ALIAS 'It''s_the_quote_that_makes_apostrophes_necessary.'
```

– The single apostrophes used to enclose an alias value do not count as part of the 1023-character limit for an alias value. For example, the alias value in the following example contains 10 characters:

```
ALIAS  'Only_ten!!'
```

– SMP/E uses the alias value exactly as is. SMP/E does not try to enforce any rules the binder may be using as a result of the CASE execution parameter.

- **Continuation records**

– Column 72 of a given 80-byte record must be a nonblank character if the control statement is continued onto the next 80-byte record. The character in column 72 denotes only continuation and is never part of an alias value.

– The data for continuation records (80-byte records 2 through *n* of an ALIAS control statement) can start in column 2 or later and end anywhere up to and including column 71 (for example, if multiple aliases are being specified).

The data for a continuation record **must** start in column 2 if it is part of an alias value that is being continued from the previous 80-byte record. An alias value that is continued from one 80-byte record to another 80-byte record must meet these requirements:

- Extend through column 71 of the first 80-byte record
- Start in column 2 of the next 80-byte record
- Have a nonblank continuation character in column 72

An example:

```
        1         2         3         4         5         6         7         8
----+----0----+----0----+----0----+----0----+----0----+----0----+----0----+----0
 ALIAS          This_is_a_very_long_value_that_is_continued_to_the_next*
 _card!
```

- **Entry points**

One format of the ALIAS statement supported for the binder allows an alternative entry point to be specified into a load module. If this format is used, each alias name with an associated entry point must be specified on its own 80-byte record, with a separate ALIAS statement; no other aliases should be specified on that statement. If multiple alias values of this format are specified on a single ALIAS control statement, only the first is recognized; the rest are ignored.

Note: When this form of the ALIAS control statement is used, the maximum length for an alias value is 61 characters.

Suppose that a load module has the following aliases: ALA1, ALA2, ALA3, and ALA4. ALA1 and ALA2 are associated with entry point names ENTRYPT1 and ENTRYPT2, respectively.

– Here are examples of how the aliases should be specified:

```
 ALIAS ALA1(ENTRYPT1)
 ALIAS ALA2(ENTRYPT2)
 ALIAS ALA3
 ALIAS ALA4
or
 ALIAS ALA3,ALA4
```

  - Here are examples of how the aliases should not be specified:

```
 ALIAS ALA1(ENTRYPT1),ALA2(ENTRYPT2),ALA3,ALA4
or
 ALIAS ALA1(ENTRYPT1),ALA3
 ALIAS ALA2(ENTRYPT2),ALA4
```

- **Multiple aliases**
  - Multiple alias values can be specified on a single ALIAS control statement as long as they are not in the form *alias(entrypoint)* Multiple alias values must be separated by commas. For example:

```
 ALIAS ALIAS1,ALIAS2,ALIAS3,ALIAS4
```

  - Multiple alias values can span multiple 80-byte records. When this occurs, there must be a nonblank character in column 72 and one of the following must be true:
    - The last alias value on the 80-byte record that is being continued must be followed by a comma and one or more blanks (', ...'). Here is an example:

```
        1         2         3         4         5         6         7         8
----+----0----+----0----+----0----+----0----+----0----+----0----+----0----+----0
 ALIAS ALIAS1,ALIAS2,                                          *
   ALIAS3,ALIAS4
```

    - The last alias value on the 80-byte record that is being continued must be followed by a comma in column 71. For example:

```
        1         2         3         4         5         6         7         8
----+----0----+----0----+----0----+----0----+----0----+----0----+----0----+----0
 ALIAS       ALIAS1,ALIAS2,A_relatively_long_ALIAS_but_not_1023_chars.,*
   ALIAS4,ALIAS5
```

    - The last alias value on the 80-byte record that is being continued can be coded such that part of the alias value appears on the current 80-byte record and part appears on the next 80-byte record (see the rules for continuation records). Here is an example:

```
        1         2         3         4         5         6         7         8
----+----0----+----0----+----0----+----0----+----0----+----0----+----0----+----0
 ALIAS   ALIAS1,ALIAS2,A_relatively_long_ALIAS.,ALIAS4,Not_too_long_but*
 _wraps.,ALIAS5,ALIAS6
```

  - If a blank (X'40') follows an alias value, SMP/E assumes there are no more alias values for the current ALIAS control statement.

- **Symbolic links**

  One format of the ALIAS control statement supported for the binder allows the definition of symbolic links for a load module in a UNIX file system. Symbolic links are defined with the (SYMLINK,*symlink*) and (SYMPATH,*sympath*) operands of the ALIAS link-edit control statement.

  Note: To replace or delete the symbolic links specified by an ALIAS control statement, you must first use the ++DELETE MCS to delete the entire load module. You can then redefine the load module with a new set of symbolic links (if any) with inline JCLIN and rebuild the load module by supplying its modules.

  The syntax rules used by SMP/E for (SYMLINK,*symlink*) and (SYMPATH,*sympath*) operands on an ALIAS link-edit control statement are the same as those listed in the previous "General Considerations" and "Continuation Records" bullets, with the following additions:

  - The SYMLINK and SYMPATH keywords must be in uppercase.

- Each SYMLINK or SYMPATH keyword must be immediately preceded by an opening parenthesis, followed by a comma and the appropriate value, and terminated by a closing parenthesis.
- Every SYMLINK must have an associated SYMPATH.
- The SYMLINK must precede the SYMPATH with which it is to be associated. See the description of the SYMLINK and SYMPATH operands in *z/OS SMP/E Reference* for a more detailed description of the rules for associating SYMPATH and SYMLINK operands to form symbolic links.
- A *symlink* or *sympath* value may be from 1 to 1023 characters.
- The rules for enclosing *symlink* or *sympath* values in single apostrophes are the same as those for alias values.
- The rules for including an apostrophe, comma, or right or left parenthesis as part of the *symlink* or *sympath* value (not a delimiter) are the same as those for alias values.
- SMP/E does not enforce any rules the binder may be using as a result of the CASE(MIXED|UPPER) execution parameter.

The following example shows an ALIAS statement that defines an alias and a symbolic link:

```
ALIAS alias_1,(SYMLINK,symbolic_link_1),(SYMPATH,symbolic_path_1)
```

CHANGE statement

CHANGE statements are saved in the LMOD entry and are associated with the DLIB module name found on the next INCLUDE statement in the JCL. If the same INCLUDE statement is processed later by JCLIN, CHANGE statements in the LMOD entry associated with this INCLUDE statement are deleted and then replaced by any associated CHANGE statements in the latest job. CHANGE statements are passed to the linkage editor only when the associated DLIB module is to be replaced in the load module.

A function SYSMOD must not contain a CHANGE statement in a link-edit step, unless PTFs can be built without an IDENTIFY statement for the changed CSECT.

Note: RESTORE processing is limited for a SYSMOD that uses the CHANGE statement in inline JCLIN. When such a SYSMOD is applied, the existing LMOD entry (if any) is first backed up on the SMPSCDS and is updated with the inline JCLIN containing the CHANGE statement. When that SYSMOD is restored, the backup copy of the LMOD entry (which does not have the updates from the CHANGE statement) replaces the target zone LMOD entry, and the information from the CHANGE statement is lost. Modules whose names were changed by the inline JCLIN remain in the load module under their changed names.

ENTRY statement

Each load module consisting of more than one distribution library module must have an ENTRY statement; otherwise, the entry point of the load module changes each time the load module is relinked by SMP/E.

EXPAND statement

EXPAND statements should not be used in JCLIN data, because they would be saved in the LMOD entry and would cause the load module to be expanded each time it is link-edited. This is not always desirable. However, the EXPAND statement is allowed as input for the ++ZAP MCS. See *z/OS SMP/E Reference* for more information.

IDENTIFY statement

IDENTIFY statements should not be used in JCLIN data. They are produced as part of servicing a module. If found in the JCLIN, they are stored in the LMOD entry and can result in incorrect data being stored during the application of service.

INCLUDE *ddname*(*name,name...*) statement

INCLUDE statements are used to identify the modules in the load module. They are also used to identify utility input to be included when the load module is link-edited. This is denoted by the TYPE comment on the INCLUDE statement. The format of the TYPE comment on the INCLUDE statement is:

INCLUDE *ddname*(*name,name...*) TYPE=UTIN

If the TYPE comment is not specified, SMP/E assumes that the INCLUDE statement identifies modules.

- **Processing modules**

  The INCLUDE statement is used to identify a module to be included in a load module as follows:

  ```
  INCLUDE ddname(name,name,...)
  ```

  where *name* identifies the module name, and *ddname* identifies the library ddname where the module resides.

  A module name must be 1 to 8 uppercase alphabetic (A-Z), numeric (0-9), national (@, #, $), or X'C0' characters. The module is expected to be a member of a data set.

  The member names are assumed to be modules existing in distribution library *ddname*. SMP/E builds MOD entries for each member name specified and sets the DISTLIB value in each MOD entry to *ddname*. (An exception to this is when the ddname is SYSLMOD. In that case, no MOD entry is built for the INCLUDE statement.) SMP/E does not refer to the *ddname* DD statement to determine the actual library referred to. Therefore, all ddnames specified on INCLUDE statements must be the actual ddnames assigned to the products.

  The INCLUDE statements are not saved in the LMOD entry, because they are not necessary when the load module is link-edited. All link-edits requested by SMP/E are CSECT-replaces; the load module is built from the new version of the updated CSECT and the existing load module from the target library.

  The ddnames SYSPUNCH and SMPOBJ are reserved for inclusions of object decks produced by assembly steps that are not to be link-edited to a distribution library during ACCEPT processing. In both cases, the name stored in the MOD entry's DISTLIB subentry is SYSPUNCH.

- **Processing utility input**

  The INCLUDE control statement can also be used to identify utility input to be included when link-editing a load module. This utility input may be a definition side deck file containing IMPORT control statements, or any other file to be included during the link-edit. A comment on the control statement indicates to SMP/E the INCLUDE statement identifies a utility input file, not a module.

  ```
  INCLUDE ddname(name,name...) TYPE=UTIN
  ```

  In this case, *name* identifies the utility input file name and *ddname* identifies the ddname of the library where the file resides.

  Each utility input file found on INCLUDE statements is saved in the UTILITY INPUT subentry list of an LMOD entry. The *name* and *ddname* determine the uniqueness of a subentry and only one subentry value for a given *name* and *ddname* combination is saved in the UTILITY INPUT subentry list.

  A utility input file can be either a member of a partitioned data set or a file in the UNIX file system. When you identify a member of a partitioned data set, the *name* must be 1 to 8 uppercase alphabetic (A-Z), numeric (0-9), national (@,#,$), or X'C0' characters. When you identify a file in the UNIX file system,

  – The *name* can be 1 to 1023 characters in length.
  – The *name* can be composed of the characters X'41' through X'FE'.
  – The *name* must be a relative filename. That is, it cannot begin or end with a slash (/).
  – The *name* can be enclosed in single quotation marks. The *name* must be enclosed in single quotation marks if it contains a single quotation mark, left parenthesis, right parenthesis, or a comma. The single quotation marks used to enclose the name (the delimiters) do not count as part of the 1023 character limit.

- Any single quotation mark specified as part of the *name* (not the delimiters) must be doubled. A pair of single quotation marks count as two characters in the 1023 character limit.
- The TYPE=UTIN comment cannot extend beyond column 71 (a nonblank character in column 72 indicates statement continuation.)

For an example, see "Example 9: JCLIN for UTIN subentries" on page 169.

- **Continuation records**
  - Column 72 of a given 80-byte record must be a nonblank character if the control statement is continued onto the next 80-byte record. The character in column 72 denotes continuation only and is never part of the *name* value.
  - The data for continuation records (80-byte records, 2 through n of an INCLUDE control statement) can start in column 2 or later and end anywhere up to and including column 71 (for example, if multiple *names* are being specified). The data for a continuation record must start in column 2 if it is part of a *name* value that is being continued from the previous 80-byte record. A *name* value that is continued from one 80-byte record to another 80-byte record must meet these requirements:
    - Extend through column 71 of the first 80-byte record.
    - Have a nonblank continuation character in column 72 of the first 80-byte record.
    - Start in column 2 of the next 80-byte record.

    Here are some examples:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
 INCLUDE SGOSSD(GOSLMOD1,GOSLMOD2,GOSLMOD3,GOSLMOD4,                   x
                GOSLMOD5,GOSLMOD6)
 INCLUDE SGOSDR('IBM/gos/goslmod3_shipped_as_''element''_MCS_which_spanx
 s_lines_looks_like_this') TYPE=UTIN
```

  - If any part of the TYPE=UTIN comment extends past column 71, the entire comment must be moved to the next line and a nonblank character placed in column 72 to indicate that the INCLUDE statement is being continued.

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
  INCLUDE SGOSDR(IBM/gos/goslmod1,IBM/gos/goslmod2,GOSLMD7,GOSLMD8)    x
       TYPE=UTIN
```

INSERT and OVERLAY statements

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.

LIBRARY statement

Typically, do not use LIBRARY statements in JCLIN data. An exception is when the CALLLIBS operand is specified on the JCLIN command or ++JCLIN MCS, or when //*CALLLIBS=YES is encountered after a job card preceding a link-edit step. JCLIN processing then allows for the LIBRARY statement to be used to specify those modules (external references) that are to be excluded from the automatic library search during the following processing:

- The current linkage editor job step (restricted no-call function)
- Any subsequent linkage editor job step (never-call function)

A LIBRARY statement should be used only if a SYSLIB DD statement is also used. It should not be used to specify additional automatic call libraries; the SYSLIB DD statement should be used instead.

NAME *lmodname*(R) statement

When SMP/E encounters either the NAME control statement or the end of input with no NAME statement, SMP/E builds an LMOD entry. How SMP/E determines the name of the LMOD depends on the JCL being scanned:

- If the NAME statement is found, SMP/E gets the LMOD name from the *lmodname* field of the NAME statement.
- If no NAME statement is found and a SYSLMOD DD statement is present, SMP/E gets the LMOD name from the member name of the data set specified. If no member name is specified, SMP/E issues an error message identifying the JOBNAME and STEPNAME and the reason for the error.
- If no NAME and SYSLMOD DD statements are found, SMP/E searches for the MOD=*name* operand in the JCL and uses that name as the LMOD name. If no MOD=*name* operand is found, SMP/E issues an error message.

The NAME statement can also be used to specify a load module's highest acceptable link edit return code value. A comment on the control statement is used to specify the return code value as follows:

```
NAME lmodname       RC=rc
or
NAME lmodname(R)    RC=rc
```

The RC=*rc* comment must be separated from the control statement operands by one or more blanks, and the comment must not extend beyond column 71 (a nonblank character in column 72 indicates statement continuation). Also, no blanks may precede or follow the equal (=) sign.

If, while scanning the link edit control statements for a load module, SMP/E finds the RC=*rc* comment on the NAME statement, SMP/E saves the specified value in the LMOD entry as the RETURN CODE subentry, creating the subentry if it does not exist or replacing any existing RETURN CODE subentry. If SMP/E finds no RC=*rc* comment, SMP/E will neither create nor change the RETURN CODE subentry in the LMOD entry.

The RETURN CODE subentry cannot be removed from an LMOD entry using JCLIN. To do this, either the load module must be deleted and then redefined without the subentry, or the subentry must be deleted using UCLIN.

The RC=*rc* comment on the NAME statement is the only method available to define a RETURN CODE subentry value within JCLIN. Therefore, if you wish to define a RETURN CODE subentry value within JCLIN, you must also define the load module's name using the NAME control statement. Load modules whose names are defined using the member name of the SYSLMOD DD statement, the MOD=*name* operand of the LINKS procedure specification, or in a COPY step, cannot also define a RETURN CODE subentry value within their JCLIN.

ORDER statement

If a specific order of CSECTs within a load module is necessary, ORDER statements are required to define the load module structure. Simply ordering the INCLUDE statements is not sufficient, because SMP/E does CSECT replacements when relinking the load module and, therefore, changes the order of the CSECTs.

REPLACE statement

REPLACE statements are saved in the LMOD entry and are associated with the DLIB module name found on the next INCLUDE statement in the JCL. If the same INCLUDE statement is processed later by JCLIN, REPLACE statements already in the LMOD entry associated with this INCLUDE statement are deleted and replaced by any associated REPLACE statements in the latest job. REPLACE statements are passed to the linkage editor only when the associated DLIB module is to be replaced in the load module.

SETSSI statement

SETSSI statements should not be used in JCLIN data. They are produced as part of servicing a module. If found in the JCLIN, they are stored in the LMOD entry and can result in incorrect data being stored during the application of service.

SYSDEFSD DD statement

SMP/E uses the SYSDEFSD DD statement to determine the side deck library for a DLL load module. SMP/E determines the ddname for the side deck library by using the lowest-level qualifier of the data set name specified in the SYSDEFSD DD statement. This ddname is saved as the SIDE DECK LIBRARY subentry in the LMOD entry.

The definition side deck for a DLL load module contains IMPORT statements for exported symbols and is saved in the side deck library by the link-edit utility during bind operations. The link-edit utility requires a SYSDEFSD data set during link edit operations whenever symbols are to be exported.

In some cases, you might not need to retain the IMPORT statements and therefore you might not want to save the definition side deck. To meet the requirements of the link-edit utility without saving the definition side deck, define SYSDEFSD as DD DUMMY. You can define a dummy side deck library by specifying the SYSDEFSD DD statement in the JCLIN input stream in any of the following ways:

```
//SYSDEFSD DD DSN=SMPDUMMY,DISP=xxx
-or-
//SYSDEFSD DD DSN=NULLFILE
-or-
//SYSDEFSD DD DUMMY
```

In each case, the SIDE DECK LIBRARY subentry for the load module will be set to SMPDUMMY. The value SMPDUMMY is treated as a special case by SMP/E and is always allocated as a dummy data set when preparing for link edit operations.

For an example, see “Example 8: JCLIN for SIDEDECKLIB subentries” on page 168.

SYSLIB DD statement

Typically, SYSLIB DD statements should not be used in JCLIN data. However, they can be used for load modules needing to implicitly include modules from other products. Such load modules are commonly used by products that:

- Are written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) that are owned by a different product
- Make use of a callable-services interface provided by another product
- Need to include stub routines or interface modules from different products that may reside in other zones

For such load modules, the SYSLIB DD statement should specify all the automatic call libraries SMP/E is to use when linking the load module. (These libraries should be target libraries.) The low-level qualifier of each data set specified in the SYSLIB concatenation is saved as a CALLLIBS subentry for the associated load module. For SMP/E to link implicitly-included modules from these libraries, the user must provide DDDEF entries for the libraries in the zone containing the LMOD entry.

SYSLIB DD statements are processed only if the CALLLIBS operand is specified on the JCLIN command or ++JCLIN MCS, or if //*CALLLIBS=YES is encountered after a job card preceding a link-edit step. If the CALLLIBS operand or the CALLLIBS comment is not specified, SMP/E ignores any SYSLIB DD statements it encounters.

Including Pathnames in a SYSLIB Concatenation: A DD statement in a SYSLIB concatenation can include the PATH operand to specify a pathname as an automatic call library. A LIBRARYDD comment statement must immediately follow this DD statement and specify the ddname to be associated with that pathname. SMP/E saves the ddname specified on the LIBRARYDD comment statement as part of the CALLLIBS list in the LMOD entry being updated or created. For an example, see “Example 7: JCLIN for load modules residing in a UNIX file system” on page 167.

Note:

1. If a DD statement in the concatenation comes between the DD statement specifying the PATH operand and the LIBRARYDD comment statement, the misplaced DD statement is ignored.
2. If the DD statement specifying the PATH operand is followed by a JCL statement other than a LIBRARYDD comment statement or a continuation DD statement for the SYSLIB concatenation, the LMOD entry is not updated or created. In addition, if the JCLIN was specified in a SYSMOD (instead of being processed by the JCLIN command), processing for that SYSMOD fails.

SYSLMOD DD statement

SMP/E uses either the SYSLMOD DD statement or the NAME statement to determine the target library for the load module, as follows:

- If a SYSLMOD DD statement is present, SMP/E determines the target library ddname by using the lowest-level qualifier of the data set name specified in the SYSLMOD DD statement.
- If no SYSLMOD DD statement is present, SMP/E determines the name by looking at the NAME=*dsname* option on the procedure statement. The ddname used is the lowest-level qualifier of the data set name specified in the NAME option.
- If no SYSLMOD DD statement or NAME=*dsname* value is found, SMP/E issues an error message.

The ddname of the target library is saved as the SYSLIB value in the LMOD entry for the load module.

A SYSLMOD DD statement can include the PATH operand to specify a pathname for installing a load module in a UNIX file system. A LIBRARYDD comment statement must immediately follow this DD statement and specify the ddname to be associated with that pathname. SMP/E saves the ddname specified on the LIBRARYDD comment statement as a SYSLIB subentry in the LMOD entry being updated or created. For an example, see "Example 7: JCLIN for load modules residing in a UNIX file system" on page 167.

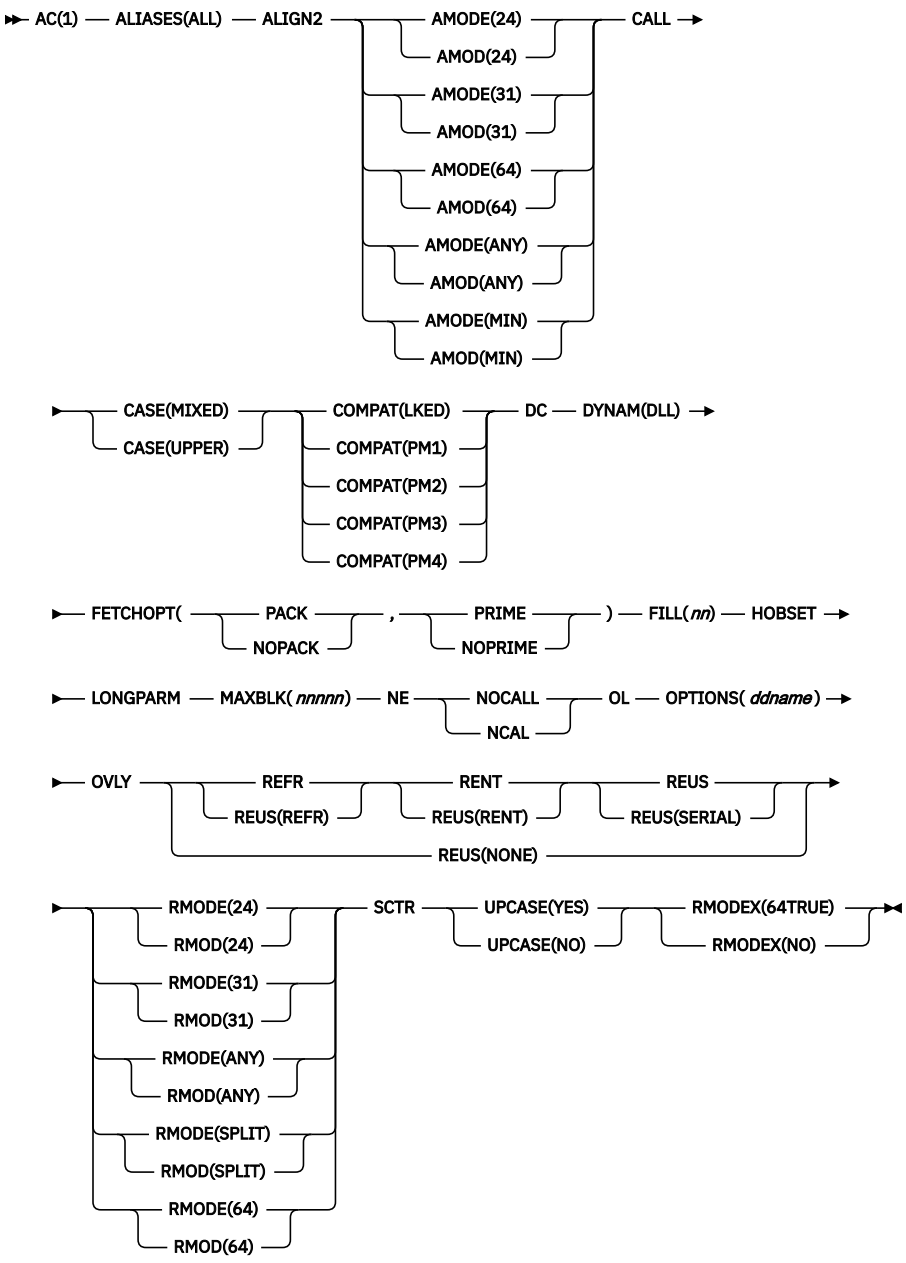
Note:

1. If the DD statement specifying the PATH operand is followed by a JCL statement other than a LIBRARYDD comment statement, the LMOD entry is not updated or created. In addition, if the JCLIN was specified in a SYSMOD (instead of being processed by the JCLIN command), processing for that SYSMOD fails.
2. An LMOD entry can have at most two SYSLIB subentries. If the LMOD entry already contains two SYSLIB subentries, SMP/E replaces the second SYSLIB ddname with the ddname found on the SYSLMOD DD statement, the NAME=*dsname* option, or the LIBRARYDD comment statement.

All other statements found in link-edit input
All other link-edit control statements found are saved in the LMOD entry in the order they are encountered, and are passed to the linkage editor whenever SMP/E needs to relink this load module.

SMP/E then scans the link-edit JCL for the link-edit attributes used to link this load module.

These are the link-edit attributes SMP/E recognizes in the PARM field and saves for future processing:



When none of the previously listed 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.

Note:

1. RMODE(31) is a synonym for RMODE(ANY).
2. The OPTIONS attribute is recognized and processed, but it is not saved as part of the LMOD entry or the MOD entry being processed. It is used as a pointer to an imbedded file containing additional option specifications, allowing the PARM string to exceed the 100-character limit.
3. All LEPARM attributes may also be specified in the format '*attribute=value*'. For example, FILL(*nn*) may also be specified as FILL=*nn*.
4. For more information on the previously listed attributes, see *z/OS SMP/E Reference*.
5. For more information on which attributes you can use with a specific link-edit utility, see *z/OS MVS Program Management: User's Guide and Reference*.

The LMOD entry constructed in this way contains the load module name, the libraries it resides in, the link-edit attributes, and the link-edit control statements required to relink the load module.

If SMP/E is creating an LMOD entry and finds an existing LMOD entry for the same load module containing only cross-zone subentries (a *stub* entry):

- SMP/E issues messages indicating that the cross-zone relationship might no longer be valid, and then deletes the cross-zone subentries from the LMOD entry.
- If deleting these cross-zone entries eliminates a TIEDTO relationship with a cross-zone, SMP/E deletes the associated TIEDTO value from the TARGETZONE entry for the set-to zone.
- Entries for related cross-zone modules are **not** updated to indicate that they are no longer part of the load module in the set-to zone, and the cross-zone's TIEDTO values are not updated.

Processing update steps

Update steps are identified by one of these EXECs:

- EXEC PGM=IEBUPDTE
- EXEC PGM=*updpgm* [if the JCLIN specified UPDATE(PGM=*updpgm*)]
- EXEC *updproc* [if the JCLIN specified UPDATE(*updproc*)]

When SMP/E recognizes an UPDATE step, it skips all the JCL until the SYSIN DD statement or the next EXEC statement is encountered. If the SYSIN statement is found, SMP/E skips all further input until one of the following is found:

- /* or // is found if `SYSIN DD *` was specified.
- /* is found if `SYSIN DD DATA` was specified.
- *xx* is found if `SYSIN DD DATA,DLM=`*xx* was specified, where *xx* can be any two characters.

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

Processing other utility steps

When SMP/E encounters an EXEC statement that does not specify one of the programs or cataloged procedures it recognizes, it skips all further JCL statements until the next EXEC statement is found. Input for certain system utility programs is not meant to be processed as JCLIN. However, the utility must be defined to SMP/E if it falls into any of these categories:

- Assembler
- Link-edit utility
- Copy
- Update

Zone and data set sharing considerations

The following identifies the phases of JCLIN processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

2. JCLIN processing

   Target zone
   Update with exclusive enqueue.

3. Termination

All resources are freed.

Chapter 10. The LINK LMODS command

The LINK LMODS command helps you relink load modules when implicitly-included modules in a particular library are updated. The LINK LMODS command will relink load modules that have a CALLLIBS subentry list in their LMOD entry (and therefore use the automatic library call option to implicitly include modules from a specified library).

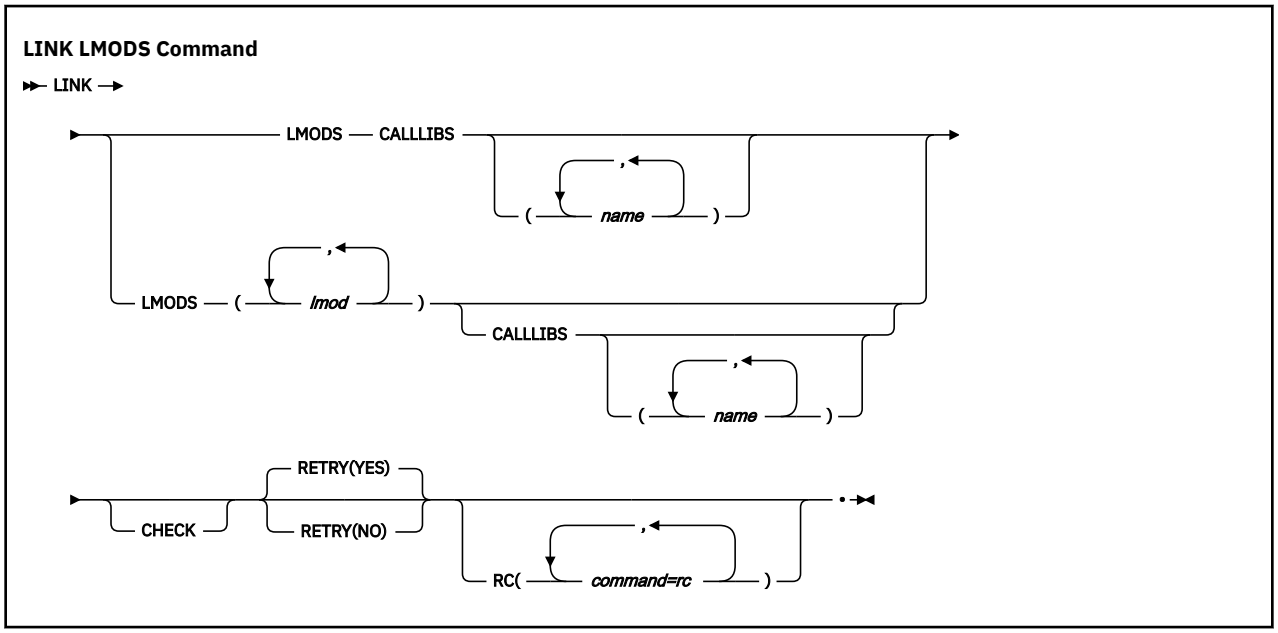
The LINK LMODS command may also be used to relink specific load modules that do not contain CALLLIBS, rebuilding them from scratch, if necessary.

For information on the rebuilding of load modules and the effects of CALLLIBS on this process, see Appendix C, “Building load modules,” on page 547.

Zones for SET BOUNDARY

For the LINK LMODS command, the SET BOUNDARY command must specify the target zone containing the LMOD entries for the load modules to be link-edited.

Syntax



Operands

LMODS
specify the load modules that should be relinked. Load modules specified on the LMODS operand are relinked in addition to any that matched the criteria of the CALLLIBS operand.

If you want to relink a particular set of load modules, specify their names on the LMODS operand.

If you want to relink all load modules whose LMOD entries contain a CALLLIBS subentry list, do not specify any names on the LMODS operand.

LMODS is a required operand for the LINK LMODS command, and it is mutually exclusive with MODULE, FROMZONE and INTOLMOD.

CALLLIBS
requests relinks for load modules whose LMOD entries contain a CALLLIBS subentry list. The specified name is a DDDEF named in an LMOD entry's CALLLIBS subentry list.

Load modules that match the criteria of the CALLLIBS operand are relinked in addition to any load modules specified on the LMODS operand.

If you want to relink only those load modules with particular CALLLIBS subentries, specify CALLLIBS and the specific names. If you want to relink all load modules with CALLLIBS subentries, specify CALLLIBS without any names.

CALLLIBS is mutually exclusive with MODULE, FROMZONE and INTOLMOD. CALLLIBS is a required operand for the LINK LMODS command when the LMODS operand is specified without a list of load module names. If a list of load module names is specified on the LMODS operand, then the CALLLIBS operand is not required.

CHECK
indicates that SMP/E should not relink any load modules. Instead, it should just take these actions:

- Test for errors other than those that occur when the load modules are actually relinked.
- Report on which load modules were relinked.

CHECK is mutually exclusive with MODULE, FROMZONE and INTOLMOD.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the LINK LMODS command.

Before SMP/E processes the LINK LMODS command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the LINK LMODS command. Otherwise, the LINK LMODS command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the LINK LMODS command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RETRY
indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

YES
indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If the retry attempt fails, the resulting *x*37 abend is treated as an unacceptable utility return code. In this case, processing continues for updates to other libraries, but processing fails for unprocessed updates for the out-of-space library.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if YES is specified.

NO
indicates that SMP/E should not try to recover from the error.

Data sets used

These data sets might be needed to run the LINK LMODS command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPPARM	SMPWRK3	SYSUT4
SMPCSI	SMPPTS	SYSLIB	Distribution library
SMPLOG	SMPRPT	SYSPRINT	Link library
SMPLOGA	SMPSNAP	SYSUT1	Target library
SMPLTS	SMPSTS	SYSUT2	Text library
SMPMTS	SMPTLIB	SYSUT3	*zone*
SMPOUT			

Note:

1. The SMPLTS data set is required only when a load module with CALLLIBS is being processed.
2. The SMPMTS, SYSLIB concatenation, and SMPSTS data sets are needed only when assemblies must be done for a module so that it can be included in a load module.
3. SMPPARM is required only if exit routines have been defined in SMPPARM member GIMEXITS.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

Output

The File Allocation report and the LINK LMODS Summary report are produced during LINK LMODS processing. These reports are described in Chapter 34, “SMP/E reports,” on page 457.

Processing

To process the LINK LMODS command, SMP/E checks to ensure that the syntax used for the LINK LMODS command is valid. SMP/E first checks for mutually exclusive operands. The following operands are mutually exclusive, and will cause the command to fail when specified together:

- LMODS
- CALLLIBS
- CHECK

with

- MODULE
- FROMZONE
- INTOLMOD

The next check done is for required operands. When CHECK is specified, LMODS is required. When CALLLIBS is specified, LMODS is required. The CALLLIBS operand is only required if the LMODS operand was specified without any values.

LMOD applicability

After the LINK LMODS command has been successfully syntax checked, SMP/E builds a candidate list of load modules to be relinked.

If specific load modules are listed on the LMODS operand, SMP/E will start with those load modules. If the CALLLIBS operand was specified, SMP/E will add those LMODs that meet the CALLLIBS specification. SMP/E will search the zone for all load modules containing CALLLIBS subentries. These load modules are added to the candidate list of load modules to be relinked if either of the following is true:

- No values were specified on the CALLLIBS operand on the LINK command.
- Values were specified, and at least one value that was specified on the CALLLIBS operand for the LINK LMODS command matches a value in the load module's CALLLIBS subentry list.

Once the list of candidate LMODS has been built, SMP/E will verify that the list is not empty.

Processing LMODs with CALLLIBS and XZMOD subentries

SMP/E determines, from the candidate list of load modules to be relinked, which load modules contain both CALLLIBS and XZMOD subentries. For such load modules, SMP/E saves the "base version" of the load module in the SMPLTS data set.

Before processing load modules with XZMOD subentries, SMP/E allocates the SMPLTS data set. Once the SMPLTS has been allocated, SMP/E allocates the SYSLIBs for the load module. After the SYSLIBs have been allocated, SMP/E allocates the CALLLIBS concatenation for the load module.

After all data sets have been allocated, SMP/E verifies that the load module exists in the SMPLTS data set. If the load module is not found in the SMPLTS data set, SMP/E rebuilds the load module. If the load module must be rebuilt, the load module will be link-edited into the SMPLTS and its target libraries.

For a load module that is not being rebuilt, SMP/E will relink the load module into its SYSLIB data sets, provided that all checks were successful. To relink the load module, SMP/E includes the base version from the SMPLTS data set. It passes the CALL parameter to the link-edit utility, along with the SYSLIB allocation that was allocated using the LMOD's CALLLIBS subentries. Link-edits with similar attributes are batched together.

Processing LMODs with CALLLIBS but no XZMOD subentries

SMP/E determines, from the candidate list of load modules to be relinked, which load modules have CALLLIBS subentries, but do not have XZMOD subentries. For each such load module:

- If the set-to zone has an UPGLEVEL subentry, SMP/E rebuilds the load module from scratch and relinks it. SMP/E deletes the base version from the SMPLTS data set, if one exists.
- If the set-to zone does **not** have an UPGLEVEL subentry, SMP/E relinks the base version found in the SMPLTS, if one exists. If the base version is not found in SMPLTS, SMP/E rebuilds the load module from scratch and relinks it. SMP/E saves a copy of the rebuilt load module in SMPLTS.

Processing LMODs without CALLLIBS subentries

SMP/E will determine which load modules do not have CALLLIBS subentries from the candidate list of load modules to be relinked. These load modules will be rebuilt from scratch. If the load module exists in any of its target libraries (SYSLIBs), the load module is still rebuilt.

When link-editing the load module, the original load module is only included if it was found in the target library, and the load module contains cross-zone modules. If the load module does not contain cross-zone modules, the original load module is not included. The members containing usable copies of modules for the load module will be INCLUDEd ahead of the INCLUDE for the old copy of the load module.

Before processing these load modules, SMP/E will allocate the SYSLIBs for the load module. Once all checks are successful, SMP/E will rebuild and relink the load module.

Scheduling and batching link-edits

The LINK LMODS command will batch link-edits to reduce the number of invocations of the link-edit utility. Link-edits for load modules will be batched if all of the following criteria are met:

- The load modules reside in the same target libraries with the same side-deck libraries
- The load modules have the same link-edit attributes
- The load modules have the same set of CALLLIBS or the load modules do not have CALLLIBS subentries

SMP/E determines which of the load modules being updated require CALLLIBS to be allocated during their link-edit. SMP/E determines this by checking the corresponding LMOD entry for a CALLLIBS subentry list. Link-edits that use a CALLLIBS allocation will be done after link-edits that do not use a CALLLIBS allocation. This is done so that updates to target libraries, which may be part of a CALLLIBS allocation, are processed first.

There are two special cases that can occur when SMP/E determines the scheduling of link-edits that require a CALLLIBS allocation:

1. Two load modules both specify a CALLLIBS allocation. Assume the names of the two load modules are A and B. Load module A specifies in its CALLLIBS allocation load module B's SYSLMOD data set (target library). In this case, SMP/E schedules the link-edit for load module B first.
2. Two load modules both specify a CALLLIBS allocation. Assume the names of the two load modules are C and D. Load module C specifies in its CALLLIBS allocation load module D's SYSLMOD data set. Load module D specifies in its CALLLIBS allocation load module C's SYSLMOD data set. In this case, SMP/E can not determine which link-edit should be performed first. SMP/E arbitrarily chooses to link-edit one of the load modules first.

Zone and data set sharing considerations

The following identifies the phases of LINK LMODS processing, and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, See Appendix B, “Sharing SMP/E data sets,” on page 543.

The following are needed for the LINK LMODS command:

1. Initialization
   - Global zone - read without enqueue
   - Set-to target zone - read without enqueue
2. LINK LMODS processing
   - Global zone - read without enqueue
   - Set-to target zone - update with exclusive Enqueue
   - SMPPTS - read with shared enqueue
   - DLIB zone related to set-to zone - read with shared enqueue
3. Termination
   - All resources are freed

Chapter 11. The LINK MODULE command

Products sometimes contain modules from other products. For example, a product may need to:

- Include another product's modules in its load modules.

  In this case, as long as the two products are in the same zone, SMP/E can automatically include the required modules in the load modules that need them (if the modules reside in the target library as single-CSECT load modules). SMP/E also tracks the inclusion of these cross-product modules in the load modules.
- Update another product's load module with one of its modules.

  In this case, as long as the two products are in the same zone, SMP/E can automatically relink the load module and include the supplied module. SMP/E also tracks the inclusion of the modules in the cross-product load module.

When such products reside in different zones, however, SMP/E cannot automatically perform the cross-zone link-edits. Instead, you can use the LINK MODULE command to perform these cross-zone link-edits as postinstallation steps within SMP/E control. The LINK MODULE command causes the required load modules in one zone to be linked with modules residing in another zone, and tracks this inclusion so subsequent APPLY and RESTORE processing can automatically maintain the affected load modules.

Note:

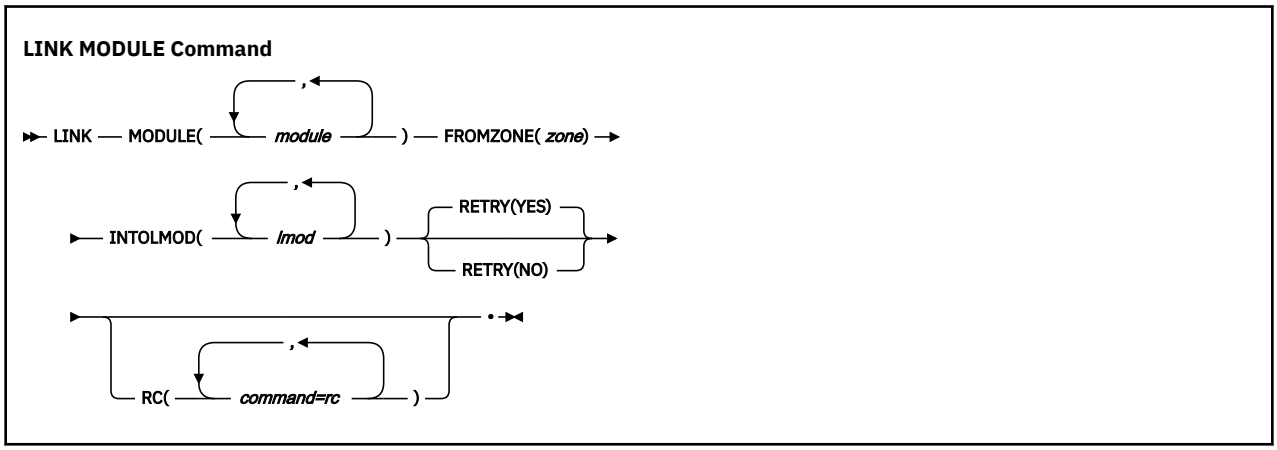
1. The zones used by the LINK MODULE command must be defined in the same global zone.
2. When SMP/E processes the LINK MODULE command, it assumes that adding the desired modules to the load modules does not require any changes to the load module definition (that is, the link-edit utility control statements or link-edit utility attributes). If any such changes are needed, make them through JCLIN before using the LINK MODULE command.
3. There are times when the LINK MODULE command is **not** appropriate to use—generally, for products written in a high-level language and, as a result, include modules from libraries (such as compiler libraries) owned by a different product. Your options for installing such a product depend on how the product was packaged.
   - SYSLIB DD statements are used in link-edit steps to implicitly include the necessary modules.

     In this case, when you install the product, the implicitly-included modules are automatically linked into the load modules. If the libraries containing those modules are updated, you can use the LINK LMODS command to rebuild the affected load modules. For more information, see the Chapter 10, “The LINK LMODS command,” on page 191.
   - No SYSLIB DD statements are used in link-edit steps in order to implicitly include the necessary modules. In this case, you must use postinstallation link-edit steps outside of SMP/E.

Zones for SET BOUNDARY

For the LINK MODULE command, the SET BOUNDARY command must specify the target zone containing the LMOD entries for the load modules to be link-edited.

Syntax



Operands

FROMZONE
specifies the zone containing the modules specified on the MODULE operand. The specified zone must be a target zone and must **not** be the same as the zone specified on the preceding SET command.

Note:

1. FROMZONE is a required operand for the LINK MODULE command.
2. The zone specified on FROMZONE must be defined in the same global zone as the zone specified on the preceding SET command.

INTOLMOD
specifies the load modules that should be link-edited to include the modules specified on the MODULE operand. These load modules must be defined in the zone specified on the preceding SET command.

Note:

1. INTOLMOD is a required operand for the LINK MODULE command.
2. Do not specify a copied (single-CSECT) load module. You cannot use the LINK MODULE command to add modules to a single-CSECT load module; such load modules do not have any link-edit utility control statements that allow for the proper management of a multiple-module load module.

MODULE
specifies the modules that should be linked into the load modules specified on the INTOLMOD operand.

Note:

1. MODULE is a required operand for the LINK MODULE command.
2. Do not specify a module having the same name as a module installed in the set-to zone that is already part of the load module being updated.
3. You can specify a module that is from a totally copied library or that is a single-CSECT load module. However, you cannot specify a module that needs to be assembled.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the LINK MODULE command.

Before SMP/E processes the LINK MODULE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the LINK MODULE command. Otherwise, the LINK MODULE command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand **must be the last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the LINK MODULE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RETRY
indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

YES
indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If the retry attempt fails, the resulting *x*37 abend is treated as an unacceptable utility return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing,see *z/OS SMP/E User's Guide*. For more information about OPTIONS entries, see *z/OS SMP/E Reference*.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if YES is specified.

NO
indicates that SMP/E should not try to recover from the error.

Data sets used

The following data sets might be needed to run the LINK MODULE command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

Distribution library	SMPMTS	SMPSTS	SYSUT2
Link library	SMPOUT	SMPTLIB	SYSUT3
SMPCNTL	SMPPARM	SMPWRK3	SYSUT4
SMPCSI	SMPPTS	SYSLIB	Target library
SMPLOG	SMPRPT	SYSPRINT	Text library
SMPLOGA	SMPSNAP	SYSUT1	*zone*
SMPLTS			

Note:

1. The SMPLTS data set is required only if a load module with CALLLIBS subentries is specified on the INTOLMOD operand of the LINK MODULE command.
2. *Distribution library* represents the DD statements required for each distribution library required to provide modules for the link-edits.
3. *Target library* represents the DD statements required for each target library required to provide modules or load modules for the link-edits.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated using the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
5. SMPPARM is required only if exit routines have been defined in SMPPARM member GIMEXITS.

Output

The File Allocation report is produced during LINK MODULE command processing. This report is described in Chapter 34, “SMP/E reports,” on page 457.

Example: Linking a GDDM module into a CICS load module

Assume you have installed GDDM and CICS®, and some of the GDDM modules must be linked into CICS load modules. GDDM resides in zone GDDTZN, and the zone controlling CICS is CICTZN. Because GDDM and CICS are controlled by different zones, SMP/E does not automatically link the GDDM modules into the CICS load modules when GDDM is installed. The LINK MODULE command can be used instead.

In this example, GDDM module ADMABCD needs to be linked into CICS load module DFHWXYZ. Module ADMABCD is installed in a target library as a single-CSECT load module when GDDM is installed. Therefore, SMP/E can use the target library version of ADMABCD to update CICS load module DFHWXYZ. (If a module does not reside in a target library as a single-CSECT load module, SMP/E uses the related distribution zone copy of the module to update the load module.)

The following commands can be used to have SMP/E install and track the installation of GDDM module ADMABCD in the CICS load module:

```
SET   BDY(CICTZN)          /* Target zone for CICS.      */.
LINK  MODULE(ADMABCD)      /* Link module ADMABCD        */
      FROMZONE(GDDTZN)     /* residing in zone GDDTZN    */
      INTOLMOD(DFHWXYZ)    /* into load module DFHWXYZ.  */.
```

These commands cause GDDM module ADMABCD to be linked into CICS load module DFHWXYZ. SMP/E also adds cross-zone subentries to the affected entries:

- An XZLMOD subentry is added to the ADMABCD MOD entry in target zone GDDTZN so that if ADMABCD is updated, it can be automatically replaced in the CICS load module.

  Note: The CICS load module is automatically updated **only** if the XZLINK subentry was previously set to AUTOMATIC in the TZONE entry for zone CICTZN. Here is an example of the commands that can be used to do this:

  ```
  SET  BDY(CICTZN)         /* Target zone for CICS.    */.
  UCLIN.
  ADD  TZONE(CICTZN)       /* Update TZONE entry       */
       XZLINK(AUTOMATIC). /* to do automatic links.   */.
  ENDUCL.
  ```

- An XZMOD subentry is added to the CICS DFHWXYZ LMOD entry in target zone CICTZN to indicate that:
  - DFHWXYZ now contains ADMABCD.
  - Any updates for ADMABCD should be accepted **only** from zone GDDTZN.
- TIEDTO subentries are added to the TZONE entries for CICTZN and GDDTZN to indicate that there is a relationship between modules and load modules in these zones.

Processing

To process the LINK MODULE command, SMP/E first ensures that the syntax used for the LINK MODULE command is valid. Next, SMP/E checks whether both the zone specified on the FROMZONE operand and the zone specified on the preceding SET command are target zones. If they are, SMP/E obtains the CSIs containing the zones for update processing with an exclusive enqueue. Once SMP/E has obtained access to the CSI data sets, it opens the zones for update mode.

If SMP/E encounters any of the following errors, the LINK MODULE command fails:

- The LINK MODULE command contains syntax errors.
- Either the zone specified on the FROMZONE operand or the zone specified on the SET command (or both) is not a target zone.

- Errors were encountered while SMP/E was acquiring the CSIs or opening the zones.

SMP/E also performs a required operand check for the LINK MODULE command. If FROMZONE is specified, MODULE and INTOLMOD are both required. If MODULE is specified, FROMZONE and INTOLMOD are both required. Finally, if INTOLMOD is specified, FROMZONE and MODULE are both required.

When all checks are satisfied, SMP/E prepares to link the load modules and invokes the link-edit utility to do so.

Preparing for linking

Before invoking the link-edit utility, SMP/E checks whether it has access to the necessary modules, load modules, and libraries. If SMP/E encounters any errors that could cause the LINK MODULE command to fail, LINK MODULE command processing stops. Otherwise, SMP/E goes on to link-edit the load modules.

Obtaining the required modules

To find a usable copy of each module specified on the MODULE operand, SMP/E checks the MOD entries in the target zone specified on the FROMZONE operand. SMP/E determines whether the modules have been installed in a target library (and are available for linking) by checking whether each MOD entry has both an FMID and an RMID value. If so, SMP/E checks the target zone MOD entry to determine whether there is a stand-alone version of the module that can be used for the link-edit. A stand-alone version exists in either of these cases:

- The module was copied into a target library from a distribution library. (It is in a copied library.)
- The module exists by itself in a load module in a target library. (It is a single-CSECT load module.)

If SMP/E finds a stand-alone copy of the module, it saves the name of its target library for subsequent use in link-edit processing.

If SMP/E cannot find a stand-alone copy of a module in a target library, it checks the related distribution zone to determine whether a distribution library contains a usable copy. If there is a MOD entry in the distribution zone, SMP/E checks whether it contains both an FMID and an RMID value. If so, the module has been accepted into a distribution library, and a usable copy for the link-edit exists. SMP/E saves the name of the distribution library for subsequent use in link-editing the load module. In addition, SMP/E compares the distribution zone MOD entry to the related target zone MOD entry to determine whether they are at the same level. If they are at different levels, a warning message is issued, but the distribution zone copy of the module is still used for the link-edit.

Note:

1. If the two copies of the module are at different levels, you may want to synchronize the target zone version of the module with the distribution zone version. You can do this by accepting any applied SYSMODs that affect the module, or by restoring applied SYSMODs. Once the synchronization is done, you may want to use the LINK MODULE command again to relink the module into the load module.
2. SMP/E does not assemble a module in order to use it with the LINK MODULE command. The module must exist as a load module so it can serve as input to the link-edit utility.
3. If the load module already contains a copy of the module (for example, as a result of previous LINK MODULE command processing), SMP/E does not check whether the copy of the module about to be linked into the load module is at an equal or higher level than the previous copy.

The following types of errors cause the LINK MODULE command to fail:

- No MOD entry was found in the FROMZONE target zone for a module specified on the MODULE operand.
- A module specified on the MODULE operand was not installed into a target library (its MOD entry in the FROMZONE target zone is missing either an FMID, an RMID, or both).
- There is no stand-alone version of a module in the target library, and one of the following errors was also found:
  - No related distribution zone is defined for the FROMZONE target zone.

- SMP/E cannot obtain access to the related distribution zone.
- No MOD entry exists in the distribution zone.

Obtaining the required load modules

SMP/E checks the LMOD entries for the load modules specified on the INTOLMOD operand to determine whether they can be processed by the LINK MODULE command. First, SMP/E verifies that none of the load modules are copied (single-CSECT) load modules. SMP/E then checks whether the LMOD entries already contain XZMOD subentries for any of the modules specified on the MODULE operand. These subentries indicate that the load module contains a module supplied by another zone. If SMP/E finds any such XZMOD entries, it verifies that the zone specified as the original supplier of the module is the same as the current FROMZONE value.

Next, SMP/E ensures that none of the modules specified on the MODULE operand will overlay a module by the same name in the set-to zone that is already part of a load module being updated and that is installed in the target libraries. For each LMOD entry found, SMP/E saves its target library information for subsequent use in the link operation.

The following types of errors cause the LINK MODULE command to fail:

- An LMOD entry does not exist.
- An LMOD entry exists only as XZMOD subentries.
- The load module has been copied (it is a single-CSECT load module).
- An LMOD entry contains an XZMOD subentry for a module specified on the MODULE operand, but the zone specified as the original supplier of the module is different from the current FROMZONE value.
- A module specified on the MODULE operand has the same name as a module from the set-to zone that is already part of a load module being updated and that is installed in the target libraries.

Checking the libraries for the modules and load modules

SMP/E makes sure it can allocate the required libraries (the target libraries for the load modules to be updated and the target or distribution libraries containing the modules to be included). If a DD statement was not specified for a library, SMP/E attempts to allocate it dynamically using the appropriate DDDEF entry, as follows:

- For load module libraries, SMP/E uses the DDDEF entries in the zone specified on the preceding SET command. If an LMOD entry contains a CALLLIBS subentry list, the zone containing the LMOD entry must also contain a DDDEF entry for each CALLLIBS library.
- For module libraries, SMP/E uses the following DDDEF entries:
  - For target libraries, it uses DDDEF entries in the zone specified on the FROMZONE operand.
  - For any necessary distribution libraries, it uses DDDEF entries in the DLIB zone related to the FROMZONE.

Next, SMP/E verifies that all the load modules to be link-edited are actually in the indicated target libraries. If a load module is not in its library, it is not link-edited from that library. If it is not in any of the indicated target libraries, it is not link-edited at all. A load module having both CALLLIBS and XZMOD subentries must exist in the SMPLTS data set, although it is not required that it already exist in the target libraries.

The following types of errors cause the LINK MODULE command to fail:

- A required DD statement is missing and the associated data set cannot be dynamically allocated.
- None of the load modules to be updated are in the indicated target libraries.

Building load modules

If a load module that is to be updated has CALLLIBS subentries, SMP/E will rebuild the load module and save it, first in the SMPLTS data set and then in its true target libraries. For more information on

how SMP/E builds load modules and on the rebuilding of load modules with CALLLIBS, see Appendix C, "Building load modules," on page 547.

Linking the load modules

SMP/E invokes the link-edit utility for each load module to be updated.

For each successful link-edit, SMP/E updates the MOD and LMOD entries involved:

- The MOD entries are updated with XZLMOD subentries to indicate the cross-zone load modules they were linked into. XZLMOD subentries specify the name of the load module and the zone name specified on the previous SET operand.
- The LMOD entries are updated with XZMOD subentries to indicate the cross-zone modules they now contain. XZMOD subentries specify the name of the module and the zone name specified on the FROMZONE operand.

SMP/E adds TIEDTO values to record the relationship between the zone specified on the FROMZONE operand and the zone specified on the preceding SET command:

- The zone name from the SET command becomes a TIEDTO subentry in the TARGETZONE entry for the zone specified on the FROMZONE operand.
- The zone name from the FROMZONE operand becomes a TIEDTO subentry in the TARGETZONE entry for the zone specified on the preceding SET command.

These subentries define the cross-zone relationship between the modules and load modules and are used during APPLY and RESTORE processing to update the load modules when the modules are updated. For more information about these subentries, see *z/OS SMP/E Reference*.

The TIEDTO and XZLMOD subentries created by the LINK MODULE command must be updated if the cross-zone relationship between the modules and load modules is altered by use of the ZONECOPY, ZONEIMPORT, ZONEMERGE, or ZONERENAME commands. For information that can help you determine what action to take, see "Updating cross-zone subentries" on page 408.

Utility failures can cause the LINK MODULE command to fail. For details on handling *x*37 abends, see the description of the RETRY operand under "Operands" on page 198.

Zone and data set sharing considerations

This section shows the phases of LINK MODULE command processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, "Sharing SMP/E data sets," on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Set-to target zone
   Read without enqueue.

   FROMZONE
   Read without enqueue.

2. LINK MODULE processing

   Global zone
   Read without enqueue.

   Set-to target zone
   Update with exclusive enqueue.

   FROMZONE
   Update with exclusive enqueue.

   SMPPTS
   Read with shared enqueue.

DLIB zone related to FROMZONE (as required)
Read with shared enqueue.

DLIB zone related to set-to zone
Read with shared enqueue.

3. Termination

   All resources are freed.

Chapter 12. The LIST command

The SMP/E data sets contain a great deal of information—the global zone, target zones, distribution zones, SMPPTS, SMPLOG, and SMPSCDS—that you may find useful when installing a new function, preparing a user modification, or debugging a problem. You can use the SMP/E LIST command to display that information.

SMP/E can display all the entries of a specified type (such as MOD, MAC, SYSMOD, and so on), or it can display information for selected entries. In addition, for SYSMOD entries, SMP/E provides some additional operands you can specify to list groups of SYSMODs that meet certain criteria.

Zones for SET BOUNDARY

To list entries in a CSI data set, you must specify the name of the zone containing the entries to be listed on the SET BOUNDARY command.

To list entries in a data set other than the CSI (such as the SMPLOG or SMPSCDS), you must specify the zone associated with that data set on the SET BOUNDARY command:

- **SMPLOG:** Specify the zone containing the DDDEF entry for the particular SMPLOG data set to be listed.
- **SMPSCDS:** Specify the target zone containing the DDDEF entry for the particular SMPSCDS data set to be listed.

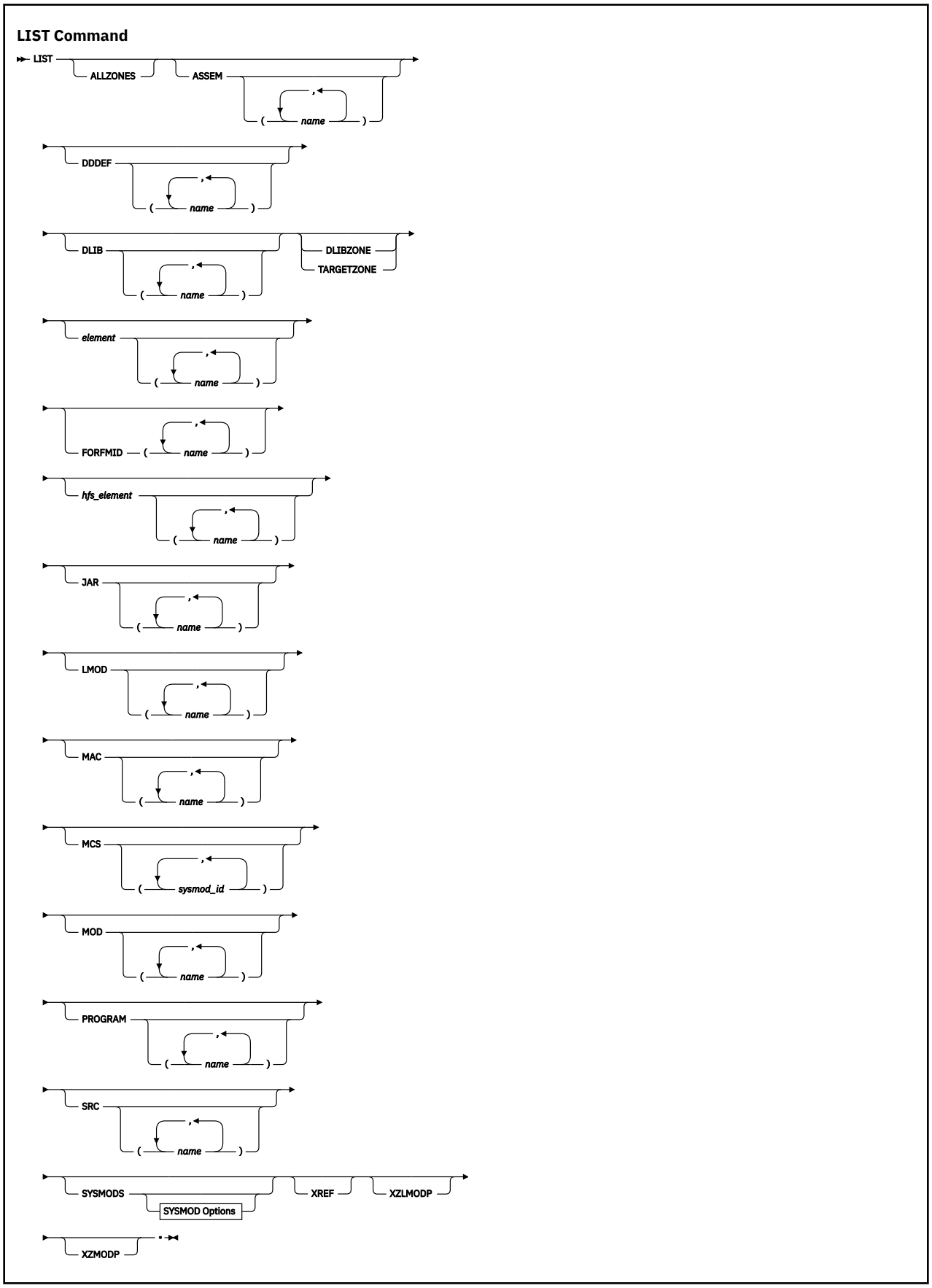
Make sure the data you request to have listed is valid for the specified zone type.

Syntax

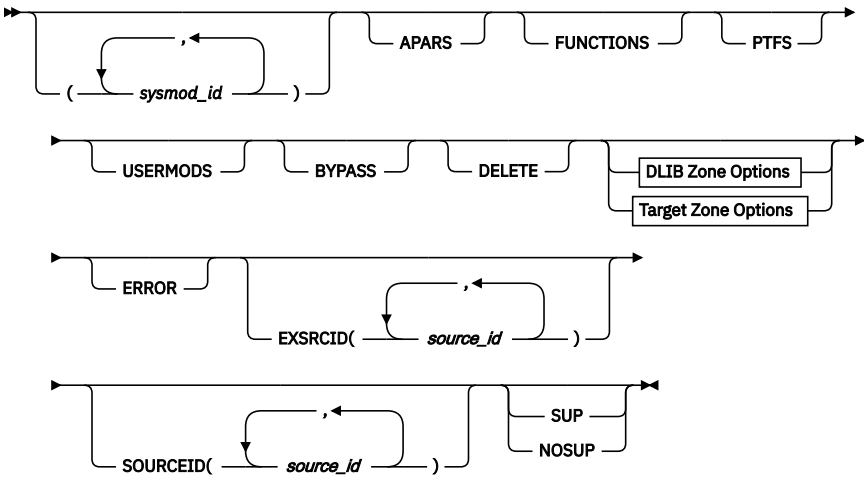
This section shows the LIST command syntax for the following zones and data sets:

- Distribution and target zones
- Global zone
- SMPLOG
- SMPSCDS

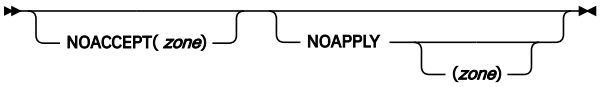
Distribution zone and target zone syntax



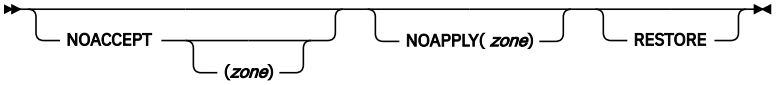
SYSMOD Options



DLIB Zone Options



Target Zone Options



Note:

1. The SYSMODS operand is optional if you specify any of the following operands:

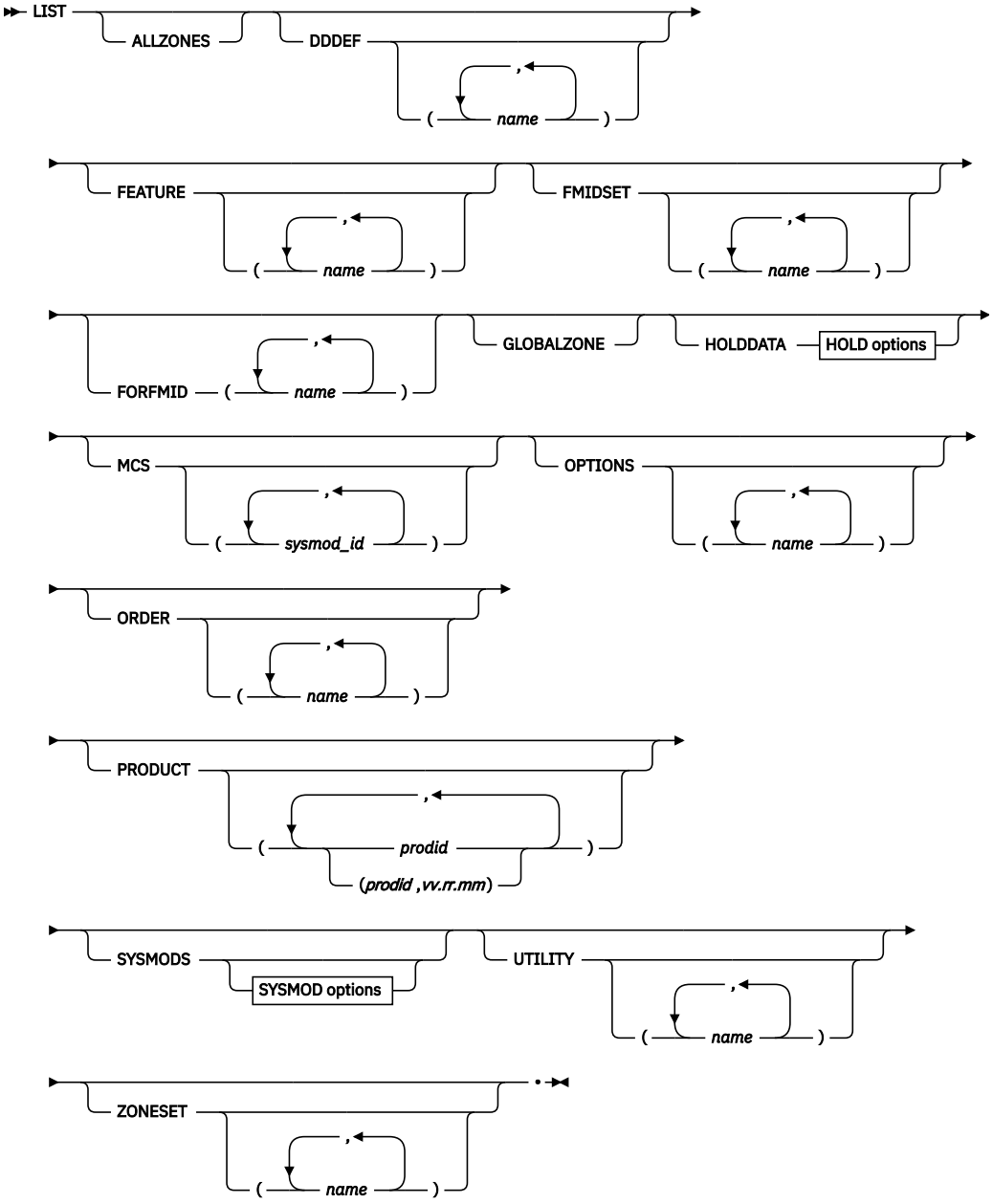
   APARS
   BYPASS
   DELETE
   ERROR
   EXSRCID
   FUNCTIONS
   MCS
   NOACCEPT
   NOAPPLY
   NOSUP
   PTFS
   RESTORE
   SOURCEID
   SUP
   USERMODS

2. The XZLMODP and XZMODP operands are valid only for target zone entries.

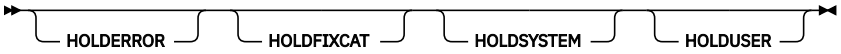
See the operand descriptions for details on all the operands.

Global zone syntax

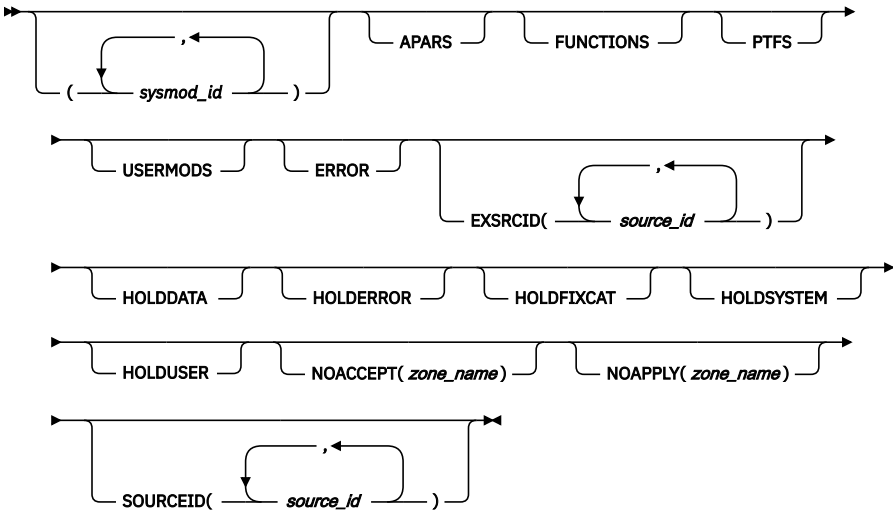
LIST COMMAND



HOLD options



SYSMOD options



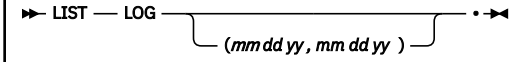
Note: The SYSMODS operand is optional if you specify any of the following operands:

APARS
ERROR
EXSRCID
FUNCTIONS
HOLDERROR
HOLDSYSTEM
HOLDUSER
NOACCEPT
NOAPPLY
PTFS
SOURCEID
USERMODS

See the operand descriptions for details on all the operands.

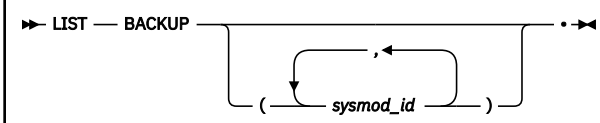
SMPLOG syntax

LIST command



SMPSCDS syntax

LIST command



Operands

ALLZONES
indicates that SMP/E should list information from the global zone and all the target and distribution zones defined by ZONEINDEX subentries.

Note:

1. ALLZONES is mutually exclusive with HOLDDATA, HOLDERROR, HOLDFIXCAT, HOLDSYSTEM, HOLDUSER, MCS, NOACCEPT and NOAPPLY.
2. You can limit the information to be listed by specifying only the entries or entry types that you need. For example:

```
SET     BDY(GLOBAL)    /* set to global zone      */.
LIST    SYSMOD(UZ12345)/* list this SYSMOD entry  */
        ALLZONES       /* from wherever it is     */.
```

   lists SYSMOD entry UZ12345 in each zone to which it has been applied or accepted.
3. ALLZONES is allowed when the SET command specifies the global zone, a target zone, or a distribution zone.

   The entries listed are the same, regardless of the type of zone you specify, because the output is determined by the additional operands on the LIST command and by the entry types valid within each zone to be listed. For example, the following lists module X in all target and distribution zones:

```
LIST ALLZONES MOD(X).
```

   The global zone is skipped, because there are no modules in the global zone.

APARS
indicates that SMP/E should list APAR SYSMODs.

Note:

1. APARS can also be specified as APAR.
2. When APARS is used with FUNCTIONS, PTFS, or USERMODS, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.

ASSEM
indicates that SMP/E should list all ASSEM entries or the specified ASSEM entries.

Note: ASSEM is allowed when the SET command specifies a target zone or distribution zone.

BACKUP
indicates that SMP/E should list all BACKUP entries or the specified BACKUP entries.

Note:

1. BACKUP is mutually exclusive with all other LIST operands.
2. BACKUP is allowed when the SET command specifies a target zone.

BYPASS
indicates that SMP/E should list entries for SYSMODs installed using the BYPASS operand.

Note:

1. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.
2. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

DDDEF
indicates that SMP/E should list all DDDEF entries or the specified DDDEF entries.

DELETE
indicates that SMP/E should list entries for function SYSMODs that have been explicitly deleted from the target zone or distribution zone by other function SYSMODs.

Note:

1. DELETE is allowed when the SET command specifies a target zone or distribution zone.
2. DELETE can also be specified as DEL.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.

DLIB
indicates that SMP/E should list all DLIB entries or the specified DLIB entries.

Note: DLIB is allowed when the SET command specifies a target zone or distribution zone.

DLIBZONE
indicates that SMP/E should list the DLIBZONE entry.

Note:

1. DLIBZONE is allowed when the SET command specifies a distribution zone.
2. DLIBZONE can also be specified as DZONE.

element
is used to list a particular type of data element entry. *element* indicates that SMP/E should list all data element entries of that type or the specified data element entries.

Note:

1. *element* is allowed when the SET command specifies a target zone or distribution zone.
2. "Data Element MCS" in the "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* shows the types of data elements that can be specified for the *element* operand.
3. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain a *xxx* value.) The "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* contains a table that shows the *xxx* values and the languages they represent.

ERROR
indicates that SMP/E should list SYSMOD entries in which the ERROR indicator is set.

Note:

1. ERROR is allowed when the SET command specifies a target zone or distribution zone.
2. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

EXSRCID
indicates that SYSMODs associated with the specified source IDs should **not** be listed.

Note:

1. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.
2. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are excluded.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU* are excluded.

- For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are excluded.
- For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are excluded.

– A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for example, SYSMODs that contain any of these source IDs are excluded: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not excluded.

Any number of asterisks and percent signs can be used within a single partially specified source ID.

The following examples are valid source ID specifications:

```
RSU0709
RSU*
IBM.Device.20%4
IBM.Device.*.zAAP
```

3. A given source ID can be explicitly specified **only once** on the EXSRCID operand.
4. The same source ID **cannot** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified, implicitly or explicitly, on the EXSRCID operand and on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs and at least one of those source IDs is specified implicitly or explicitly on the source ID operand, the SYSMOD is excluded from processing if another one of its source IDs is specified implicitly or explicitly on the EXSRCID operand.

   For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.
7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

FEATURE
indicates SMP/E should list all FEATURE entries or the specified FEATURE entries.

Note:

1. FEATURE is allowed when ALLZONES is specified or the SET command specifies the global zone.
2. FEATURE with the FORFMID operand lists only FEATUREs with the specified FMIDs.

FMIDSET
indicates that SMP/E should list all FMIDSET entries or the specified FMIDSET entries.

Note:

1. FMIDSET is allowed when the SET command specifies the global zone.
2. FMIDSET can also be specified as FMSET.
3. To list element and SYSMOD entries owned by an FMID defined in a particular FMIDSET entry, use the FORFMID operand, not FMIDSET. The FMIDSET operand provides a listing only of the specified FMIDSET entries, not a listing of the entries owned by FMIDs defined in the specified FMIDSET entries.

FORFMID
indicates that SMP/E should list only entries currently owned by one of the specified FMIDs or by an FMID defined in one of the specified FMIDSET entries.

Note:

1. You can specify FMIDs, FMIDSET entries, or both.
2. Only element and SYSMOD entries are listed by the FORFMID operand.

3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not, **unless** an element type operand was also specified. In that case, FORFMID limits the element entries that are listed.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
5. FORFMID with the HOLDDATA operand lists only SYSMODs with the specified FMID that have been received.

FUNCTIONS
indicates that SMP/E should list function SYSMODs.

Note:

1. FUNCTIONS can also be specified as FUNCTION.
2. When FUNCTIONS is used with APARS, PTFS, or USERMODS, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.

GLOBALZONE
indicates that SMP/E should list the GLOBALZONE entry.

Note:

1. GLOBALZONE is allowed when the SET command specifies the global zone.
2. GLOBALZONE can also be specified as GZONE.

hfs_element
is used to list a particular type of hierarchical file system element entry. *hfs_element* indicates that SMP/E should list all hierarchical file system element entries of that type or the specified hierarchical file system element entries.

Note:

1. *hfs_element* is allowed when the SET command specifies a target zone or distribution zone.
2. "Hierarchical File System Element MCS" in the "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* shows the types of hierarchical file system elements that can be specified for the *hfs_element* operand.
3. To list UNIX shell scripts for the zone, enter the LIST command for the *hfs_element* type of SHELLSCR. To list all shell scripts for the zone, specify SHELLSCR by itself. To list only specific shell scripts, include the names of the shell script files with the SHELLSCR operand. An example is shown in “Example 5: List entries for specific UNIX shell scripts” on page 224.
4. Some types of hierarchical file system elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *hfs_element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *hfs_element* operand does not contain any *xxx* value.) The "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* contains a table that shows the *xxx* values and the languages they represent.

HOLDDATA
indicates that SMP/E should list HOLDDATA. How the HOLDDATA is listed depends on whether you specify the SYSMOD operand with the HOLDDATA operand.

- When specified with the SYSMOD operand, HOLDDATA indicates that SMP/E should list only SYSMODs that are held, and should include the ++HOLD modification control statements (HOLDDATA) associated with the SYSMOD entries that are listed. No separate HOLDDATA entries are listed.
- When specified without the SYSMOD operand, HOLDDATA indicates that SMP/E should list all HOLDDATA entries. No SYSMOD entries are listed.

  You can limit which HOLDDATA entries are listed by coding one or more of the following operands:

HOLDERROR
HOLDFIXCAT
HOLDSYSTEM
HOLDUSER

If you specify more than one type of hold, SMP/E lists only entries containing holds for **all** the specified types. For example, the following commands list all HOLDDATA entries with **both** HOLDERROR and HOLDSYSTEM reason IDs:

```
SET      BDY(GLOBAL)    /* set to global zone       */.
LIST     HOLDDATA       /* list only the HOLDDATA   */
         HOLDERROR      /* entries that contain     */
         HOLDSYSTEM     /* both error and system    */
                        /* holds                    */.
```

Note:

1. HOLDDATA is allowed when the SET command specifies the global zone.
2. HOLDDATA with the FORFMID operand lists only SYSMODs with the specified FMID that have been received.
3. Table 15 on page 214 summarizes the LIST results for various combinations of the HOLDDATA operand with other related operands.

Table 15. Information listed for HOLDDATA combined with other operands

HOLD-related operands	Information listed when the SYSMOD operand is specified	Information listed when the SYSMOD operand is not specified
HOLDDATA (without HOLDERROR, HOLDFIXCAT, HOLDSYSTEM, or HOLDUSER)	SYSMOD entries plus all associated ++HOLD statements.	All ++HOLD statements of all types.
HOLDDATA (with HOLDERROR, HOLDFIXCAT, HOLDSYSTEM, or HOLDUSER)	SYSMOD entries for SYSMODs that have the specified HOLDDATA types, plus all associated ++HOLD statements for those SYSMODs.	All ++HOLD statements of the specified types.
HOLDERROR, HOLDFIXCAT, HOLDSYSTEM, or HOLDUSER (without HOLDDATA)	SYSMOD entries for SYSMODs that have the specified HOLDDATA types. No ++HOLD statements included with the SYSMOD entries.	SYSMOD entries for SYSMODs that have the specified HOLDDATA types. No ++HOLD statements are included with the SYSMOD entries.

HOLDERROR

When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDERROR indicates that SMP/E should list only SYSMODs associated with error hold reason IDs. The associated ++HOLD modification control statements are not listed.

Note: If the reason IDs are bypassed or resolved, these SYSMODs might not actually be held during APPLY or ACCEPT processing.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDERROR indicates that HOLDDATA entries for error hold reason IDs should be listed. No SYSMOD entries are listed.

Note:

1. HOLDERROR is allowed when the SET command specifies the global zone.
2. HOLDERROR can also be specified as HOLDERR.

HOLDFIXCAT
When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDFIXCAT indicates that SMP/E should list only SYSMODs associated with the fix category hold reason IDs. The associated ++HOLD modification control statements are not listed.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDFIXCAT indicates that HOLDDATA entries for fix category hold reason IDs should be listed. No SYSMOD entries are listed.

When specified with the HOLDDATA and the SYSMOD operands, HOLDFIXCAT indicates that HOLDDATA entries for fix category hold reason IDs should be listed, and the SYSMOD entries for those held SYSMODs should be listed.

Note:

1. HOLDFIXCAT is allowed only when the SET command specifies the global zone.
2. HOLDFIXCAT and ALLZONES are mutually exclusive.

HOLDSYSTEM
When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDSYSTEM indicates that SMP/E should list only SYSMODs associated with system hold reason IDs. The associated ++HOLD modification control statements are not listed.

Note: If the reason IDs are bypassed or resolved, these SYSMODs might not actually be held during APPLY or ACCEPT processing.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDSYSTEM indicates that SMP/E should list HOLDDATA entries for system hold reason IDs. No SYSMOD entries are listed.

Note:

1. HOLDSYSTEM is allowed when the SET command specifies the global zone.
2. HOLDSYSTEM can also be specified as HOLDSYS.

HOLDUSER
When specified without the HOLDDATA operand (and either with or without the SYSMOD operand), HOLDUSER indicates that SMP/E should list only SYSMODs associated with user hold reason IDs. The associated ++HOLD modification control statements are not listed.

Note: If the reason IDs are bypassed or resolved, these SYSMODs might not actually be held during APPLY or ACCEPT processing.

When specified with the HOLDDATA operand but without the SYSMOD operand, HOLDUSER indicates that HOLDDATA entries for user hold reason IDs should be listed. No SYSMOD entries are listed.

Note: HOLDUSER is allowed when the SET command specifies the global zone.

JAR
indicates that SMP/E should list all Java Archive (JAR) file entries or the specified JAR entries.

LMOD
indicates that SMP/E should list all LMOD entries or the specified LMOD entries.

Note: LMOD is allowed when the SET command specifies a target zone or distribution zone.

LOG
indicates that SMP/E should list either the total contents of the LOG or the contents within a selected date range.

(*mm dd yy*, *mm dd yy*) specifies a range of dates within the data set to be listed. If no date range is specified, the contents of the entire LOG data set are listed.

The dates are specified as *mm dd yy*, where *mm* is the month (01 to 12), *dd* is the day (01 to 31), and *yy* is the year (00 to 99). Blanks separate the month, day, and year.

The following commands list the data in the LOG for June 8 through June 11, 2008:

```
SET      BDY(TGT1)      /* set to target zone       */.
LIST     LOG(06 08 07   /* list log within this     */
             06 11 07)  /* date range               */.
```

These commands list the data in the LOG for one day, June 9, 2008:

```
SET      BDY(TGT1)      /* set to target zone       */.
LIST     LOG(06 09 07   /* list log for this one    */
             06 09 07)  /* day                      */.
```

Note:

1. LOG is mutually exclusive with all other LIST operands.
2. To determine which LOG data set to list, SMP/E checks the SMPLOG DDDEF entry in the zone specified on the SET command.
3. SMP/E views its LOG data set as one "logical" data set, even though there might actually be two separate physical data sets: SMPLOG and SMPLOGA. So, if an SMPLOGA DDDEF is defined in the zone and data has spilled over from the SMPLOG data set into the SMPLOGA data set, LIST LOG also lists the contents of the SMPLOGA data set. You can also specify a date range that spans the SMPLOG and SMPLOGA data sets, or a date range that is only in the SMPLOGA data set, because SMP/E views the two data sets as a single "logical" data set.

MAC
indicates that SMP/E should list all MAC entries or the specified MAC entries.

MCS
specifies that SMP/E should list all or the specified. `LIST MCS` can be used to print PTF cover letters.

- If no SYSMOD IDs are specified, SMP/E lists the modification control statements associated with all the SYSMOD entries in the current zone.
- If SYSMOD IDs are specified, SMP/E lists only the modification control statements for the specified SYSMOD entries. For example, the following commands list only the modification control statements for AZ12345:

```
SET      BDY(TGT1)      /* set to target zone       */.
LIST     APAR           /* list all APAR type       */
         SYSMOD         /* SYSMODs plus             */
         MCS(AZ12345)   /* this one MCS entry       */.
```

MOD
indicates that SMP/E should list all MOD entries or the specified MOD entries.

NOACCEPT
indicates that SMP/E should list SYSMOD entries from the current zone that are **not** accepted into a particular distribution zone. You can use NOACCEPT to:

- See which SYSMODs have been received but have not yet been accepted into the specified distribution zone.

  To do this, specify the global zone on the SET command and the distribution zone you want to check on the NOACCEPT operand.
- See which SYSMODs have been applied in a particular target zone but have not yet been accepted into one of these zones:
  - Its related distribution zone
  - The distribution zone specified on NOACCEPT

  To do this, specify the desired target zone on the SET command and take one of these actions:
  - To check for SYSMODs that have not been accepted into the related distribution zone, specify NOACCEPT without a zone name.

  - To check for SYSMODs that have not been accepted into a particular distribution zone, specify NOACCEPT with the appropriate distribution zone name.
- Compare which SYSMODs are accepted in two distribution zones.

  To do this, specify one distribution zone on the SET command and the other on the NOACCEPT operand. SMP/E lists the SYSMODs that have been accepted into the set-to zone, but not into the NOACCEPT zone.

For examples, see "Examples" on page 223.

Note:

1. NOACCEPT can also be specified as NOACC.
2. If you specify either the global zone or a distribution zone on the SET command, you **must** specify a distribution zone on NOACCEPT.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
5. You can also use the REPORT SYSMODS command to compare zones and to generate the commands needed to install SYSMODs in the zone where they are not installed. See Chapter 20, "The REPORT SYSMODS command," on page 325 for more information.

NOAPPLY
indicates that SMP/E should list SYSMOD entries from the current zone that are **not** applied to a particular target zone. You can use NOAPPLY to:

- See which SYSMODs have been received but have not yet been applied to the specified target zone.

  To do this, specify the global zone on the SET command and the target zone you want to check on the NOAPPLY operand.
- See which SYSMODs have been accepted into a particular distribution zone but have not yet been applied to one of these zones:
  - Its related target zone
  - The target zone specified on NOAPPLY

  To do this, specify the desired distribution zone on the SET command and take one of these actions:
  - To check for SYSMODs that have not been applied to the related target zone, specify NOAPPLY without a zone name.
  - To check for SYSMODs that have not been applied to a particular target zone, specify NOAPPLY with the appropriate target zone name.
- Compare which SYSMODs are applied to two target zones.

  To do this, specify one target zone on the SET command and the other on the NOAPPLY operand. SMP/E lists the SYSMODs that have been applied to the set-to zone but not to the NOAPPLY zone.

For more information, see "Examples" on page 223.

Note:

1. NOAPPLY can also be specified as NOAPP.
2. If you specify either the global zone or a target zone on the SET command, you **must** specify a target zone on NOAPPLY.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

5. You can also use the REPORT SYSMODS command to compare zones and to generate the commands needed to install SYSMODs in the zone where they are not installed. See Chapter 20, "The REPORT SYSMODS command," on page 325 for more information.

NOSUP
indicates that SMP/E should list entries for SYSMODs that have not been superseded.

Note:

1. NOSUP is mutually exclusive with SUP.
2. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

OPTIONS
indicates that SMP/E should list all OPTIONS entries or the specified OPTIONS entries.

Note: OPTIONS is allowed when the SET command specifies the global zone.

ORDER
indicates that SMP/E should list all ORDER entries in the global zone or the specified ORDER entries.

Note: ORDER is allowed when ALLZONES is specified or the SET command specifies the global zone.

PRODUCT
indicates SMP/E should list all PRODUCT entries or the specified PRODUCT entries.

Note: PRODUCT is allowed when ALLZONES is specified or the SET command specifies the global zone.

PROGRAM
indicates that SMP/E should list all program element entries or the specified program element entries.

PTFS
indicates that SMP/E should list PTF SYSMODs.

Note:

1. PTFS can also be specified as PTF.
2. When PTFS is used with APARS, FUNCTIONS, or USERMODS, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD has also been specified, even if it has not.

RESTORE
indicates that SMP/E should list SYSMOD entries in which the RESTORE indicator is set. These SYSMODs have been incompletely restored and are in error.

Note:

1. RESTORE is allowed when the SET command specifies a target zone.
2. RESTORE can also be specified as RES.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

SOURCEID
indicates that SMP/E should list only SYSMOD entries associated with one of the specified SOURCEID values.

Note:

1. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
2. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are selected.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU are selected.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are selected.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are selected.
     - A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for example, SYSMODs that contain any of these source IDs are selected: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not selected.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source IDs:

   ```
   RSU0709
   RSU*
   IBM.Device.20%4
   IBM.Device.*.zAAP
   ```

3. A given source ID can be explicitly specified **only once** on the source ID operand.
4. The same source ID **cannot** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified, implicitly or explicitly, on the SOURCEID operand and also on the EXSRCID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs of which at least one is specified either implicitly or explicitly on the SOURCEID operand and another is specified either implicitly or explicitly on the EXSRCID operand, the SYSMOD is excluded from processing.

   For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.
7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

SRC
indicates that SMP/E should list all SRC entries or the specified SRC entries.

SUP
indicates that SMP/E should list entries for SYSMODs that have been superseded.

Note:

1. SUP is mutually exclusive with NOSUP.
2. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD was also specified, even if it was not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

4. To list "dummy" entries for superseded SYSMODs (entries for SYSMODs that were superseded but not installed), do **not** specify a SYSMOD type operand. No SYSMOD type is associated with such entries.

SYSMODS
indicates that SMP/E should list all SYSMOD entries or the specified SYSMOD entries.

You can limit which SYSMOD entries are listed by coding one or more of the following SYSMOD qualifier operands:

APARS, FUNCTIONS, PTFS, or USERMODS
BYPASS
DELETE
ERROR
EXSRCID
FORFMID
HOLDERROR, HOLDFIXCAT, HOLDSYSTEM, or HOLDUSER
NOACCEPT
NOAPPLY
NOSUP or SUP
RESTORE
SOURCEID

For the operands shown on separate lines, SMP/E lists only SYSMOD entries that meet **all** the specified criteria. For the operands shown on the same line, SMP/E lists SYSMOD entries that meet **any one** of the specified criteria. For example, the following commands list all APAR SYSMODs that were previously installed and subsequently have been superseded:

```
SET      BDY(TGT1)      /* set to target zone       */.
LIST     SYSMOD         /* list SYSMODs             */
         APAR           /* that are APARs           */
         SUP            /* and are superseded       */.
```

On the other hand, these commands list APAR **or** function SYSMODs that were previously installed and subsequently have been superseded:

```
SET      BDY(TGT1)      /* set to target zone       */.
LIST     SYSMOD         /* list SYSMODs             */
         APAR           /* that are APARs           */
         FUNCTION       /*      or functions        */
         SUP            /* and are superseded       */.
```

You can expand the information listed for SYSMOD entries by coding one or more of the following operands:

HOLDDATA
MCS

Note:

1. SYSMODS can also be specified as SYSMOD.
2. If any of the SYSMOD qualifier operands (other than HOLDDATA or MCS) are specified without the SYSMOD operand, SMP/E assumes that you want the SYSMOD entries listed and, therefore, processes as if SYSMOD was also specified.
3. When either MCS or HOLDDATA is specified without a name list on the same LIST command as the SYSMOD operand, SMP/E assumes you want the or HOLDDATA only for those SYSMOD entries that are listed, not all the MCS and HOLDDATA entries. If you want to list **all** the MCS or HOLDDATA entries, use the LIST command with the MCS or HOLDDATA operand, but without the SYSMOD operand and without any of the previously identified SYSMOD qualifier operands.

TARGETZONE
indicates that SMP/E should list the TARGETZONE entry.

Note:

1. TARGETZONE is allowed when the SET command specifies a target zone.
2. TARGETZONE can also be specified as TZONE.
3. The XZLINK(DEFERRED) value is displayed only when the TARGETZONE entry contains TIEDTO records.

USERMODS
indicates that SMP/E should list USERMOD SYSMODs.

Note:

1. USERMODS can also be specified as USERMOD.
2. When USERMODS is used with APARS, FUNCTIONS, or PTFS, SMP/E lists any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD had also been specified, even if it has not.

UTILITY
indicates that SMP/E should list all UTILITY entries or the specified UTILITY entries.

Note: UTILITY is allowed when the SET command specifies the global zone.

XREF
generates cross-reference information appropriate to the entry type being listed. Table 16 on page 221 shows the information included for each entry type:

Table 16. XREF information for each type of entry

Entry type	XREF information
ASSEM entries	A list of all macros whose MAC entry indicates that this module should be reassembled
Element entries	A history of all SYSMODs affecting this element
LMOD entries	A list of all MOD entries that are linked or copied to this load module
SYSMOD entries	• A list of all SYSMODs specifying this SYSMOD on the ++VER DELETE operand • A list of all SYSMODs specifying this SYSMOD on the ++VER NPRE operand • A list of all SYSMODs specifying this SYSMOD on the ++VER PRE operand • A list of all SYSMODs specifying this SYSMOD on the ++VER REQ operand • A list of all SYSMODs specifying this SYSMOD on the ++VER SUP operand • A list of all SYSMODs specifying this SYSMOD on the ++VER VERSION operand • A list of all SYSMODs specifying this SYSMOD on the ++IF REQ operand

Note: SMP/E uses extra time and more storage to generate the additional data requested by the XREF operand.

XZLMODP
indicates that SMP/E should list MOD entries for all modules that have been linked into load modules controlled by a different target zone. (The MOD entries for these modules contain XZLMODP subentries.)

Note:

1. XZLMODP is allowed only when the SET command specifies a target zone.
2. The appropriate MOD entries are listed, regardless of whether the MOD operand was specified on the LIST command.
3. If both MOD and XZLMODP are specified, only MODs with cross-zone subentries are listed. If a list of MODs and XZLMODP is specified, all the specified MODs, as well as all the MODs with cross-zone subentries, are listed.

XZMODP
indicates that SMP/E should list LMOD entries for all load modules containing modules from a different target zone. (The LMOD entries for these load modules contain XZMODP subentries.)

Note:

1. XZMODP is allowed only when the SET command specifies a target zone.
2. The appropriate LMOD entries are listed regardless of whether the LMOD operand was specified on the LIST command.
3. If both LMOD and XZMODP are specified, only LMODs with cross-zone subentries are listed. If a list of LMODs and XZMODP is specified, all the specified LMODs, as well as all the LMODs with cross-zone subentries, are listed.

ZONESET
indicates that SMP/E should list all ZONESET entries or the specified ZONESET entries.

Note: ZONESET is allowed when the SET command specifies the global zone.

For additional information about listing a specific entry type, see the section for that entry in the "SMP/E data set entries" chapter in *z/OS SMP/E Reference*. The description of the data in the entry, as well as examples for using the LIST command, are contained there.

Syntax notes

1. Except where noted, you can specify multiple operands on a single LIST command. In comparison with specifying the operands on separate LIST commands, the overall performance of SMP/E is improved.
2. The order in which the entries are listed depends on their order in the SMP/E data set being used, not on the order specified on the LIST command.
3. You can mix mass-mode and select-mode requests on the same LIST command. For example:

```
SET      BDY(TGT1)      /* Set to target zone.      */.
LIST     MOD            /* List all modules,        */
         MAC(MAC01      /* only two macros,         */
             MAC02)     /*                          */
         SRC(SRC01      /* only two source,         */
             SRC02)     /*                          */
         DLIB           /* all DLIBs,               */
         DDDEF          /* all DDDEFs,              */
         SYSMOD(UZ00001 /* only these five SYSMODs. */
                UZ00002 /*                          */
                UZ00003 /*                          */
                UZ00004 /*                          */
                UZ00005)/*                          */.
```

4. If a given SYSMOD is specified on the SYSMODS operand, the SYSMOD **is** listed, regardless of whether it would be included or excluded by other operands.

Data sets used

These data sets might be needed to run the LIST command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPPTS	SMPSNAP
SMPCSI	SMPLOGA	SMPRPT	*zone*
SMPLIST	SMPOUT	SMPSCDS	

Note:

1. SMPPTS is required only if the MCS operand is specified.
2. SMPSCDS is required only if the BACKUP operand is specified.
3. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, SMP/E dynamically allocates the data sets using the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

1. When the ALLZONES operand is specified, SMP/E displays the GLOBALZONE entry first, followed by all the entries within the global zone. Then, each zone defined by a ZONEINDEX subentry is processed in alphanumeric sequence by zone name. If SMP/E is unable to allocate the VSAM data set containing a particular zone, no further processing is done for that zone. Processing continues with the next zone.
2. Because SMP/E always opens the global zone for all processing, if there is an error during an attempt to open the global zone, you cannot process any SMP/E commands. Therefore, you cannot obtain a list of error messages form the SMPLOG data set.

   If you want to list the SMPLOG, and the CSI is damaged so that it cannot be opened, define a temporary CSI data set and use it to list the SMPLOG.
3. For more information about each entry type, see the "SMP/E Data Set Entries" chapter in *z/OS SMP/E Reference*.

Output

The LIST command output for each entry type is illustrated in the relevant section of the "SMP/E Data Set Entries" chapter in *z/OS SMP/E Reference*.

The following reports are produced during LIST processing:

- File Allocation report
- LIST Summary report

For descriptions of these reports, see Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the LIST command.

For examples of LIST commands and related output for each entry type, see the "SMP/E Data Set Entries" chapter in *z/OS SMP/E Reference*.

Example 1: List all the entries in a particular zone

Suppose you have initialized the global zone with the definition entries needed to process that zone, and you want to list all those entries to check them. To do this, use the LIST command without any operands, as shown:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
LIST                        /* List all entries.        */.
```

Example 2: List all the entries of a particular type

Suppose you want to check all the DDDEF entries defined in target zone TGT1. To do this, use the LIST command with just the DDDEF operand, as shown:

```
SET      BDY(TGT1)          /* Set to target zone       */.
LIST     DDDEF              /* List all DDDEF entries.  */.
```

If you want to check all the DDDEF entries defined in the global zone and all the zones defined to that global zone, add the ALLZONES operand to the previously shown LIST command. (The global zone or any zone defined to it can be specified on the SET command.)

Example 3: List specific entries

Suppose you encounter a problem on your system and need to determine whether SYSMOD UR12345 has been installed in target zone TGT1. To do this, use the LIST command with the SYSMOD operand and the SYSMOD ID, as shown:

```
SET      BDY(TGT1)          /* Set to target zone.      */.
LIST     SYSMOD(UR12345)    /* List this SYSMOD.        */.
```

If you want to check for that SYSMOD in the all the zones defined to the global zone, add the ALLZONES operand to the previously shown LIST command. (The global zone or any zone defined to it can be specified on the SET command.)

Example 4: List entries applicable to specific FMIDs

Suppose you need to determine whether target zone TGT1 contains entries for any elements owned by function SYSMOD HXY1100. To do this, use the LIST command with the FORFMID operand, as shown:

```
SET      BDY(TGT1)          /* Set to target zone.      */.
LIST                        /* List all entries         */
         FORFMID(HXY1100)   /* for this FMID.           */.
```

Example 5: List entries for specific UNIX shell scripts

Suppose you need to determine whether target zone TGT1 contains entries for UNIX shell scripts *script1* or *script2*. To do this, use the LIST command with the SHELLSCR operand, as shown:

```
SET      BDY(TGT1)          /* Set to target zone.      */.
LIST                        /* List all UNIX shell scripts        */
         SHELLSCR(script1,script2)   /* for this zone.          */.
```

Example 6: Check which SYSMODs are received but not installed

Suppose you have received service into the global zone and are in the process of installing the service on your system. You want to see which of the SYSMODs you have received have not yet been installed in target zone TGT1. To do this, use the LIST command with the NOAPPLY operand and the zone name, as shown:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
LIST     SYSMODS            /* List the SYSMODs that    */
                            /* have been received but   */
         NOAPPLY(TGT1)      /* that have not been       */
                            /* applied to TGT1.         */.
```

To see which of the SYSMODs you have received have not yet been installed in distribution zone DLIB1, use the LIST command with the NOACCEPT operand and the zone name, as shown:

```
SET      BDY(GLOBAL)         /* Set to global zone.      */.
LIST     SYSMODS             /* List the SYSMODs that    */
                             /* have been received but   */
         NOACCEPT(DLIB1)     /* that have not been       */
                             /* accepted in DLIB1.       */.
```

Example 7: Check whether SYSMODs are installed in the related zone

Suppose you need to compare the service level of target zone TGT1 with that of its related distribution zone, DLIB1. To do this, use the LIST command with the NOACCEPT operand, as shown below. (Because you are checking the related zone, you do not need to specify the zone name.)

```
SET      BDY(TGT1)           /* Set to TGT1.             */.
LIST     SYSMODS             /* List the SYSMODs that    */
                             /* have been applied but    */
         NOACCEPT            /* that have not been       */
                             /* accepted in the related  */
                             /* zone.                    */.
```

To see whether any SYSMODs have been installed in DLIB1 but not in TGT1, use the LIST command with the NOAPPLY operand, as shown:

```
SET      BDY(DLIB1)          /* Set to DLIB1.            */.
LIST     SYSMODS             /* List the SYSMODs that    */
                             /* have been accepted but   */
         NOAPPLY             /* that have not been       */
                             /* applied to the related   */
                             /* zone.                    */.
```

Note: You can also use the REPORT SYSMODS command to compare zones. Besides telling you which SYSMODs are installed in one zone but not in another, REPORT SYSMODS also indicates which of the uninstalled SYSMODs are applicable to the second zone and generates commands you can run to install the SYSMODs in the second zone. For more information, see Chapter 20, “The REPORT SYSMODS command,” on page 325.

Example 8: Compare the SYSMODs installed in two zones of the same type

Suppose you need to compare the service level of your production system and your test system, and you want to see which of the SYSMODs on the test system are not yet installed on the production system. To compare the target zones of the two systems, use the LIST command with the NOAPPLY operand, as shown:

```
SET      BDY(TGT1)           /* Set to test zone TGT1.   */.
LIST     SYSMODS             /* List the SYSMODs that    */
                             /* have been applied to     */
         NOAPPLY(TGT2)       /* TGT1 but not to          */
                             /* production zone TGT2.    */.
```

To compare the distribution zones of the two systems, use the LIST command with the NOACCEPT operand and the test system zone name, as shown:

```
SET      BDY(DLIB1)          /* Set to test zone DLIB1.  */.
LIST     SYSMODS             /* List the SYSMODs that    */
                             /* have been accepted in    */
         NOACCEPT(DLIB2)     /* DLIB1 but not in         */
                             /* production zone DLIB2.   */.
```

Example 9: Requesting SYSMOD and HOLDDATA

To see a list of the actions associated with system holds, use the LIST command with both the SYSMODS and HOLDDATA operands, as shown:

```
SET      BDY(GLOBAL)    /* set to global zone       */.
LIST     SYSMODS HOLDDATA       /* list only the HOLDDATA   */.
```

The following example output from the LIST SYSMODS HOLDDATA shows the actions

```
LIST SYSMODS HOLDDATA.
GLOBAL SYSMOD ENTRIES
NAME
HBB7730   TYPE               = FUNCTION
          STATUS             = REC
          DATE/TIME REC      = 06.300 12:39:49
          SOURCEID           = SRC2006
          SREL VER(001)      = Z038
          MAC                = MACRO3
          HOLDERROR          = IR12345  ++HOLD(HBB7730) ERROR FMID(HBB7730) REASON(IR12345) CLASS(CLASS01)
                                          DATE(84200) COMMENT(SMRTDATA(CHGDTE(022206) SYMP(IPL))).
          HOLDFIXCAT         = AK18603  ++HOLD(HBB7730) FMID(HBB7730) REASON(AK18603) FIXCAT DATE(06230)
                                          CLASS(PSP)RESOLVER(UK14916)
                                          CATEGORY(IBM.Coexistence.z/OS.V1R8).
          HOLDFIXCAT         = AO23456  ++HOLD(HBB7730) FMID(HBB7730) REASON(AO23456) FIXCAT DATE(06230)
                                          CLASS(PSP)RESOLVER(UO17895)
                                          CATEGORY(IBM.Device.zIIP).
          HOLDSYSTEM(EXT)    = DOC      ++HOLD(HBB7730) SYSTEM FMID(HBB7730) REASON(DOC) DATE(96023)
                                          COMMENT(TEST EXTERNAL HOLD FOR FUNCTION SYSMOD.).
          HOLDSYSTEM(INT)    = DOC      ++HOLD(HBB7730) FMID(HBB7730) SYS REASON(DOC) DATE(96023)
                                          COMMENT(  ---------- COMMENT LINE 1 HBB7730 FOR DOC -------
                                                    ---------- COMMENT LINE 2 HBB7730 FOR DOC -------
                                                    ---------- COMMENT LINE 3 HBB7730 FOR DOC ------- ) .
          HOLDSYSTEM(INT)    = UCLIN    ++HOLD(HBB7730) FMID(HBB7730) SYS REASON(UCLIN) DATE(96023)
                                          COMMENT(    ---------- COMMENT LINE 1 HBB7730 FOR UCLIN -------
                                                    ---------- COMMENT LINE 2 HBB7730 FOR UCLIN -------
                                                    ---------- COMMENT LINE 3 HBB7730 FOR UCLIN ------- ) .
          HOLDUSER           = US56789  ++HOLD(HBB7730) USER FMID(HBB7730) REASON(US56789) CLASS(ABCDEFG)
                                          DATE(84200) COMMENT(TESTING USER HOLD FOR A FUNCTION SYSMOD.).
```

Processing

Before SMP/E lists any entries, it first determines what type of LIST processing has been requested:

- If no entry names or entry types have been specified, SMP/E does mass-mode processing (for example, if `LIST.` or `LIST ALLZONES.` is specified). See “Mass-mode processing” on page 226 for a description of mass-mode processing.

  If entry types without entry names are specified, SMP/E does mass-mode processing (for example, if `LIST MAC MOD.` is specified to list all the MAC and MOD entries in a target zone).
- If entry names are specified (for example, if `LIST MAC(MACA,MACB) MOD(MODA,MODB).` is specified to list specific MAC and MOD entries in a target zone), SMP/E does select-mode processing. See “Select-mode processing” on page 226 for a description of select-mode processing.

Note: A single LIST command may combine mass-mode and select-mode processing.

Mass-mode processing

In mass-mode processing, SMP/E checks whether any entry types have been specified. If so, SMP/E lists all entries of each specified type it finds in the set-to zone. SMP/E reads sequentially through the set-to zone. For each entry it finds, it checks whether the entry is of the specified type and whether the entry meets any other criteria specified (such as SYSMOD, MCS, FORFMID, and HOLDDATA). If the entry meets all the requirements, SMP/E formats and prints the data.

If no entry types are specified, SMP/E lists all the entries in the set-to zone. SMP/E reads sequentially through the set-to zone. For each entry it finds, it formats and prints the data.

Select-mode processing

In select-mode processing, SMP/E lists all the explicitly specified entries that were found in the set-to zone. SMP/E goes directly to each specified entry and locates the data for the entry. For each entry it finds, it formats and prints the data.

Zone and data set sharing considerations

The following identifies the phases of LIST processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: Either the target zone or the distribution zone is used during initialization, according to the zone type specified in the previous SET command.

2. LIST processing

   Global zone
   Read with shared enqueue.

   SMPPTS
   Read with shared enqueue.

   Target zone
   Read with shared enqueue.

   DLIB zone
   Read with shared enqueue.

   Note: The zones used depend on the LIST command operands and the zone type specified in the previous SET command.

3. Termination

   All resources are freed.

Chapter 13. The LOG command

During SMP/E processing, messages recording what has occurred are written to SMPOUT. Critical messages are also written to the SMPLOG data set to provide a permanent record of processing. Still other messages are written to SMPLOG to record internal processing, such as updating a SYSMOD entry or deleting an MTSMAC entry. In addition to the messages written by SMP/E, you may want to store messages in the SMPLOG, such as why a SYSMOD is being installed and who is installing it. You can do this using the LOG command.

Zones for SET BOUNDARY

Each zone should have its own SMPLOG data set, which should be defined by a DDDEF entry in that zone. If you want to use the LOG command to update a particular SMPLOG data set, the SET BOUNDARY command must specify the zone containing the DDDEF entry for that data set.

Syntax

Operands

text
represents the text that is to be written to the SMPLOG.

- LOG text may be in single-byte characters (such as English alphanumeric characters) or double-byte characters (such as Kanji).
- You can enter up to 250 bytes of data on a single LOG command, including blanks. For double-byte data, the 250-byte maximum includes all shift-in and shift-out characters as well as the double-byte characters. If you enter more than 250 bytes of data on one LOG command, the text is truncated. If you need to enter more than the maximum number of characters, use two or more LOG commands.
- SMP/E compresses the text to eliminate consecutive blanks.
- The text may span multiple lines.
- If parentheses are used in the text, they must be matched pairs.
- The format of the text stored in the SMPLOG data set may not be exactly as entered on the LOG command. For more information, see “Processing” on page 231.

Data sets used

The following data sets might be needed to run the LOG command. They can be defined by DD statements or, preferably, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	*zone*
SMPCSI	SMPLOGA	SMPSNAP	

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated by use of the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

The File Allocation report is produced during LOG processing. For a description of this report, see Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the LOG command.

Example 1: Writing a message

You can use the following commands to write a message to three SMPLOG data sets associated with the global zone, target zone MVSTST1, and distribution zone MVSDLB1:

```
SET      BDY(GLOBAL)        /* Process global zone.       */.
LOG      (THIS MESSAGE WILL GO TO THE SMPLOG ASSOCIATED
          WITH THE GLOBAL ZONE).
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.  */.
LOG      (THIS MESSAGE WILL GO TO THE SMPLOG ASSOCIATED
          WITH TARGET ZONE MVSTST1).
SET      BDY(MVSDLB1)       /* Process MVSDLB1 DLIB zone. */.
LOG      (THIS MESSAGE WILL GO TO THE SMPLOG ASSOCIATED
          WITH DISTRIBUTION ZONE MVSDLB1).
```

This example assumes you are having SMP/E dynamically allocate the SMPLOG DD statement; that is, no SMPLOG DD statement was provided in the JCL, and each zone contains a DDDEF entry for the SMPLOG. As SMP/E processes each SET command, SMP/E dynamically frees the SMPLOG data set currently allocated, and then dynamically allocates the SMPLOG DD statement, now pointing to the SMPLOG data set for the appropriate zone. SMP/E then writes the message to that SMPLOG data set.

Example 2: Coding parentheses correctly

The following set of SMP/E commands has an error in the LOG command:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
LOG      (I AM ABOUT TO RECEIVE SERVICE LEVEL
          0701 IN ORDER TO INSTALL A NEW FUNCTION.
          (FUNCTION IS JXX1112).
RECEIVE  SOURCEID(PUT0701)  /* Receive service level.   */.
```

The LOG command does not have matched parentheses. The string `(FUNCTION IS JXX1112)` is in parentheses within the LOG command, but no closing parenthesis is found for the LOG command itself. SMP/E continues scanning the SMPCNTL input, looking for the closing parenthesis. Because it is not found, and because (we assume) there are no more commands, SMP/E issues an error message, and the message is not written to the SMPLOG.

Assume that you had made an additional mistake in the RECEIVE command as follows:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
LOG      (I AM ABOUT TO RECEIVE SERVICE LEVEL
          0701 IN ORDER TO INSTALL A NEW FUNCTION.
          (FUNCTION IS JXX1112).
RECEIVE  SOURCEID(PUT0701)  /* Receive service level.   */.
```

Here, a 9 was mistakenly typed instead of a `(`. SMP/E finds the matching parenthesis after the SOURCEID operand, considers it to be the one for the LOG command, and would consider the text closed. Because no other data is found after the closing parenthesis (other than the period and a valid command comment) SMP/E writes the RECEIVE command to the SMPLOG as part of the LOG command text.

Example 3: Listing an SMPLOG data set

The following is an example of listing the SMPLOG for the global zone:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
LIST     LOG                /* List total log.          */.
```

In addition, parts of the SMPLOG can be listed by specifying a date range, as follows:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
LIST     LOG                /* List part of SMPLOG      */
            (06 01 07,      /* from 06/01/07 through    */
             07 01 07).     /* 07/01/07.                */.
```

For additional information on listing the SMPLOG data set, see Chapter 12, “The LIST command,” on page 205.

Processing

When SMP/E processes the LOG command, the text from the command is placed in an internal buffer, either until the end of text is encountered or until the buffer is full.

As SMP/E is moving the text from the LOG command to the buffer area, it compresses the text by removing consecutive blanks. Thus, the format in which the text is entered in the LOG command (such as spacing or number of lines) is not the same as when you list the LOG (using the LIST LOG command).

After SMP/E has finished scanning the LOG command, the buffer that was built is written to the SMPLOG data set. As each message is written, the current date and time are added to the message text so you can later list the SMPLOG for a specified date range.

Chapter 14. The RECEIVE command

RECEIVE is the first SMP/E command used to process any SYSMOD or HOLDDATA. When a source of SYSMODs and HOLDDATA is specified, the RECEIVE command reads the SYSMODs and HOLDDATA and stages them into the global zone, the SMPPTS, and temporary data sets (SMPTLIBs) for later SMP/E processing. The following sources can be used:

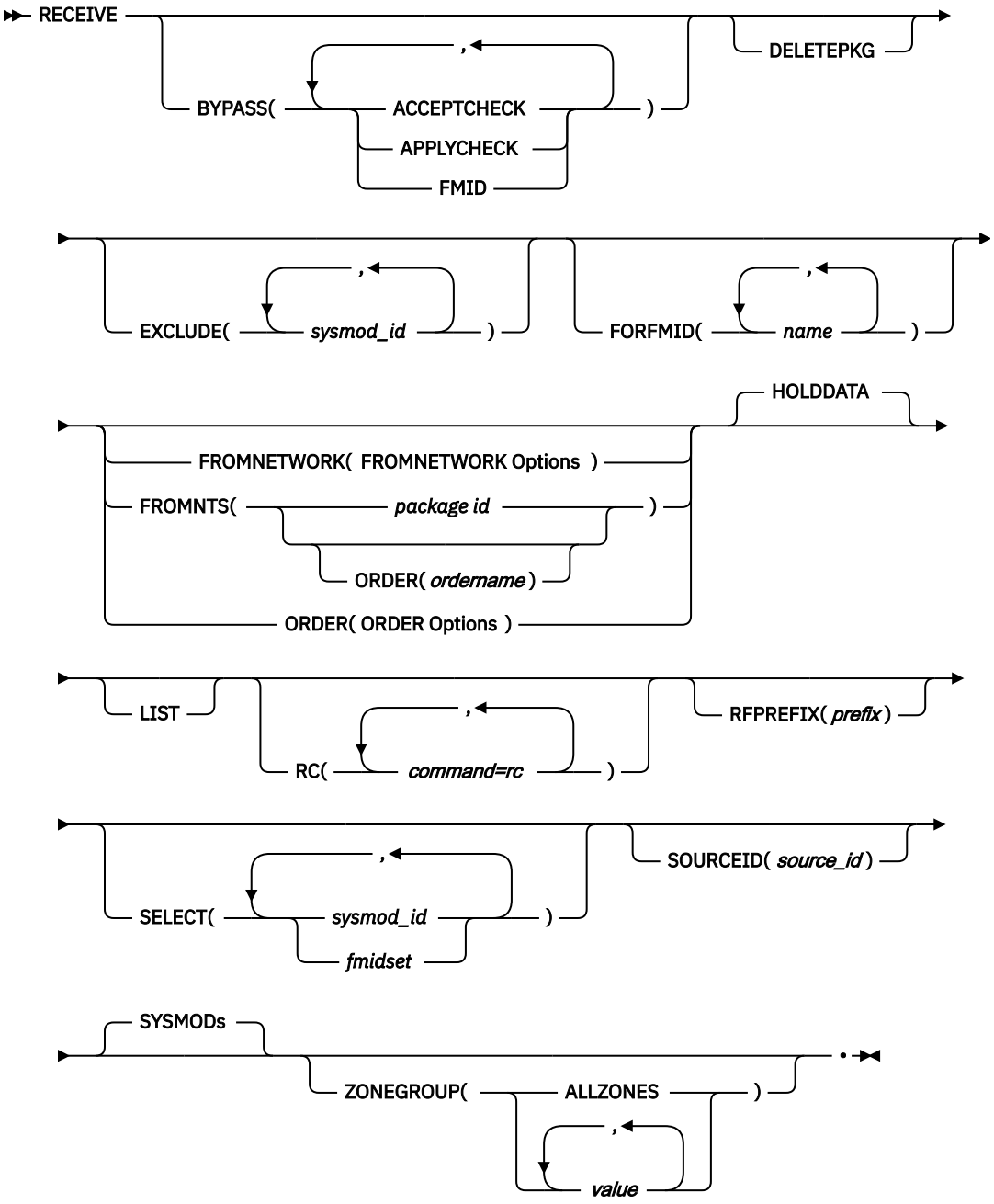
- SMPPTFIN and SMPHOLD, which can be tape files, DASD data sets, or files in a UNIX file system. SMPPTFIN contains the modification control statements defining the SYSMODs, as well as any related ++ASSIGN, ++FEATURE, and ++PRODUCT statements. SMPHOLD contains exception SYSMOD data (++HOLD and ++RELEASE statements).
- a package on an FTP or HTTP(S) server.
- a package in the SMPNTS directory.
- The RECEIVE command can also be used to submit requests for PTF SYSMODs and HOLDDATA to an IBM server when a particular source is not on hand. The packages of SYSMODs and HOLDDATA can be automatically downloaded, read, and staged for later processing.

Zones for SET BOUNDARY

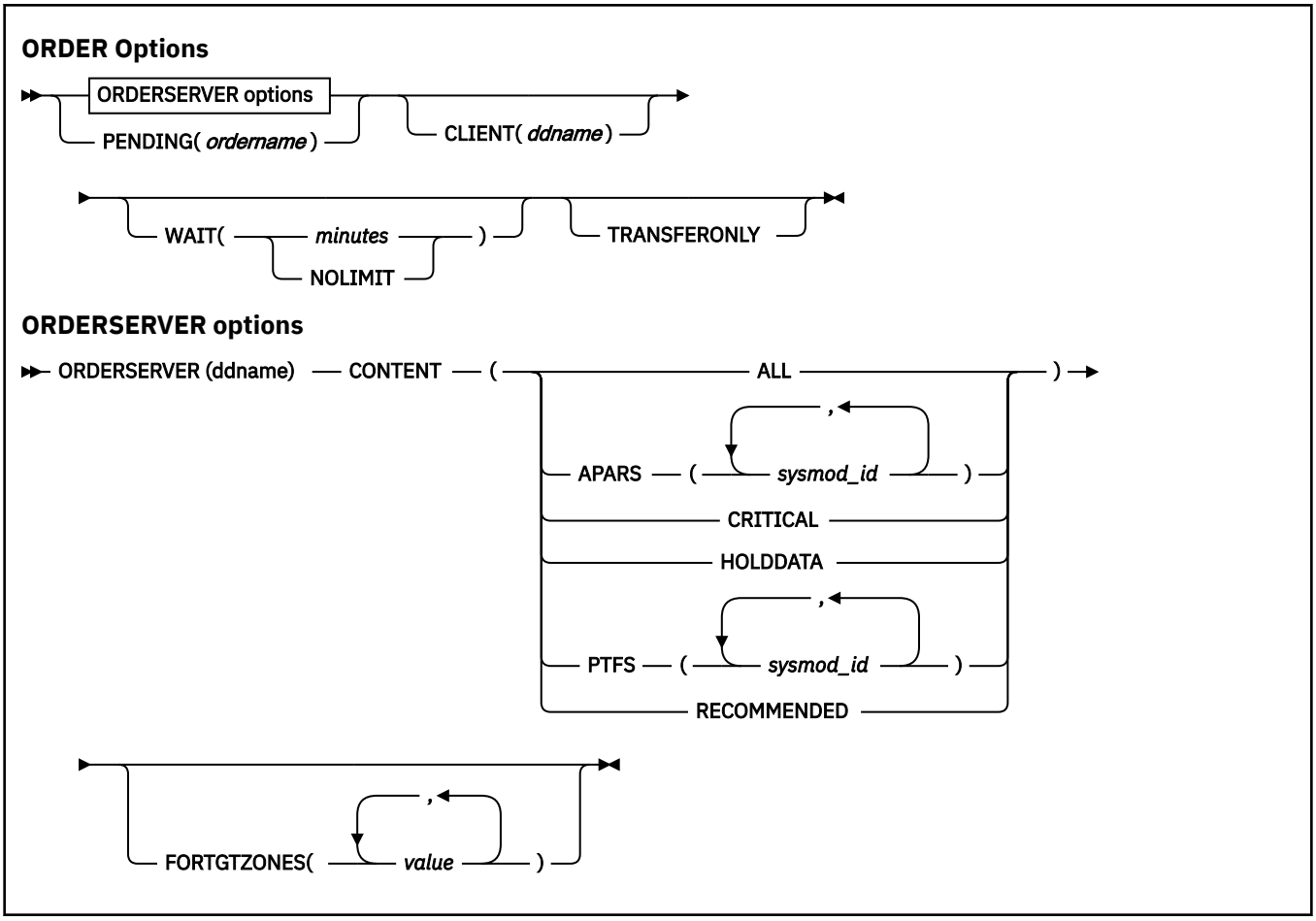
For the RECEIVE command, the SET BOUNDARY command must specify the global zone.

Syntax

RECEIVE Command



FROMNETWORK Options



Operands

BYPASS
You can specify one of these options:

ACCEPTCHECK
APPLYCHECK
FMID

Note:

1. ACCEPTCHECK can also be specified as ACCCHK.
2. APPLYCHECK can also be specified as APPCHK.
3. BYPASS(ACCEPTCHECK APPLYCHECK) is mutually exclusive with ZONEGROUP.
4. BYPASS(FMID) is mutually exclusive with FORFMID.

BYPASS(ACCEPTCHECK)
indicates that selected SYSMODs should be received, even if they have been accepted.

BYPASS(APPLYCHECK)
indicates that selected SYSMODs should be received, even if they have been applied.

BYPASS(FMID)
indicates that all SYSMODs and HOLDDATA should be received, even if the associated FMID is not yet defined in the global zone.

DELETEPKG
specifies that following a successful completion of a RECEIVE FROMNETWORK, FROMNTS, or ORDER the package id subdirectory in the SMPNTS directory that was created or used as input for the command should be deleted.

Note:

1. A "successful RECEIVE" is defined as no warning or error conditions after the package files have been downloaded, up to the point where the subdirectory is to be deleted.
2. This operand is ignored when FROMNETWORK, FROMNTS, or ORDER is not specified on the RECEIVE command.
3. DELETEPKG is mutually exclusive with the TRANSFERONLY option of FROMNETWORK and ORDER.

EXCLUDE
specifies one or more SYSMOD IDs that should not be received.

Note: EXCLUDE can also be specified as E.

FORFMID
indicates that only SYSMODs, FEATUREs, and HOLDDATA for the specified FMIDs or FMIDSETs should be received. Any FMIDSETs specified must already be defined in the global zone.

Note:

1. FORFMID is mutually exclusive with BYPASS(FMID).
2. You cannot use the FORFMID operand as a substitute for UCLIN to add an FMID or FMIDSET to the global zone.
3. You can use FORFMID to receive a given function SYSMOD, along with other SYSMODs that are applicable to that function. For example, if function HXY1100 has not yet been received, you can use FORFMID(HXY1100) to receive that function plus the applicable SYSMODs.

   SMP/E adds the FMID of the function to the global zone when it receives the function. As a result:

   - Any SYSMODs that are applicable to the function and come **before** the function in the SMPPTFIN input stream will **not** be received, because the FMID of the function is not yet in the global zone.
   - Any SYSMODs that are applicable to the function and come **after** the function in the SMPPTFIN input stream **can** be received, because the FMID of the function is now in the global zone.
4. FORFMID does not affect SYSMODs specified on the SELECT operand.

FROMNETWORK
specifies the input for this RECEIVE command is a GIMZIP package on a TCP/IP connected server that supports downloads using FTP or HTTP(S). FROMNETWORK can also be specified as FROMNET.

SERVER(*ddname*)
specifies the 1- to 8-character DD or DDDEF name that points to the data set that provides information about the server that contains input for this RECEIVE command. SERVER is required if FROMNETWORK is specified. See “Defining data sets and files for RECEIVE FROMNETWORK and RECEIVE ORDER processing” on page 246 for a description of the SERVER data set and its contents.

CLIENT(*ddname*)
specifies the 1- to 8-character DD or DDDEF name that points to the data set where RECEIVE can get information about the FTP or HTTP(S) client environment on the local machine, such as how to navigate a local firewall when using FTP. See “Defining data sets and files for RECEIVE FROMNETWORK and RECEIVE ORDER processing” on page 246 for a description of the CLIENT data set and its contents.

TRANSFERONLY
specifies that RECEIVE FROMNETWORK processing should stop after the network package has been transferred into the SMPNTS file structure of a UNIX file system.

Note:

1. TRANSFERONLY is mutually exclusive with DELETEPKG.
2. When TRANSFERONLY is specified, the BYPASS, EXCLUDE, FORFMID, HOLDDATA, LIST, SELECT, SOURCEID, SYSMODs, and ZONEGROUP operands do not apply and are not processed if specified.

3. When TRANSFERONLY is not specified, RECEIVE FROMNETWORK processing performs the more typical function of RECEIVE (in addition to transferring the network package) by processing SYSMODs and HOLDDATA from the just transferred network package into the global zone, the SMPPTS, and temporary data sets (SMPTLIBs) for later SMP/E processing.

Note:

1. FROMNETWORK is mutually exclusive with FROMNTS and ORDER.
2. When FROMNETWORK is specified:
   - a DD or DDDEF for SMPNTS is required.
   - DDs or DDDEFs for SMPPTFIN and SMPHOLD are not required and are ignored if supplied.
   - RFPREFIX is ignored, if specified.
   - RECEIVE FROMNETWORK processing requires either the Integrated Cryptographic Services Facility (ICSF) or JAVA 2 Version 1 Release 4 in order to compute a SHA-1 hash value.
3. The FTP or HTTP(S) conversation generated by the RECEIVE FROMNETWORK command is recorded in the print data set for the HFSCOPY utility (the default is SYSPRINT). This output may be useful for diagnosing problems with RECEIVE FROMNETWORK.

FROMNTS
specifies that the input for this RECEIVE command is a package in the SMPNTS directory. The package is identified by either:

- a package-id value which is the subdirectory for the package in the SMPNTS, or
- an ORDER entry name. The subdirectory for the package is the value of the PKGID subentry in the ORDER entry specified on the FROMNTS operand.

The RECEIVE FROMNTS command looks for input in the following subdirectories of the package identifier directory within the SMPNTS directory:

Subdirectory
Contents

SMPPTFIN
Modification control statements that define SYSMODs and ++ASSIGN, ++FEATURE and ++PRODUCT statements.

SMPHOLD
Exception SYSMOD data (++HOLD and ++RELEASE statements)

SMPRELF
RELFILE data sets

Valid characters for the package id value include X'41' through X'FE' excluding '/' (X'61'), '<' (X'4C'), '>' (X'6E'), '"' (X'7F') and '&' (X'50'). However, if the package id value contains any characters other than uppercase alphabetic (A-Z), numeric (0-9), or national (@,#,$), it must be enclosed in single apostrophes. Any apostrophes specified as part of the package id value itself, not the delimiting apostrophes, must be doubled. Double apostrophes count as two characters in the 50 character limit. The single apostrophes used to delimit the package id value do not count as part of the 50 character limit.

Note:

1. FROMNTS is mutually exclusive with FROMNETWORK and ORDER.
2. When FROMNTS is specified:
   - a DD or DDDEF for SMPNTS is required.
   - DDs or DDDEFs for SMPPTFIN and SMPHOLD are not required and are ignored if supplied.
   - RFPREFIX is ignored, if specified.

ORDER
indicates that SMP/E order HOLDDATA or PTF SYSMODs directly from an IBM server and download the resulting package when the IBM server fulfills the order. HOLDDATA packages contain the last 2 years

of Enhanced HOLDDATA for the entire z/OS platform. PTF packages contain the PTFs ordered based on the specified content criteria and are tailored to the specific SMP/E environment. In addition, the last 2–years of Enhanced HOLDDATA is included in the PTF packages.

ORDERSERVER
specifies the 1- to 8-character ddname that points to the data set where RECEIVE can get the necessary information about the IBM order server that fulfills the order requests from SMP/E. See "Defining data sets and files for RECEIVE FROMNETWORK and RECEIVE ORDER processing" on page 246 for a description of the ORDERSERVER data set and its contents.

CLIENT
specifies the 1- to 8-character ddname that points to the data set where RECEIVE can get the necessary information about the local z/OS client, such as information about navigating an FTP firewall and an HTTP proxy server. See "Defining data sets and files for RECEIVE FROMNETWORK and RECEIVE ORDER processing" on page 246 for a description of the CLIENT data set and its contents.

CONTENT
indicates the desired PTF and/or HOLDDATA content for the order. You can specify one of the following options:

ALL
include all available PTFs that are applicable to the SMP/E environment in the resulting PTF package.

APARS
include the resolving PTF SYSMODs for one or more specified APARS. The resulting PTF package is tailored to the SMP/E environment and will contain the requested PTFs, plus any requisite PTFs that are not already present in the SMP/E environment.

The specified values must be 7-alphanumeric character SYSMOD-ids.

CRITICAL
include all available PTFs that resolve a critical problem and are applicable to the SMP/E environment in the resulting PTF package. A critical problem is a HIPER (high impact pervasive) or a PE (PTF in error).

HOLDDATA
include only HOLDDATA in the resulting package. The package will contain the last 2-years of Enhanced HOLDDATA for the entire z/OS platform. For more information, go to Enhanced HOLDDATA for z/OS (service.software.ibm.com/holddata/390holddata.html).

PTFS
include one or more specified PTF SYSMODs in the resulting package. The resulting PTF package is tailored to the SMP/E environment and will contain the requested PTFs, plus any requisite PTFs that are not already present in the SMP/E environment.

The specified values must be 7-alphanumeric character SYSMOD-ids.

RECOMMENDED
include all available recommended PTFs that are applicable to the SMP/E environment in the resulting PTF package. A recommended PTF is identified with a Recommended Service Update SOURCEID (RSU*yymm*) or resolves a critical problem (HIPER or PE). The resulting PTF package includes PTFs through and including the most current RSU level.

FORTGTZONES
identifies the specific target zones SMP/E should use to create the software inventory on which the resulting PTF package will be tailored. The software inventory is used to determine:

- which PTFs are applicable to the SMP/E environment, and
- which PTFs are already present in the SMP/E environment and, therefore, do not need to be included in the resulting package.

Note: PTFs requested on the CONTENT PTFS operand are included in the resulting package whether they are already present in the SMP/E environment or not.

Using the FORTGTZONES operand, you can specify target zone names, ZONESET names, or both. If FORTGTZONES is not specified, SMP/E uses all target zones (defined by a ZONEINDEX subentry in the current global zone), plus the global zone to create the software inventory.

The specified values must be 1- to 8-alphanumeric characters.

PENDING
specifies the name of an ORDER entry whose package has not yet been downloaded. The PENDING operand indicates that SMP/E should attempt to download the package for the specified order. Such an order request is described by an ORDER entry in the global zone that has a status of PENDING.

WAIT(*minutes* | NOLIMIT)
where *minutes* is a decimal number between 0 and 1440. The WAIT operand indicates how long in minutes SMP/E is to wait until an order is ready for download. This includes the time associated with connecting to the order server and preparing the order package for download. If the order is not ready within the specified time limit, then RECEIVE command processing stops.

If WAIT(NOLIMIT) is specified, SMP/E waits for an unlimited time until the order is ready for downloading. If WAIT(0) is specified for a new ORDER, SMP/E sends the order request to the IBM server and completes RECEIVE command processing without waiting and without downloading the resulting package. If WAIT(0) is specified for a pending order, SMP/E processes the order only if it is ready for downloading immediately.

This operand is optional. If the WAIT operand is not specified, SMP/E waits 120 minutes by default for orders to be ready for downloading.

TRANSFERONLY
indicates that RECEIVE ORDER processing should stop after the package has been downloaded into the SMPNTS directory. SMP/E does not read the PTFs and HOLDDATA from the package and does not store it in either the global zone or the SMPPTS data set.

To receive the PTFs and HOLDDATA from a TRANSFERONLY package, use the RECEIVE FROMNTS command subsequently to process the package stored in the SMPNTS directory. Specify the ORDER entry name with the ORDER suboperand on the FROMNTS operand.

HOLDDATA
indicates that applicable data from SMPHOLD should be received.

- If you specify SELECT or FORFMID, HOLDDATA is received only for the SYSMODs included by those operands. For details, see "Example 4: Receiving selected SYSMODs and HOLDDATA" on page 262.
- If you specify neither SELECT nor FORFMID, HOLDDATA is received for all FMIDs defined in the global zone.

LIST
indicates that SMP/E should list the modification control statements for each applicable SYSMOD.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the RECEIVE command.

Before SMP/E processes the RECEIVE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the RECEIVE command. Otherwise, the RECEIVE command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the RECEIVE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RFPREFIX
specifies the data set prefix used to construct the full data set name for RELFILE data sets when a SYSMOD packaged in RELFILE format is being processed by the RECEIVE command. This operand should be used when the name of the RELFILE data set starts with a prefix not identified by the RFDSNPFX operand on the header MCS of the associated SYSMOD. (For example, you may need to use RFPREFIX if you have loaded RELFILEs into DASD data sets and have specified your own high-level qualifier for those data sets.)

Unless SMP/E is informed otherwise, it assumes that the name of a RELFILE data set is *sysmod_id*.F*n*, where *sysmod_id* is the ID of the associated SYSMOD and *n* is the file number of the relative file.

Note: The RFPREFIX can contain from 1 to 26 alphanumeric (uppercase A–Z, 0–9), national ($, #, @), dash (-), or period (.) characters.

After constructing the RELFILE data set name, SMP/E calculates the length of the full data set name (including both the RFPREFIX and RFDSNPFX values). If the length of the full data set name exceeds 44 characters, processing stops for the RECEIVE command.

Note:

1. The RFPREFIX operand is ignored unless the FILES operand is specified on the header MCS statement for the SYSMOD being processed (++APAR, ++FUNCTION, ++PTF, or ++USERMOD MCS).
2. If the RFDSNPFX operand is specified on the header MCS, the RFPREFIX value specified on the RECEIVE command precedes the value from the RFDSNPFX operand when the name of the RELFILE data set is constructed.
3. If the RELFILE data sets are on DASD or are on tape and cataloged, the RELFILE data set name must not match the name to be used for the SMPTLIB data sets. If these data sets have the same name, the SMPTLIB data sets cannot be allocated. The DSPREFIX value in the OPTIONS entry is used to specify the high-level qualifier for the SMPTLIB data set names. For a description of the OPTIONS entry, see *z/OS SMP/E Reference*.

SELECT
specifies one or more SYSMOD IDs that should be received. Any FEATUREs associated with these SYSMODs may also be received.

You may specify any combination of individual SYSMOD IDs and FMIDSET names, provided that there are no duplicate SYSMOD IDs nor any duplicate FMIDSET names. For each FMIDSET specified, all FMIDs defined in the FMIDSET are processed as if they were explicitly specified in the SELECT list.

Each SYSMOD must be in the SMPPTFIN data set, and the associated FMID must be either defined in the global zone or received concurrently.

Note:

1. SELECT can also be specified as S.
2. SMP/E does not do APPLYCHECK and ACCEPTCHECK processing for SYSMODs explicitly specified in the SELECT list.
3. If both SYSMODs and HOLDDATA are being processed and a selected SYSMOD is not received, the associated HOLDDATA **can** be received.
4. If you specify HOLDDATA and do not specify SYSMODs, only the HOLDDATA for the selected SYSMODs is received. The SYSMODs themselves are not received, nor are any associated + +FEATURE or ++PRODUCT MCS.
5. When using FMIDSETs on the SELECT operand, remember that:
   - A value specified in the SELECT list is processed as an FMIDSET if the GLOBAL zone contains an FMIDSET entry by that name.
   - A value specified in the SELECT list is processed as a SYSMOD ID if it is not defined as an FMIDSET in the GLOBAL zone and it is a valid SYSMOD ID.
   - If the value in the SELECT list is valid both as a SYSMOD ID and as an FMIDSET name, it is processed (for SELECT) as an FMIDSET. If you want to select a SYSMOD that has the same name

as an FMIDSET, you must define that SYSMOD in an FMIDSET and then include that FMIDSET name in the SELECT list.

If this same value is specified on the EXCLUDE operand, it will be processed as a SYSMOD ID (because only SYSMOD IDs are valid on EXCLUDE) and will **not** be rejected as a duplication of the identical FMIDSET name in the SELECT list.

- Any given value (whether it represents a SYSMOD ID, an FMIDSET, or both) may **not** appear more than once in the SELECT list.
- Any given SYSMOD ID may not simultaneously appear in both the SELECT and EXCLUDE lists, unless it is also a valid FMIDSET name.
- A SYSMOD ID may be explicitly specified in the SELECT list and also included in an FMIDSET that is also specified in the SELECT list, provided the SYSMOD ID does not have the same name as the FMIDSET. The duplicate SYSMOD ID is ignored.

SOURCEID
specifies a 1- to 64-character source identifier to be assigned to the SYSMODs being received. The SOURCEID value can consist of any nonblank character (X'41' through X'FE') except single quotation marks, asterisks, percent character, comma, left parenthesis and right parenthesis. SMP/E assigns this the source ID to all the SYSMODs that are processed by this RECEIVE command. SMP/E also assigns the source ID to any SYSMODs that would have been received, but were not received because they had already been received.

Note:

1. SMP/E does not check whether the source ID is unique. If the same value is specified on multiple RECEIVE commands, all SYSMODs processed by both commands have the same source ID.
2. If a source ID was specified for a particular SYSMOD on an ++ASSIGN statement in the input stream, that inline source ID is added to the source ID assigned to the SYSMOD by the RECEIVE command.
3. A source ID value is case-sensitive and is stored in the CSI in the form in which it was entered on the SOURCEID operand.

SYSMODs
indicates that data from SMPPTFIN should be received.

Note: SYSMODs can also be specified as SYSMOD.

ZONEGROUP
identifies a list of zones to be interrogated during APPLYCHECK and ACCEPTCHECK processing for this RECEIVE command. This list can include zone names, ZONESET names, or both. Each value in the list must be 1 to 8 alphanumeric or national (@, #, and $) characters. You can specify target zones, distribution zones, or any combination of the two.

Values specified on the ZONEGROUP operand override any values specified in the RECZGRP and RECEXZGRP subentries of the OPTIONS entry.

Specifying ZONEGROUP(ALLZONES) indicates that all zones defined by a ZONEINDEX subentry in the GLOBALZONE entry are eligible. When ALLZONES is specified, any other values specified on ZONEGROUP are ignored.

Note that the ZONEGROUP operand (and the RECZGRP and RECEXZGRP subentries) have no effect on SYSMODs specified on the SELECT list, because SMP/E does not do APPLYCHECK and ACCEPTCHECK processing for SYSMODs explicitly specified in the SELECT list.

Syntax notes

1. The GLOBALZONE entry must contain at least one SREL value in order to receive either SYSMODs or exception SYSMOD data. If no SREL is defined, RECEIVE processing fails. If this happens, you can use UCLIN to add an SREL to the global zone, and then rerun the RECEIVE job.
2. When you specify SELECT with FORFMID, you can also specify EXCLUDE to exclude specific SYSMODs or HOLDDATA for the specified FMIDs.

When you specify SELECT without FORFMID, however, there is no need to specify EXCLUDE.

3. If you specify neither SELECT, EXCLUDE, nor FORFMID, SMP/E considers all the data in SMPPTFIN and SMPHOLD eligible for processing.
4. SMP/E receives data from both SMPPTFIN and SMPHOLD if you specify either of the following situations:
   - Both SYSMODs and HOLDDATA
   - Neither SYSMODs nor HOLDDATA

   If you specify only SYSMODs, HOLDDATA is excluded. If you specify only HOLDDATA, SYSMODs are excluded.
5. When requesting a new order with the RECEIVE ORDER command, the ORDERSERVER data set is required. When processing a PENDING order with the RECEIVE ORDER command, the ORDERSERVER data set is not required.
6. If CONTENT(HOLDDATA) is specified when an order is requested with the RECEIVE ORDER command, the resulting package will contain only HOLDDATA and no PTFs. Therefore, the HOLDDATA selection operand will be assumed when the resulting package is received.

Data sets used

The following data sets might be needed to run the RECEIVE command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

client	SMPHRPT	SMPPTS	SYSUT2
server	SMPJHOME	SMPRPT	SYSUT3
orderserver	SMPLOG	SMPSNAP	SYSUT4
SMPCNTL	SMPLOGA	SMPTLIB	*ZONE*
SMPCPATH	SMPNTS	SMPWKDIR	
SMPCSI	SMPOUT	SYSPRINT	
SMPHOLD	SMPPTFIN	SYSUT1	

Note:

1. SMPHRPT is an optional data set. If SMPHRPT is defined, the HOLD reports are directed there.
2. For RECEIVE processing, the SMPCSI DD statement refers to the data set containing the global zone, which is where SMP/E stores information about those SYSMODs received.
3. SMPNTS and SYSUT4 are required only when FROMNETWORK, FROMNTS, or ORDER are specified.
4. The SMPHOLD DD statement is required only if exception SYSMOD data is to be received from SMPHOLD.
5. The SMPPTFIN DD statement is required only if SYSMODs or ++ASSIGN statements are to be received from SMPPTFIN.
6. *client* represents the DD statement specified on the CLIENT keyword on either RECEIVE FROMNETWORK or RECEIVE ORDER command. It is required only when CLIENT is specified.
7. *server* represents the DD statement specified on the SERVER keyword on a RECEIVE FROMNETWORK command. It is required only when SERVER is specified.
8. *orderserver* represents the DD statement specified on the ORDERSERVER keyword on a RECEIVE ORDER command. It is required only when ORDERSERVER is specified.
9. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, SMP/E dynamically allocates the data sets using the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

10. When TRANSFERONLY is specified, SMPHOLD, SMPPTFIN, SMPPTS, SMPTLIB, SYSUT1, SYSUT2, SYSUT3, SYSUT4, and *zone* are not used.
11. SMPWKDIR is an optional DD statement used to specify a directory in a UNIX file system for the storage of temporary files created during SMP/E processing. If an SMPWKDIR DD statement or DDDEF entry is not specified, SMP/E will use both the system /tmp directory and the package directory in the SMPNTS directory for temporary work files.
12. SMPJHOME is an optional DD statement used to specify the path used to locate the Java runtime. SMP/E requires the Java runtime for RECEIVE ORDER processing, and to calculate SHA-1 hash values for RECEIVE FROMNETWORK and RECEIVE FROMNTS when ICSF is not active. If the Java runtime is not specified using an SMPJHOME DD statement or DDDEF entry, then the directory can be specified on the *javahome* attribute of the client data set for the RECEIVE ORDER and RECEIVE FROMNETWORK commands.
13. SMPCPATH is an optional DD statement used to specify the search path for the SMP/E Java application classes required for RECEIVE ORDER processing. When ICSF is not active, the application classes are also required to calculate SHA-1 hash values for RECEIVE FROMNETWORK and RECEIVE FROMNTS. If the classpath is not specified using an SMPCPATH DD statement or DDDEF entry, then the directory may be specified on the *classpath* attribute of the CLIENT data set for the RECEIVE ORDER and RECEIVE FROMNETWORK commands. If classpath is not specified either in the client data set or by using an SMPCPATH DD statement or DDDEF entry, then a default classpath value of `"/usr/lpp/smp/classes/"` is used.

Usage notes

This section provides usage notes for the RECEIVE command.

Receiving SYSMODs packaged in relative files

SMP/E can receive SYSMODs packaged in relative file format. These relative files can be on either DASD or tape. When the files are on DASD, the data sets must be cataloged and can be either partitioned data sets or sequential data sets (as produced by the copy utility when unloading partitioned data sets).

If a SYSMOD being received is packaged in relative files, those files are loaded into temporary direct access data sets as part of RECEIVE processing. (For more information about packaging SYSMODs in relative files, see Standard Packaging Rules for z/OS-based Products (publibz.boulder.ibm.com/epubs/pdf/gimpkg80.pdf).) These temporary data sets, called SMPTLIBs, can be allocated before you receive the SYSMOD, or they can be dynamically allocated during RECEIVE processing. (For details on defining SMPTLIB data sets, see *z/OS SMP/E Reference*.)

If you want to allocate the SMPTLIBs yourself, you must let SMP/E know where to find the data sets. There are several ways you can do this:

- **Catalog:** You can catalog the SMPTLIB data sets, as well as allocate them. SMP/E allocates the data sets through the catalog when trying to find the data sets.
- **DDDEF entry:** Create a DDDEF entry named SMPTLIB in the global zone, and specify the following information:
  - `VOLUME`, to indicate the DASD volume or volumes on which the data sets reside.
  - Do not specify an initial or final disposition. SMP/E determines the appropriate values for RECEIVE processing.
- **DD statement:** Include an SMPTLIB DD statement in the RECEIVE JCL, and specify the following information:
  - `VOL`, to indicate the DASD volume or volumes on which the data sets reside.
  - `UNIT`, to indicate the unit on which the data sets reside.
  - Do not specify an initial or final disposition. SMP/E determines the appropriate values for RECEIVE processing.

Note: If the unit for the SMPTLIB data sets is not defined in SYSALLDA, you must use a DDDEF entry instead of a DD statement to define the data sets. Otherwise, the data sets are not allocated.

Note:

1. If an error occurs when SMP/E is unloading the relative files or allocating SMPTLIBs during RECEIVE processing, preallocated SMPTLIBs **do** get deleted.
2. If the SMPTLIBs are not cataloged, SMP/E automatically catalogs them, and uncatalogs the data sets when it deletes them.

If you want SMP/E to dynamically allocate the SMPTLIBs, you can specify the information it needs on the SMPTLIB DD statement, in the SMPTLIB DDDEF entry, in the OPTIONS entry used for RECEIVE processing, or in some combination of these. Table 17 on page 244 shows what information SMP/E needs, and where it can be specified. (SMP/E checks the sources of information in the order shown.) Do not specify an initial or final disposition. SMP/E determines the appropriate values for RECEIVE processing.

Note:

1. For recommendations on the space required for a particular product's SMPTLIB data sets, see the installation documentation for that product. Either allocate the data sets yourself with this space, or specify it as indicated in Table 17 on page 244 if the SMPTLIB data sets are being dynamically allocated.
2. If you are not using SMS to manage your storage, then to properly allocate the SMPTLIB data sets, you need to provide allocation information from the sources listed in Table 17 on page 244.

   If you are using SMS to manage your storage, then depending on how SMS is implemented on your system, you may not need to provide allocation information from the sources listed in Table 17 on page 244. If information required by SVC 99 (the dynamic allocation routine) is not defined to either SMP/E or SMS (such as information about volumes or space allocation), allocation fails for the SMPTLIB data sets.

   Because the necessary information can be provided outside of SMP/E (through SMS), SMP/E does not issue error messages if any of this information is not specified.

Table 17. Information used to dynamically allocate SMPTLIBs

Information	How to specify on the DD statement	How to specify in the DDDEF entry	How to specify in the OPTIONS entry	Default
Volume	VOL From 1 to 5 volumes	VOLUME From 1 to 5 volumes	Not specified here	No default
Unit	UNIT (must be part of SYSALLDA)	UNIT	Not specified here	SYSALLDA is used as the unit type.
Space allocation	Not specified here	SPACE	DSSPACE	No default, unless SMPPARM member GIMDDALC contains a GIMDDALC control statement. See *z/OS SMP/E Reference* for details.
Data set prefix	Not specified here	DSPREFIX This is used in the data set name: *prefix.sysmod_id*.F*nnnn*	DSPREFIX This is used in the data set name: *prefix.sysmod_id*.F*nnnn*	No prefix is used in the data set name: *sysmod_id*.F*nnnn*

SMS considerations when using PDSEs with SMPTLIB data sets

If you do not use SMS...

If you are not using Storage Management Subsystem (SMS) to manage your storage, you can skip this section.

If you let SMP/E dynamically allocate SMPTLIB data sets (instead of preallocating them yourself or predefining the DSNTYPE value to be used), SMP/E determines whether to allocate the SMPTLIB data set with a DSNTYPE value of PDS or LIBRARY, based on the format of the corresponding RELFILE data set. If SMP/E cannot determine the DSNTYPE of the RELFILE data set, SMP/E allocates the SMPTLIB data set using the DSNTYPE value specified in the SMPTLIB DDDEF. In either case, SMP/E specifies the DSNTYPE value to use for the SMPTLIB dynamic allocation and, as a result, any DSNTYPE value from the IGDSMS*xx* default is not used.

If SMP/E cannot determine what DSNTYPE value to use by either of the previously described methods, SMP/E does not pass a DSNTYPE value to dynamic allocation. The DSNTYPE is, in this case, determined by the system default or by the SMS subsystem.

Note: You should disable any existing ACS filter routines that force the allocation of an SMPTLIB data set with a particular DSNTYPE, because such routines will interfere with SMP/E's ability to specify the correct DSNTYPE itself.

Receiving SYSMODs created by the BUILDMCS command

The RECEIVE command can process SYSMODs created by the BUILDMCS command (see Chapter 4, “The BUILDMCS command,” on page 107). The RECEIVE command processing for such SYSMODs is similar to that for relative files. The RECEIVE command uses the FROMDS operands in the MCS created by the BUILDMCS command to determine which data sets to use as input for SMPTLIB creation. The BUILDMCS command includes the FROMDS operand in the following MCS:

- Data elements
- Hierarchical file system elements
- JCLIN
- MAC
- MOD
- SRC

See the description of these MCS in *z/OS SMP/E Reference* for more information on the FROMDS operand and its suboperands.

Restarting RECEIVE FROMNETWORK

If errors occur during the transfer of files for a package, it may be necessary to restart the RECEIVE FROMNETWORK operation in order to complete the transfer of the entire package. You can simply rerun the RECEIVE FROMNETWORK command, and SMP/E will determine which files have already been transferred successfully, and which files need to be transferred again or have not yet been transferred. Before transferring a file, RECEIVE processing checks to see if the file already exists in the package directory of the SMPNTS. If it does, the hash value for the existing file is calculated and compared to the hash value supplied in the package attribute file. If they match, the file is used as it exists in the package directory and is not transferred again. If the hash value of the existing file does not match the value supplied in the package attribute file, the file is transferred from an FTP or HTTP(s) server and replaces the existing file in the package directory.

Defining an installation-wide exit routine for RECEIVE processing

You can define an installation-wide exit routine that examines each MCS (including comments and text) as it is read in from the SMPPTFIN data set. The exit routine is activated after the first record is read from SMPPTFIN, and is deactivated when RECEIVE processing stops. The RECEIVE exit routine can do any of these tasks:

- Change the input record
- Skip the input record
- Insert a record
- Stop RECEIVE processing for the current SYSMOD
- Stop RECEIVE processing
- Stop SMP/E processing

For more information about coding this exit routine, see *z/OS SMP/E Reference*.

Defining data sets and files for RECEIVE FROMNETWORK and RECEIVE ORDER processing

One or more data sets and files may be required for RECEIVE FROMNETWORK command processing:

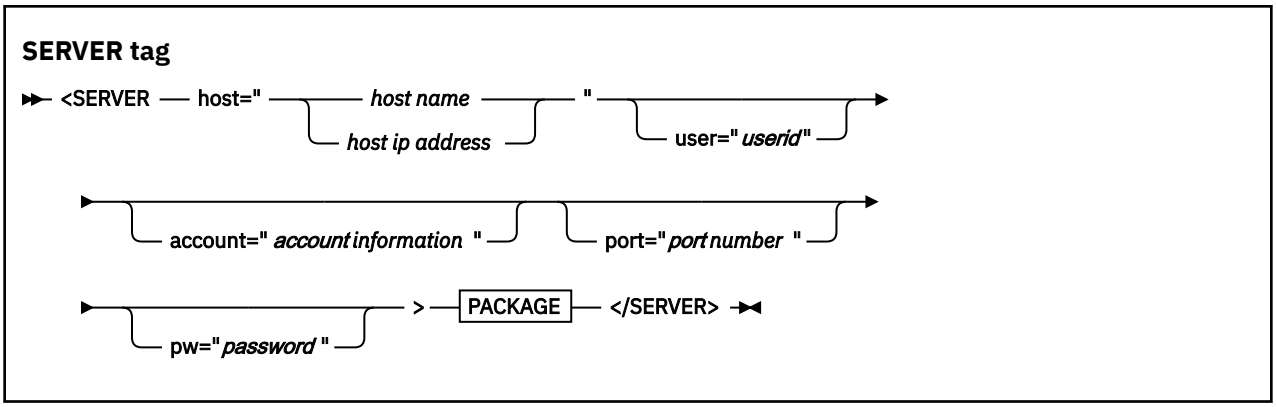
- A SERVER data set must be defined to provide the necessary information about the FTP or HTTP(S) server from which a network package is to be received and about the network package itself.
- A CLIENT data set may also be required to provide information about the local host, such as information about the local firewall requirements.
- A FTP.DATA file may be required to provide information to the z/OS FTP client to permit SMP/E to do secure transfers and to navigate SOCKS firewalls.

One or more data sets and files may be required for RECEIVE ORDER command processing:

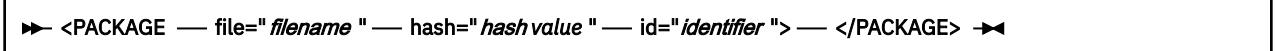
- An ORDERSERVER data set must be defined to provide the necessary information about the IBM order server which will fulfill an SMP/E HOLDDATA or PTF order request.
- A CLIENT data set may also be required to provide information about the local host, such as information about navigating an FTP firewall and an HTTP proxy server.
- A FTP.DATA file may be required to provide information to the z/OS FTP client to permit SMP/E to do secure transfers and to navigate SOCKS firewalls.

Content of SERVER data set

The SERVER data set contains information about a TCP/IP connected host running an FTP or HTTP(S) server. See "Syntax rules for XML statements" on page 3 for a description of the general syntax rules that apply to this data set.



PACKAGE tag



<SERVER>
specifies the start of SERVER data. This tag is required.

The attributes for the <SERVER> tag are:

host
identifies the FTP or HTTP(S) server from which a package is to be received. The `host` attribute must specify either:

host name
a fully qualified internet host name that can be resolved by the domain name system to an internet address. A host name is a text string up to 255 characters drawn from the alphabet (A-Z), digits (0-9), period (.), and minus sign (-). Periods are allowed only as delimiters within domain style names. No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be a letter or a digit. The last character must not be a minus sign or period.

host ip address
an internet address defining the host's location on the internet. The internet address can be an IPv4 or IPv6 address. An IP address can be up to 255 nonblank characters, excluding the reserved XML characters, < (X'4C'), > (X'6E'), double quotation mark (X'7F'), and ampersand (X'50').

One `host` attribute specifying either a host name or a host IP address is required. The `host` attribute value must be enclosed in quotation marks

user
specifies a user ID that will give you access to the host machine. This attribute is a text string up to 80 characters of any type and must be enclosed in quotation marks. If anonymous logins are accepted by this host, specify `user="anonymous"`. Although the `user` attribute is optional for SMP/E, a user ID might be required by the host server.

account
specifies the account information for the specified user ID. This attribute is a text string up to 80 characters of any type and must be enclosed in quotation marks. The `account` attribute is optional.

port
specifies the port number to be used for TCP/IP operations on the specified host. The port number must be a decimal number in the range 1 through 65535. This attribute value must be enclosed in quotation marks. The `port` attribute is optional.

pw
specifies a password value that will give you access to this host. This attribute is a text string up to 80 characters of any type and must be enclosed in quotation marks. Although the pw attribute is optional for SMP/E, a password might be required by this host.

Note: Use system facilities to restrict access to the data set named on the SERVER operand to ensure the security of the password value.

</SERVER>
specifies the end of SERVER data. This tag is required.

<PACKAGE>
specifies the start of PACKAGE data. All data about the package to be retrieved from this SERVER follows this tag and precedes the </PACKAGE> tag. This tag is required. Only one PACKAGE tag may be specified within the SERVER data.

The attributes for the <PACKAGE> tag are:

file
specifies the full name and path of a package attribute file. This file contains a list of all the files that are to be transmitted by this RECEIVE command, as well as information about those files. The path can contain 1 to 266 characters of type X'40' through X'FE'. The `file` attribute value is required and must be enclosed in quotation marks.

hash
specifies the 40 character hexadecimal representation of the 20 byte SHA-1 hash value of the package attribute file. The `hash` attribute value is required and must be enclosed in quotation marks.

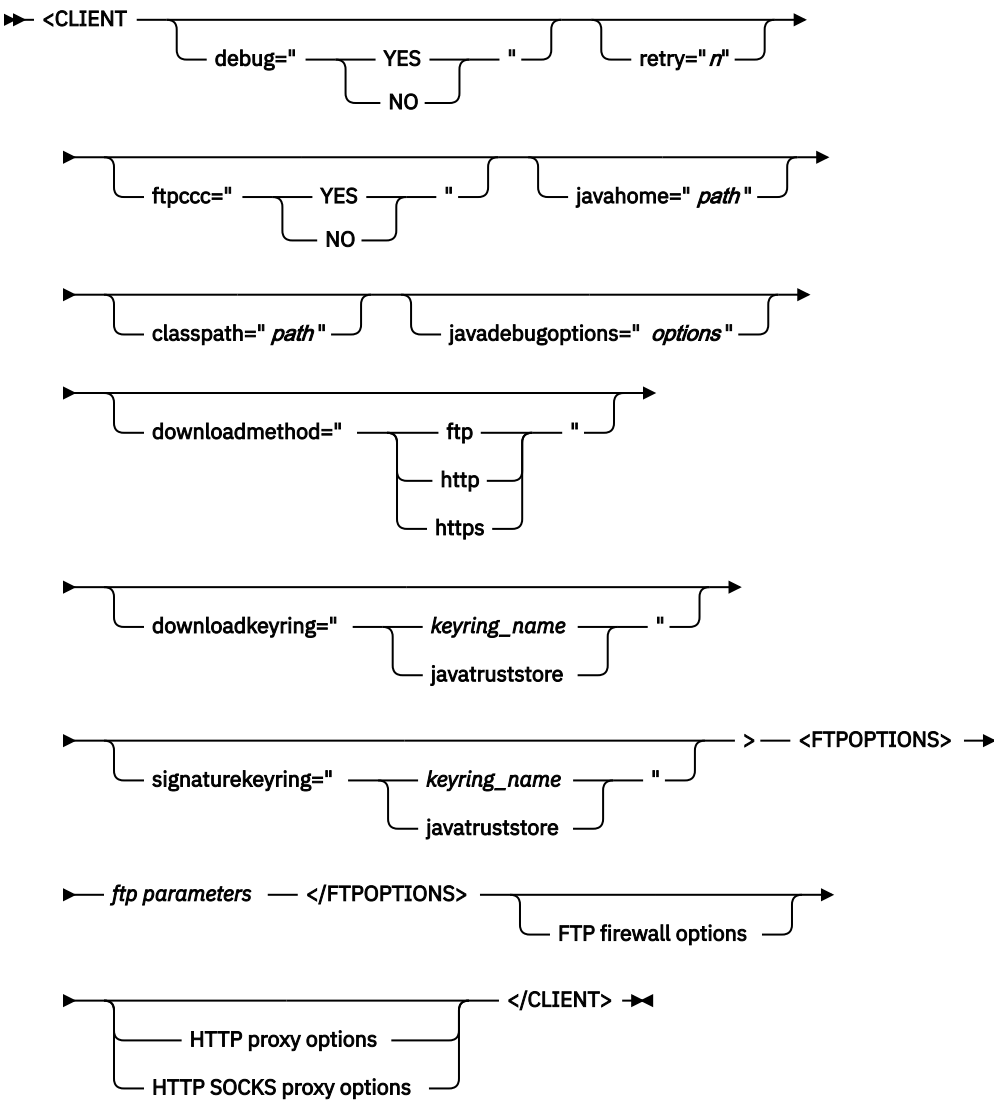
id
specifies a 1 to 50 character value assigned to identify the package by the SMP/E user. This value is used to name the subdirectory within the SMPNTS directory where the transferred files are staged. The value can contain any character from X'41' through X'FE', except a slash ("/"). The `id` attribute value is required and must be enclosed in quotation marks.

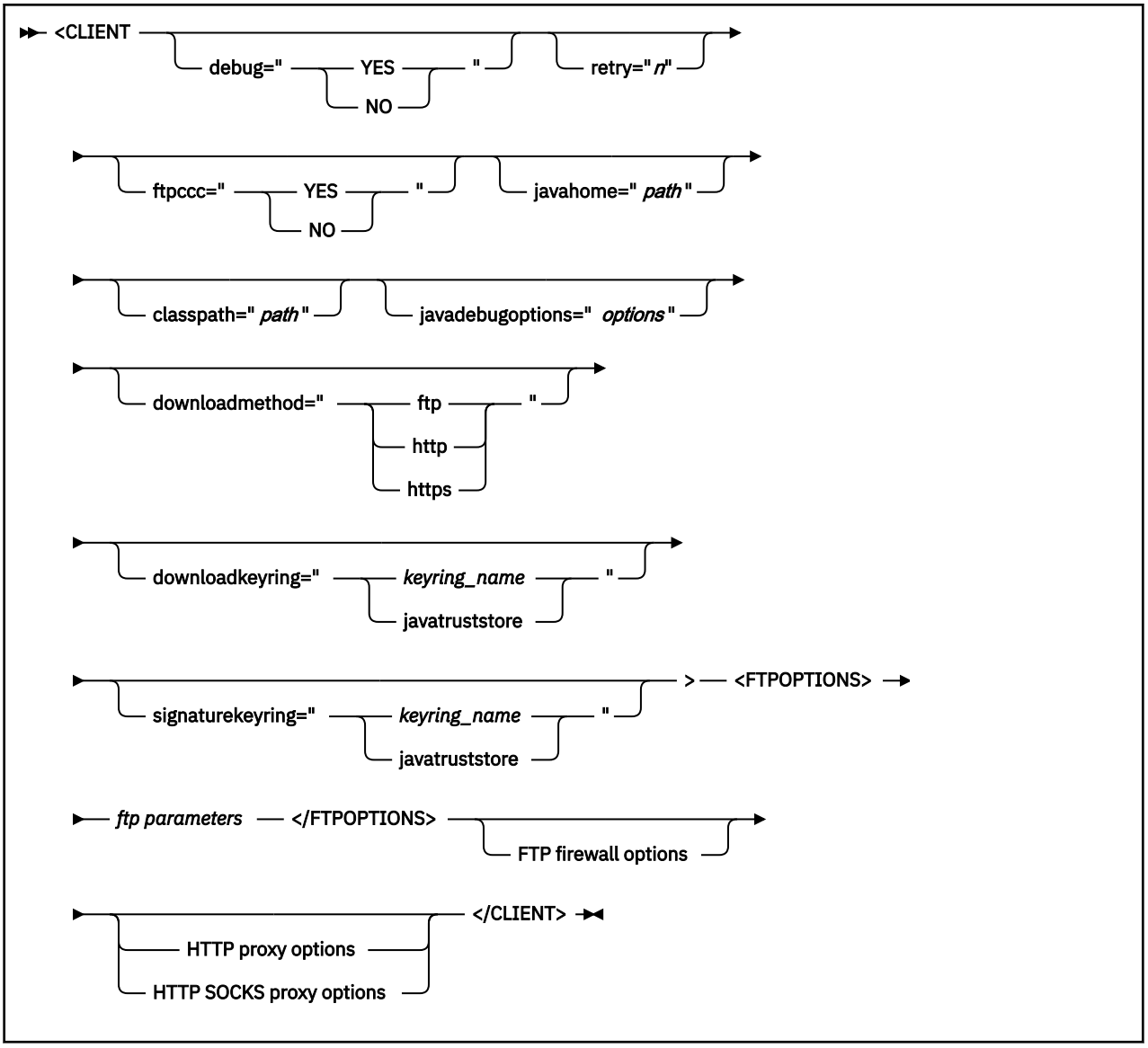
</PACKAGE>
specifies the end of PACKAGE data. This tag is required.

Content of CLIENT data set

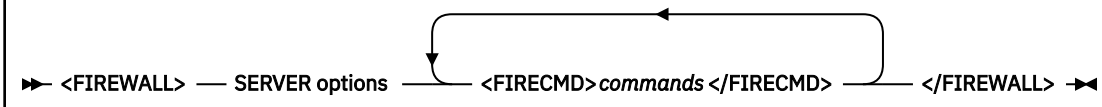
The CLIENT data set contains information about the environment of the local system, such as how to navigate an FTP firewall and an HTTP proxy server. See “Syntax rules for XML statements” on page 3 for a description of the general syntax rules that apply to this data set.

CLIENT tag

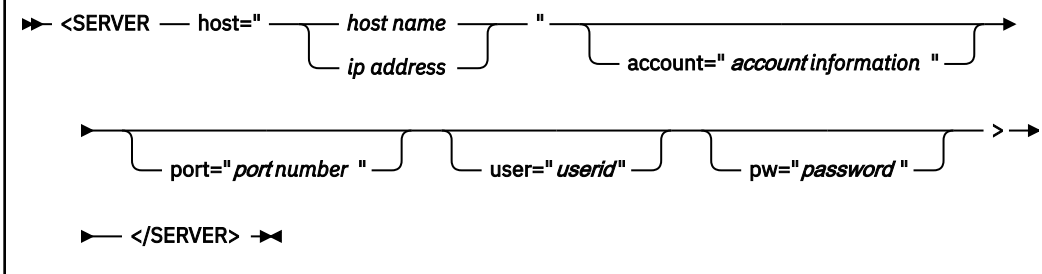

<CLIENT
debug="
YES
NO
"
retry="n"
ftpccc="
YES
NO
"
javahome="path"
classpath="path"
javadebugoptions="options"
downloadmethod="
ftp
http
https
"
downloadkeyring="
keyring_name
javatruststore
"
signaturekeyring="
keyring_name
javatruststore
"
>
<FTPOPTIONS>
ftp parameters
</FTPOPTIONS>
FTP firewall options
HTTP proxy options
HTTP SOCKS proxy options
</CLIENT>



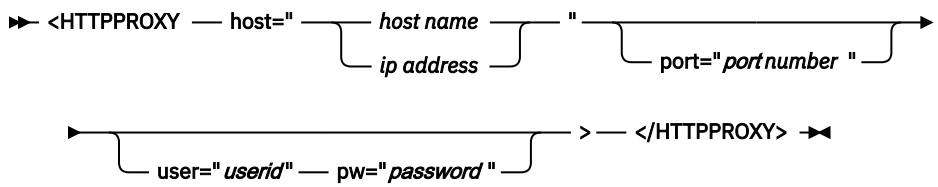
FTP firewall options



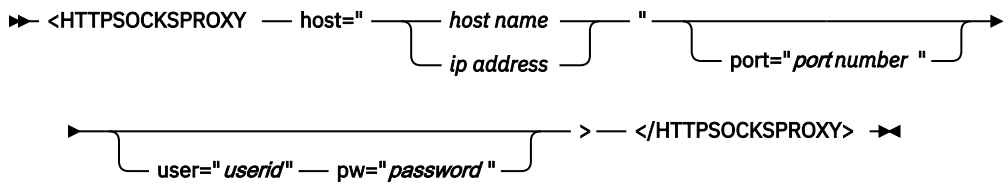
SERVER options



HTTP proxy options



HTTP SOCKS proxy options



<CLIENT>
specifies the start of the local environment data. This tag is required.

The attributes for the <CLIENT> tag are:

debug
specifies whether SMP/E should invoke the z/OS FTP client using the `-d` parameter to generate tracing output. A value of YES indicates that tracing output should be generated. A value of NO is equivalent to not specifying the debug attribute and indicates that tracing output should not be generated. The attribute value must be enclosed in quotation marks. The value may be entered in mixed case, but is folded to uppercase for verification. The debug attribute is optional.

retry
One numeric character (0-9) indicating the number of times to retry the transfer of a file when its SHA-1 hash value does not match the hash value expected for this file. This tag is optional. If the `retry` attribute is not specified, no retries will be attempted. This attribute must be enclosed in quotation marks.

ftpccc
specifies whether SMP/E should use the z/OS FTP client CCC subcommand. The CCC subcommand (Clear Command Channel) changes the FTP control connection from encrypted mode to clear text mode.

A value of YES is the default if *ftpccc* is not specified. It indicates that after the FTP client connects to the server, negotiates a secure and encrypted connection, and authenticates using user and password, SMP/E will send the CCC subcommand to change the control connection from encrypted to clear text. Subsequent clear text commands and responses on the FTP control connection can often help solve data connection problems for encrypted connections behind a Network Address Translation (NAT) firewall router.

A value of NO indicates a secure and encrypted control connection will not be changed to clear text. A value of NO is sometimes required when a firewall prohibits changing an encrypted FTP control connection to clear text.

The *ftpccc* attribute value must be enclosed in quotation marks. Also the value may be entered in mixed case, but is folded to uppercase for verification.

The *ftpccc* attribute is optional and its value is ignored if the FTP control connection is not encrypted.

javahome
specifies the path used to locate the Java runtime. SMP/E requires the Java runtime for several operations, including RECEIVE ORDER processing, downloading GIMZIP packages using HTTP or

HTTPS, and to calculate SHA-1 hash values for RECEIVE FROMNETWORK when ICSF is not active. If the Java runtime is not specified using an SMPJHOME DD statement or DDDEF entry, then the directory can be specified on the *javahome* attribute. For example, `javahome="/usr/lpp/java/J7.0"`. If a javahome value is specified, it must be a valid directory. A valid directory is from 1 to 255 characters in length, and must include only characters X'40' through X'FE' excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

Note: The javahome directory can also be specified using the SMPJHOME DD statement or DDDEF entry. When specified, these values override the *javahome* attribute value from the client data set. For more information about setting the javahome directory, see the chapter on "Preparing to use RECEIVE ORDER processing" in *z/OS SMP/E User's Guide*.

classpath
specifies the search path for the SMP/E Java application classes. If the classpath is not specified using an SMPCPATH DD statement or DDDEF entry, then the directory may be specified on the *classpath* attribute. For example, `classpath="/usr/lpp/smp/classes/"`. If classpath is not specified either in the client data set or by using an SMPCPATH DD statement or DDDEF entry, a default classpath value of `"/usr/lpp/smp/classes/"` is used. If a classpath value is specified, it must be a valid directory. A valid directory is from 1 to 255 characters in length, and must include only characters X'40' through X'FE' excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

Note: The classpath directory can also be specified using the SMPCPATH DD statement or DDDEF entry. When specified, these values override the *classpath* attribute value from the client data set. For more information about setting the classpath directory, see the chapter on "Preparing to use RECEIVE ORDER processing" in *z/OS SMP/E User's Guide*.

javadebugoptions
specifies a character string, provided to you by IBM, to indicate what debug and trace information should be produced when invoking the SMP/E java application classes. The debug and trace information is written to the print file for the HFSCOPY utility (SYSPRINT is SMP/E's default print file, and is used if no PRINT subentry was specified in the active UTILITY entry for the HFSCOPY utility).

If a debug options value is specified, it must be from 1 to 300 characters in length, and include only characters X'40' through X'FE' excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

Typical debug options that may be specified are as follows:

-showversion
to indicate the version information for the level of Java being used.

-Dcom.ibm.smp.debug=severe
provides detailed debug and trace information for the SMP/E java classes.

-Djava.security.debug
provides detailed debug information regarding java access control processing.

-Djavax.net.debug
provides detailed debug information for the phases of SSL/TLS processing.

downloadmethod
Identifies the network protocol to use for downloading the package files from the server to the local z/OS system, and may be a value of `ftp`, `http`, or `https`. If `downloadmethod` is not specified, the default is `ftp`. If `https` is specified, then certificates are required to perform the SSL handshake with the HTTPS server and thus to encrypt the network activity. The location of the necessary certificates is defined on the `downloadkeyring` attribute.

downloadkeyring
Identifies the location of the certificates required to perform SSL operations with the HTTPS server where the files to be downloaded reside. The name for a security manager keyring or the keyword `javatruststore` may be specified.

keyring-name
Identifies the name for a security manager keyring that contains the certificates required to perform SSL operations with the HTTPS server. The specified name may be for a real or virtual keyring. Keyring names are from 1 to 237 characters in length and may include only characters X'40' through X'FE' excluding the forward slash (/) and the reserved XML characters, less than (<), greater than (>), double quotation mark ("), and ampersand (&).

If the keyring is associated with a userid other than that used to execute SMP/E, then the keyring name must be prefixed with the associated userid. Userids are from 1 to 8 alphanumeric characters in length, can consist entirely of numerics and need not begin with any particular character. The userid is separated from the keyring value by a forward slash (userid/keyring).

To indicate all of the Certificate Authority (CA) certificates defined in the security manager may be used to perform SSL operations, use the CERTAUTH virtual keyring by specifying the userid/keyring value “*AUTH*/*”.

Note: The userid under which SMP/E runs must be properly authorized to the FACILITY class resource IRR.DIGTCERT.LISTRING or to the RDATALIB class resource *keyring-owner*.*keyring-name*.LST for SMP/E to use the specified keyring. READ access is required for a userid to use its own keyring or the CERTAUTH virtual keyring. UPDATE access is required to use a keyring from another user. If the RDATALIB class is used, for any real keyring, READ access to *keyring-owner*.*keyring-name*.LST is required. For CERTAUTH virtual keyring, READ access to CERTIFAUTH.IRR_VIRTUAL_KEYRING.LST is required.

javatruststore
Indicates all of the certificates in the default Java truststore may be used to perform the SSL handshake. A Java truststore is a Java keystore file containing the collection of trusted Certificate Authority (CA) certificates. The default Java truststore is located relative to the Java home directory, which is specified on the `javahome` attribute in the <CLIENT> tag, or on the SMPJHOME DD statement.

For the RECEIVE FROMNETWORK command and the GIMGTPKG service routine, if `downloadmethod=https`, then `downloadkeyring` is required. For the RECEIVE ORDER command, if `downloadmethod=https`, then the keyring can be identified either with the `downloadkeyring` attribute, or the `keyring` attribute in the ORDERSERVER data set.

signaturekeyring
Identifies the location of the Certificate Authority (CA) certificate required to validate the digital signature of the GIMZIP package. The name for a security manager keyring or the keyword `javatruststore` may be specified.

keyring-name
Identifies the name for a security manager keyring. The specified name may be for a real or virtual keyring. Keyring names are from 1 to 237 characters in length and may include only characters X'40' through X'FE' excluding the forward slash (/) and the reserved XML characters, less than (<), greater than (>), double quotation mark ("), and ampersand (&).

If the keyring is associated with a userid other than that used to execute SMP/E, then the keyring name must be prefixed with the associated userid. Userids are from 1 to 8 alphanumeric characters in length, can consist entirely of numerics and need not begin with any particular character. The userid is separated from the keyring value by a forward slash (userid/keyring).

To indicate all of the Certificate Authority (CA) certificates defined in the security manager may be used to perform SSL operations, use the CERTAUTH virtual keyring by specifying the userid/keyring value “*AUTH*/*”.

Note: The userid under which SMP/E runs must be properly authorized to the FACILITY class resource IRR.DIGTCERT.LISTRING or to the RDATALIB class resource *keyring-owner*.*keyring-name*.LST for SMP/E to use the specified keyring. READ access is required for a userid to use its own keyring or the CERTAUTH virtual keyring. UPDATE access is required to use a keyring from another user. If the RDATALIB class is used, for any real keyring, READ access

to *keyring-owner*.*keyring-name*.LST is required. For CERTAUTH virtual keyring, READ access to CERTIFAUTH.IRR_VIRTUAL_KEYRING.LST is required.

javatruststore
Indicates all of the certificates in the default Java truststore. A Java truststore is a Java keystore file containing the collection of trusted Certificate Authority (CA) certificates. The default Java truststore is located relative to the Java home directory, which is specified on the `javahome` attribute in the <CLIENT> tag, or on the SMPJHOME DD statement.

<FTPOPTIONS>
specifies the start of the FTP parameters section of the CLIENT data set. The <FTPOPTIONS> section can be specified only once.

ftp parameters
identifies the z/OS FTP Client parameters that are to be specified when SMP/E invokes the z/OS FTP Client program to transfer a package. SMP/E runs the FTP program in the z/OS UNIX environment, therefore the ftp parameters must be supplied in the standard UNIX flag format (for example, -d). See *z/OS Communications Server: IP User's Guide and Commands* for information about the FTP Client parameters.

If ftp parameters are specified, their value must be 1 to 300 characters in length, and include characters X'40' through X'FE' only, excluding the reserved XML characters, less than (<), greater than (>), and ampersand (&). SMP/E does not validate the specified parameters before passing them to the FTP Client program.

Note:

1. You do not need to specify the *-e* and *-d* FTP parameters in the <FTPOPTIONS> element. SMP/E automatically includes the *-e* parameter. Also, if the debug="yes" attribute is specified in the CLIENT tag, SMP/E includes the *-d* parameter, when invoking the FTP Client program.
2. Do not specify the host name or port number in the ftp parameters.

</FTPOPTIONS>
specifies the end of the FTP parameters section. This tag is required if <FTPOPTIONS> was specified.

<FIREWALL>
specifies the start of the firewall section of the local FTP client environment data.

The firewall section of the CLIENT data set should be specified if your TCP/IP environment requires that you connect to the firewall machine to execute FTP commands.

Following the <FIREWALL> tag are:

<SERVER>
specifies the start of the server information of the firewall section of the FTP client environment data. This tag is required if <FIREWALL> was specified.

The <SERVER> section of the <FIREWALL> entry contains the information that SMP/E needs to connect to the firewall host.

The attributes for the <SERVER> tag are:

host
identifies the firewall host. The `host` attribute must specify either:

host name
a fully qualified internet name for the firewall host that can be resolved by the domain name system to an internet address. A host name is a text string up to 255 characters drawn from the alphabet (A-Z), digits (0-9), period (.), and minus sign (-). Periods are allowed only as delimiters within domain style names. No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be a letter or a digit. The last character must not be a minus sign or period.

host ip address
an internet address defining the firewall host's location on the internet. An internet address can be an IPv4 or IPv6 address. An IP address can be up to 255 nonblank characters excluding the reserved XML characters, < (X'4C'), > (X'6E'), double quotation mark (X'7F') and ampersand (X'50').

One `host` attribute specifying either a host name or a host IP address is required. The `host` attribute value must be enclosed in quotation marks

user
id that will give you access to the firewall host machine. This attribute must be enclosed in quotation marks.

pw
specifies a password value that will give you access to this firewall host. This attribute must be enclosed in quotation marks.

Note: Users should use existing system facilities to restrict access to the data set named on the CLIENT operand to ensure the security of the password value.

account
specifies the account information to be used for the specified user ID. This attribute must be enclosed in quotation marks. This field is a text string up to 80 characters of any type. The `account` attribute is optional.

port
specifies the port number to be used for TCP/IP operations on the specified host. The port number must be a decimal number in the range 1 through 65535. This attribute value must be enclosed in quotation marks. The `port` attribute is optional.

</SERVER>
specifies the end of server section of the FTP client environment data. This tag is required if <SERVER> was specified.

<FIRECMD>
specifies the start of the firewall specific command section of the FTP client environment data. This tag is required if the <FIREWALL> tag is specified.

firewall specific commands
Enter the sequence of commands necessary to FTP to an outside host through your firewall. Each command must be on a separate line in the data set. Each command is a text string of up to 80 characters of any type.

SMP/E allows the following substitution variables within the commands in the <FIRECMD> section. Each substitution variable begins with the "&" character and ends with a semi-colon.

&ACCOUNT;
When `&ACCOUNT;` is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value specified in the `account` attribute of the <SERVER> section in the <FIREWALL> section is substituted into the command string. If the `account` attribute is omitted, blanks will be substituted.

&PORT;
When `&PORT;` is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value specified in the `port` attribute of the <SERVER> section in the <FIREWALL> section in this CLIENT data set, is substituted into the command string. If no `port` attribute was specified, a blank will be substituted.

&PW;
When `&PW;` is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value specified in the `pw` attribute of the <SERVER> section in the <FIREWALL> section of this CLIENT data set is substituted into the command string. If the `pw` attribute is omitted, blanks will be substituted.

&REMOTE_ACCOUNT;
When &REMOTE_ACCOUNT; is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value substituted into the command string depends on the form of the RECEIVE command being executed. For RECEIVE FROMNETWORK, the value specified in the account attribute entry of the <SERVER> section in the data set identified by the SERVER DD is substituted into the command string. If the account attribute is omitted, blanks will be substituted. For RECEIVE ORDER, the proper value is provided automatically by the IBM order server.

&REMOTE_HOST;
When &REMOTE_HOST; is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value substituted into the command string depends on the form of the RECEIVE command being executed. For RECEIVE FROMNETWORK, the value specified in the host attribute of the <SERVER> section in the SERVER data set is substituted into the command string. For RECEIVE ORDER, the proper value is provided automatically by the IBM order server.

&REMOTE_PORT;
When &REMOTE_PORT; is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value substituted into the command string depends on the form of the RECEIVE command being executed. For RECEIVE FROMNETWORK, the value specified in the port attribute of the <SERVER> section in the SERVER data set is substituted into the command string. If no port attribute was specified, a blank will be substituted. For RECEIVE ORDER, the proper value is provided automatically by the IBM order server.

&REMOTE_PW;
When &REMOTE_PW; is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value substituted into the command string depends on the form of the RECEIVE command being executed. For RECEIVE FROMNETWORK, the value specified in the pw attribute of the <SERVER> section in the data set identified by the SERVER DD is substituted into the command string. If the pw attribute is omitted, blanks will be substituted. For RECEIVE ORDER, the proper value is provided automatically by the IBM order server.

&REMOTE_USER;
When &REMOTE_USER; is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value substituted into the command string depends on the form of the RECEIVE command being executed. For RECEIVE FROMNETWORK, the value specified in the user attribute of the <SERVER> section in the data set identified by the SERVER DD is substituted into the command string. If the user attribute is omitted, blanks will be substituted. For RECEIVE ORDER, the proper value is provided automatically by the IBM order server.

&USER;
When &USER; is encountered within lines in the <FIRECMD> section of the CLIENT data set, the value specified in the user attribute of the <SERVER> section in the <FIREWALL> section in this CLIENT data set, is substituted into the command string. If the user attribute is omitted, blanks will be substituted

</FIRECMD>
specifies the end of the firewall specific command section of the FTP client environment data. This tag is required if <FIRECMD> was specified.

</FIREWALL>
specifies the end of firewall section of the FTP client environment data. This tag is required if <FIREWALL> was specified.

<HTTPPROXY>
specifies the start of the HTTP proxy section of the CLIENT data set. Data should be specified only if your network environment requires that you use a proxy server to perform HTTP communications with a remote server.

host
identifies the proxy server used for HTTP communications. This attribute value must be either:

hostname
a fully qualified internet host name that can be resolved by the domain name system to an internet address. Host name values are from 1 to 255 characters, from the set of mixed case alphanumeric, period (.), and minus sign (-). The first and last character must be alphanumeric.

ip address
an internet address defining the host's location on the internet. The internet address can be an IPv4 or IPv6 address. An IP address can be from 1 to 255 nonblank characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

port
specifies the port number to be used for TCP/IP operations with the specified proxy server. The port number must be a decimal number in the range 1 through 65535. The port value is optional, and SMP/E's default value is 80.

user
specifies a user ID that will allow access to the proxy server. User ID values can be from 1 to 80 characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

pw
specifies a password value that will allow access to the proxy server. Password values can be from 1 to 80 characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

</HTTPPROXY>
specifies the end of the HTTP proxy data.

<HTTPSOCKSPROXY>
specifies the start of the HTTP SOCKS proxy section of the CLIENT data set. Data should be specified only if your network environment requires that you use a SOCKS proxy server to perform HTTP communications with a remote server.

host
identifies the SOCKS proxy server used for HTTP communications. This attribute value must be either:

hostname
a fully qualified internet host name that can be resolved by the domain name system to an internet address. Host name values are from 1 to 255 characters, from the set of mixed case alphanumeric, period (.), and minus sign (-). The first and last character must be alphanumeric.

ip address
an internet address defining the host's location on the internet. The internet address can be an IPv4 or IPv6 address. An IP address can be from 1 to 255 nonblank characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

port
specifies the port number to be used for TCP/IP operations with the specified SOCKS proxy server. The port number must be a decimal number in the range 1 through 65535. The port value is optional, and SMP/E's default value is 1080.

user
specifies a userid that will allow access to the SOCKS proxy server. User ID values can be from 1 to 80 characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

pw
specifies a password value that will allow access to the SOCKS proxy server. Password values can be from 1 to 80 characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&).

</HTTPSOCKSPROXY>
specifies the end of the HTTP SOCKS proxy data.

</CLIENT>
specifies the end of local client environment data. This tag is required.

Example of FTP firewall data

Suppose that to download files using FTP from a remote host, you must connect to a local firewall server, logon to that firewall server by specifying a user ID and password, and then use the **SITE** command to connect to the remote FTP host. In this case, you would include information in the CLIENT data set, as shown in Figure 7 on page 258:

```
<CLIENT>
  <FIREWALL>
    <SERVER
      host="local.firewall.host"
      user="USER1" pw="PASSWORD1">
    </SERVER>
    <FIRECMD>&USER;</FIRECMD>
    <FIRECMD>&PW;</FIRECMD>
    <FIRECMD>SITE &REMOTE_HOST;</FIRECMD>
  </FIREWALL>
</CLIENT>
```

Figure 7. Example of CLIENT data set content

Navigating through a local firewall using either the RECEIVE FROMNETWORK or RECEIVE ORDER command and the GIMGTPKG service routine

In SMP/E V3R3, the RECEIVE FROMNETWORK command was enhanced to use the z/OS Communications Server FTP client. Due to how the FTP client and local firewalls prompt for a user ID and password, this enhancement requires changes to the *<FIRECMD>* tags used to navigate through a local firewall. The *<FIRECMD>* tags are specified in the CLIENT data set for the RECEIVE FROMNETWORK and RECEIVE ORDER commands and the SMPCLNT data set for the GIMGTPKG Service Routine.

Prior to SMP/E V3R3, you may have specified the following *<FIRECMD>* tags:

```
<FIRECMD>USER &USER;</FIRECMD>
<FIRECMD>PASS &PW;</FIRECMD>
```

With SMP/E V3R3 and later, if your local firewall or the FTP client prompts for the user ID, then do not specify the USER subcommand on the *<FIRECMD>* tag. Only specify the user ID or the appropriate substitution variable on the *<FIRECMD>* tag as shown here:

```
<FIRECMD>&USER;</FIRECMD>
```

Similarly, if your local firewall or the FTP client prompts for the password, then do not specify the PASS subcommand on the *<FIRECMD>* tag. Only specify the password or the appropriate substitution variable on the *<FIRECMD>* tag as shown here:

```
<FIRECMD>&PW;</FIRECMD>
```

With SMP/E V3R3 and later, you may need to add two additional *<FIRECMD>* tags if once connected to your local firewall, you are not yet connected to the remote server. To connect to the remote server, add the following *<FIRECMD>* tags:

```
<FIRECMD>USER &REMOTE_USER;</FIRECMD>
<FIRECMD>&REMOTE_PW;</FIRECMD>
```

In the end, the commands you specify in the *<FIRECMD>* tags should be the same as those you use with the z/OS Communications Server FTP client. Since the behavior of various firewalls differ, the best way to determine the necessary *<FIRECMD>* tags is to perform an FTP operation using the FTP client in a JCL job and then specify the same commands that were needed in that operation in the *<FIRECMD>* tags in the CLIENT and/or SMPCLNT data set.

```
//job      JOB job parameters...
//FTP      EXEC PGM=FTP
//OUTPUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//INPUT DD *
firewall_host
firewall_userid
firewall_password
SITE remote_host
; This is a comment line
; USER remote_userid ; uncomment if needed
; remote_password ; uncomment if needed
DIR
QUIT
```

Figure 8. Example FTP JCL job:

FTP.DATA file

The z/OS Communications Server provides an FTP client program to perform file transfer operations. SMP/E uses this FTP client program to transfer packages for RECEIVE FROMNETWORK and RECEIVE ORDER command processing, and the GIMGTPKG service routine. Certain functions of the FTP client program, such as performing file transfers in a secure mode or properly navigating local SOCKS firewalls, require the use of a configuration file (FTP.DATA) to the FTP client program. If you run FTP on z/OS V1R8 or higher, you can use the *-f* FTP parameter to identify a specific FTP.DATA file. You can specify the *-f* parameter using the <FTPOPTIONS> tag in the CLIENT data set.

```
<FTPOPTIONS>-f "//'USER1.FTP.DATA'"</FTPOPTIONS>
```

Note: Support for the <FTPOPTIONS> tag in the CLIENT data set is added to SMP/E V3R3 and later with APAR IO05806.

Using the *-f* parameter overrides the default search order used by the FTP client program to find the FTP.DATA file. If you do not specify the *-f* parameter, or if you are using a lower level of the FTP client program, the default search order for FTP.DATA is as follows:

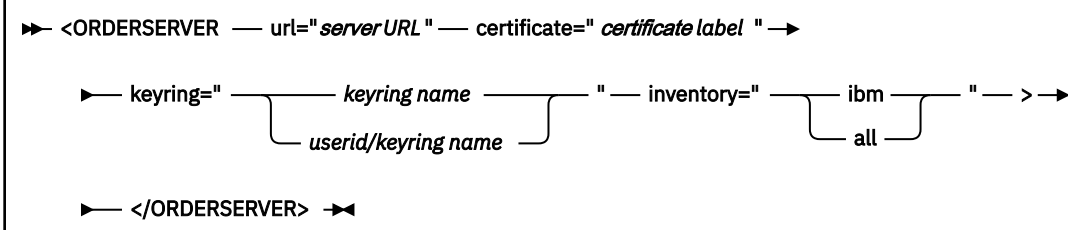
1. $HOME/ftp.data
2. *userid*.FTP.DATA
3. /etc/ftp.data
4. SYS1.TCPPARMS(FTPDATA) data set
5. *tcpip_hlq*.FTP.DATA file

For information about the contents of the FTP.DATA file and the *-f* parameter, see *z/OS Communications Server: IP User's Guide and Commands*. Also, ensure that you have the fix for APAR PK31326 before you use the *-f* parameter.

Content of ORDERSERVER data set

The ORDERSERVER data set provides the necessary information about the IBM order server that will fulfill an SMP/E HOLDDATA or PTF order request. The information in the data set is described using the new <ORDERSERVER> tag and attributes. See "Syntax rules for XML statements" on page 3 for a description of the general syntax rules that apply to this data set.

ORDERSERVER Tag



<ORDERSERVER>
specifies the start of ORDERSERVER data. This tag is required.

The attributes for the <ORDERSERVER> tag are:

url
specifies the Uniform Resource Locator (URL) for the order server. A URL value can be from 1 to 255 nonblank characters excluding the reserved XML characters, less than (<), greater than (>), double quotation mark (") and ampersand (&). A URL value is of the form:

```
protocol://host[:port][/path]
```

where host is either a host name or IP address. For example, `url="https://eccgw01.boulder.ibm.com/services/projects/ecc/ws/".`

The actual order server urls are as follows:

```
https://eccgw01.boulder.ibm.com/services/projects/ecc/ws/
```

and

```
https://eccgw02.rochester.ibm.com/services/projects/ecc/ws/
```

certificate
specifies the label to identify which certificate to use for access to the server. This certificate is generated during the registration process on IBM Shopz (www.ibm.com/software/shopzseries/ShopzSeries_public.wss) and is stored in a security product data base on your z/OS system. The certificate is used to identify and authenticate the user to the server. Certificate labels are from 1 to 32 characters in length and may include only characters X'40' through X'FE' excluding the reserved XML characters, less than (<), greater than (>), double quotation mark ("), and ampersand (&). See *z/OS SMP/E User's Guide* for detailed information on using ShopzSeries to obtain a certificate and how to store the certificate in your security product data base (RACF®).

keyring
specifies the security manager key ring that contains the certificate required for access to the server. Key ring names are from 1 to 237 characters in length and may include only characters X'40' through X'FE' excluding the forward slash (/) and the reserved XML characters, less than (<), greater than (>), double quotation mark ("), and ampersand (&).

If the key ring is associated with a user ID other than that used to execute SMP/E, then the key ring name must be prefixed with the associated user ID. User IDs are from 1 to 8 alphanumeric characters in length, can consist entirely of numbers and need not begin with any particular character. The user ID is separated from the key ring value by a forward slash (userid/keyring).

inventory
specifies which form of the software inventory will be included in the request to the order server. A value of `ibm` indicates the software inventory will be produced in the form expected by the IBM Automated Service Request server. This form of the software inventory identifies all installed FMIDs and only PTFs whose IDs match the naming convention used by IBM. This is the default value.

A value of `all` indicates the software inventory will be produced in a generic form that identifies all installed FMIDs and PTFs regardless of their naming convention.

</ORDERSERVER>
specifies the end of ORDERSERVER data. This tag is required.

Output

This section describes the listings and reports produced during RECEIVE processing.

Listings

SMP/E does not print the SYSMOD or exception modification control statements during RECEIVE processing unless you specify the LIST operand on the RECEIVE command. If LIST is specified, the following occurs:

- If SELECT or EXCLUDE is specified, the message control statements for those SYSMODs either selected or not specifically excluded are written to SMPOUT.
- If a SYSMOD is not applicable, no modification control statements are listed for that SYSMOD.

  Note: For SYSMODs with construction errors, SMP/E lists the modification control statements up to the point of the error.
- Header modification control statements (that is, ++FUNCTION, ++PTF, ++APAR, ++USERMOD) whose SYSMOD ID does not appear in the first record are written to SMPOUT, even though the SYSMOD was not selected or was excluded.

Reports

These reports are produced during RECEIVE processing:

- File Allocation report
- RECEIVE Exception SYSMOD Data report
- RECEIVE Summary report
- RECEIVE Product Summary report

For descriptions of these reports, see Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the RECEIVE command.

Example 1: Receiving SYSMODs and HOLDDATA

Periodically, you must install service and process the related HOLDDATA for functions on your system. IBM supplies you with service and HOLDDATA on CBPDO packages and ESO tapes. Both of these contain service and HOLDDATA, plus ++ASSIGN statements, which assign source IDs for PTFs that have been received.

When installing products and service, the first step is to receive the service and HOLDDATA into your SMP/E data sets. Here is a sample job that receives SYSMODs and HOLDDATA from a CBPDO and assigns them a source ID of PDO9824:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  SOURCEID(PDO9824)  /* Receive SYSMODs and
                               assign SOURCEID.         */.
```

Besides receiving SYSMODs and HOLDDATA from the tape, SMP/E assigns a source ID of PDO9824 to the SYSMODs it has just received. In addition, it assigns the source IDs specified on the ++ASSIGN statements to SYSMODs that were just received or that had been previously received.

These source IDs can be specified later on the APPLY and ACCEPT commands to limit which SYSMODs should be installed.

Example 2: Receiving HOLDDATA only

If you do not want to keep the SYSMODs from your service tape in the SMPPTS until you are ready to install them, you should ensure that SMP/E has at least the exception SYSMOD information contained on that tape. This is important, as the information could affect some of those SYSMODs that are present on the SMPPTS and prevent inadvertent application of a PE-PTF. To process only the exception SYSMOD data, the following commands can be used:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  HOLDDATA           /* Receive HOLDDATA.
                               Note no SOURCEID is
                               assigned to HOLDDATA.    */.
```

Example 3: Receiving SYSMODs only

If you choose to receive **only** exception data from the service tape (as in Example 2), you must receive those SYSMODs when you are ready to actually install them (during APPLY or ACCEPT command processing). This can be done by specifying the following commands:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  SYSMODS            /* Receive SYSMODs only
                               and assign               */
         SOURCEID(MYSRCID)  /* SOURCEID.                */.
```

The SOURCEID operand assigns each of the SYSMODs received a SOURCEID value of MYSRCID that you can use during APPLY or ACCEPT to assist in selecting SYSMODs. SMP/E also assigns this source ID to any SYSMODs that would have been received, but were not received because they had already been received.

Example 4: Receiving selected SYSMODs and HOLDDATA

The RECEIVE command can be used to select individual SYSMODs or exception SYSMOD data applicable to selected SYSMODs. In this example, the commands cause SMP/E to receive two SYSMODs specified plus any exception SYSMOD data applicable to the SYSMODs:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  S(UZ12345,UZ12346) /* Receive selected         */
         SYSMODS            /* SYSMODs                  */
         HOLDDATA           /* and HOLDDATA applicable
                               to them,                 */
         SOURCEID(MYSRCID)  /* and assign a SOURCEID.   */.
```

Note: The SYSMODs and HOLDDATA operands can be left off the RECEIVE command; processing is the same.

Example 5: Receiving SYSMODs and HOLDDATA for a specific FMID

You may want to receive SYSMODs and HOLDDATA for a particular FMID or FMIDSET. The following commands receive SYSMODs and HOLDDATA for the specified FMID, plus the selected SYSMOD:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  FORFMID(HXP1400)   /* Receive SYSMODs and
```

```
                              HOLDDATA for HXP1400,    */
         S(UZ12345)         /* plus selected SYSMOD.    */.
```

Example 6: Receiving RELFILEs from DASD

Suppose a PTF packaged in RELFILE format (UR00001) has been shipped to you electronically, and you have read the RELFILEs into DASD data sets. The high-level qualifier you used for the data set names is MYHLQ; the rest of the data set name follows the naming convention required by SMP/E. For example, the name of the data set containing RELFILE 1 is 'MYHLQ.UR00001.F1'. Now you want to receive the PTF from the DASD data sets. The following commands accomplish this task:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  SYSMODS            /* Receive SYSMODs          */
         SOURCEID(IBMLINK)  /* and assign a SOURCEID.   */
         RFPREFIX(MYHLQ)    /* HLQ, RELFILEs on DASD.   */.
```

Example 7: Excluding SYSMODs selected with an FMIDSET

A SYSMOD ID defined in an FMIDSET specified on the SELECT list may be excluded from processing with the EXCLUDE operand, as shown in this example:

If FMIDSTX contains FUNC001, FUNC002, FUNC003, and FUNC004, then, to RECEIVE all but FUNC003, the command would be:

```
RECEIVE SELECT(FMIDSTX) EXCLUDE(FUNC003).
```

Example 8: Issuing an internet service retrieval request

The following is an example RECEIVE ORDER command to submit an Internet Service Retrieval request for PTFs from the IBM Automated Delivery Request server. This example orders two specific PTFs (UQ12345 and UQ98765) and any requisites for these PTFs that are not already present in the ZOS14 target zone.

```
//jobname  JOB ...
//RECEIVE  EXEC PGM=GIMSMP
//SMPCSI   DD DSN=SMPE.GLOBAL.CSI,DISP=SHR
//SMPNTS   DD PATH='/u/smpe/smpnts/',PATHDISP=KEEP
//SMPOUT   DD SYSOUT=*
//SMPRPT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
  SET      BOUNDARY(GLOBAL).
  RECEIVE  SYSMODS HOLDDATA
           ORDER(                   /* Place an order for service */
             ORDERSERVER(ORDRSRVR)
             CLIENT(MYCLIENT)
             CONTENT(
               PTFS(UQ12345,UQ98765) /* Get these PTFs, and any.. */
                    )               /* ..requisites..           */
             FORTGTZONES(ZOS14)     /* ..for this target zone     */
                ).
/*
//ORDRSRVR DD *
  <ORDERSERVER
    url="https://eccgw01.boulder.ibm.com/services/projects/ecc/ws/"
    keyring="MRWKYRNG"
    certificate="SMPE Client Certificate">
  </ORDERSERVER>
/*
//MYCLIENT DD *
   <CLIENT
     javahome="/usr/lpp/java/J1.4"
     classpath="/usr/lpp/smp/classes">
   </CLIENT>
```

Note: An alternative URL for the IBM Automated Delivery Request server is `https://eccgw02.rochester.ibm.com/services/projects/ecc/ws`.

Example 9: Downloading a pending order

The following is an example RECEIVE command to download the package for an order that is in a pending state.

```
//jobname  JOB ...
//RECEIVE  EXEC PGM=GIMSMP
//SMPCSI   DD DSN=SMPE.GLOBAL.CSI,DISP=SHR
//SMPNTS   DD PATH='/u/smpe/smpnts/',PATHDISP=KEEP
//SMPOUT   DD SYSOUT=*
//SMPRPT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
  SET      BOUNDARY(GLOBAL).
  RECEIVE  SYSMODS HOLDDATA
           ORDER(
           PENDING(ORD00035) /* Get the specified pending order  */
           CLIENT(MYCLIENT)
                ).
//MYCLIENT DD*
  <CLIENT
    javahome="/usr/lpp/java/J1.4"
    classpath="/usr/lpp/smp/classes">
  </CLIENT>
/*
```

In this example, SMP/E determines if the HOLDDATA or PTF order associated with the specified ORDER entry is ready for downloading and, if so, downloads the ORDER package.

Processing input from SMPPTFIN and SMPHOLD

RECEIVE processing includes these steps:

- Processing relative files
- Selecting SYSMODs
- Selecting ++FEATURE and ++PRODUCT statements
- Selecting ++HOLD and ++RELEASE statements
- Processing SYSMODs
- Processing ++ASSIGN statements
- Processing ++FEATURE and ++PRODUCT statements
- Processing ++HOLD and ++RELEASE statements
- Compacting inline data

Processing relative files

SMP/E can receive SYSMODs packaged in relative file format from DASD, as well as tape. If SMP/E determines that the SMPPTFIN data set is located on tape, it assumes that the relative files are on tape. Otherwise, SMP/E assumes the RELFILEs are on DASD and are cataloged. (If SMP/E assumes the RELFILEs are on DASD but they actually are not, allocation fails for the RELFILEs.)

If a SYSMOD being received is packaged in relative files, those files are loaded into temporary direct access data sets as part of RECEIVE processing. (For more information about packaging SYSMODs in relative files, see Standard Packaging Rules for z/OS-based Products (publibz.boulder.ibm.com/epubs/pdf/gimpkg80.pdf).) These temporary data sets, called SMPTLIBs, can be allocated before RECEIVE processing, or they can be dynamically allocated during RECEIVE processing. For more information, see “Receiving SYSMODs packaged in relative files” on page 243. (Regardless of how the SMPTLIB data sets are allocated, they do not appear in the File Allocation report.) When the RELFILEs are on DASD, SMP/E checks to ensure that the RELFILE data set name is not the same as the SMPTLIB data set name. If it is, RECEIVE processing stops.

To allocate SMPTLIB data sets, SMP/E checks the following, in the order shown:

1. **Catalog information:** SMP/E first tries to allocate the data set through the catalog. Therefore, if the SMPTLIB data set is preallocated and already cataloged, SMP/E uses that data set instead of allocating a new one.
2. **Volume information:** If the data set was not allocated through the catalog, but a volume list was specified in either the SMPTLIB DD statement or the SMPTLIB DDDEF entry, SMP/E searches the specified volumes to see whether the data set was preallocated on any of them.
   - If so, SMP/E uses the preallocated data set instead of allocating a new one.
   - Otherwise, SMP/E tries to dynamically allocate a new data set on each specified volume using the parameters specified in the SMPTLIB DD statement or the SMPTLIB DDDEF entry (and the DSSPACE value in the current OPTIONS entry, if specified) until the data set is allocated.
3. **No catalog or volume information:** If no catalog or volume information is available to allocate the SMPTLIB data sets, SMP/E tries to dynamically allocate a new data set using the parameters specified in the SMPTLIB DD statement or the SMPTLIB DDDEF entry (and the DSSPACE value in the current OPTIONS entry, if specified).

Dynamically allocating SMPTLIB data sets

When dynamically allocating data sets, SMP/E uses information provided on the SMPTLIB DD statement, SMPTLIB DDDEF entry, the current OPTIONS entry, and GIMDDALC control statements in SMPPARM member GIMDDALC. If there is a DD statement for the SMPTLIBs, SMP/E uses the volume specified on that statement. It then checks the SMPTLIB DDDEF entry to get additional information, such as space allocation parameters and the data set prefix. If SMP/E cannot find that information in the DDDEF entry, or if there is no SMPTLIB DDDEF entry, SMP/E gets the information from the OPTIONS entry that is in effect. Finally, SMP/E will use information provided in GIMDDALC control statements in SMPPARM member GIMDDALC. (SMP/E always attempts to allocate data sets, even if no volume or space allocation information is provided.) If SMP/E finds a data set prefix, it assigns each SMPTLIB data set the name *prefix.sysmod_id.*F*nnnn*, where:

prefix
is defined by the DSPREFIX value in the DDDEF or OPTIONS entry. If there is no DSPREFIX value, no high-level qualifier is assigned.

Note:

1. Any prefix value (RFDSNPFX or RFPREFIX) specified for the associated RELFILE data set name is **not** included in the SMPTLIB data set name.
2. If the RELFILE data sets are on DASD or are on tape and are cataloged, the RELFILE data set name must not match the name to be used for the SMPTLIB data sets. If these data sets have the same name, the SMPTLIB data sets cannot be allocated. The DSPREFIX value in the OPTIONS entry is used to specify the high-level qualifier for the SMPTLIB data set names. For a description of the OPTIONS entry, see *z/OS SMP/E Reference*.

sysmod_id*.F*nnnn
is the data set name of the relative file being loaded. *sysmod_id* is the ID of the associated SYSMOD, and *nnnn* is the relative file number of the file (for example, F1 or F2).

SMP/E determines the DSNTYPE value to use for the SMPTLIB data sets as follows:

1. SMP/E checks the format of the associated RELFILE data set and uses this information to determine the appropriate DSNTYPE value for the SMPTLIB data set, as indicated in Table 18 on page 265. Any DSNTYPE value specified in the SMPTLIB DDDEF entry is ignored. (Letting SMP/E determine the DSNTYPE based on the RELFILE data set is the recommended approach.)

Table 18. Relationship between format of RELFILE data set and DSNTYPE value for SMPTLIB data set

Format of RELFILE data set	**DSNTYPE value used for SMPTLIB data set**
DSNTYPE(PDS)	DSNTYPE(PDS)
DSNTYPE(LIBRARY)	DSNTYPE(LIBRARY)

Table 18. Relationship between format of RELFILE data set and DSNTYPE value for SMPTLIB data set (continued)

Format of RELFILE data set	DSNTYPE value used for SMPTLIB data set
Unloaded PDS	DSNTYPE(PDS)
Unloaded PDSE	DSNTYPE(LIBRARY)

2. If SMP/E cannot determine the DSNTYPE value of the RELFILE data set, it uses the DSNTYPE value specified in the SMPTLIB DDDEF entry.
3. If no DSNTYPE value is specified in the SMPTLIB DDDEF entry, SMP/E does not pass a DSNTYPE value to dynamic allocation. In this case, the DSNTYPE value is determined by the system default or by the SMS subsystem.

Note: It is possible that the system on which you are running SMP/E does not support PDSEs:

- It does not support PDSEs at all.
- It does not support PDSE load libraries (program objects).
- SMS is not activated or set up to support PDSEs.

If the system level does not support PDSEs, SMP/E detects this, issues a warning message, and does not attempt to allocate SMPTLIB data sets as PDSEs when determining the DSNTYPE to use. In the other cases, attempts to allocate PDSEs fail and error messages are issued indicating the reason for the failure.

Cataloging SMPTLIB data sets

SMP/E automatically catalogs SMPTLIB data sets that have been either dynamically allocated or preallocated, but not cataloged. It uncatalogs the SMPTLIB data sets when it deletes them.

Loading the relative files

SMP/E uses IEBCOPY or a user-defined copy utility to load the relative files into the SMPTLIB data sets. (A user-defined utility is specified through OPTIONS and UTILITY entries.) Each element defined by an MCS in the SYSMOD is selectively copied, including each ALIAS, DALIAS, TALIAS, and MALIAS name. This selective copying ensures that the relative files contain the correct elements. If the copy is successful, any unused space in the SMPTLIB data sets is released.

Note:

1. Any required aliases must be in the distribution libraries before the libraries are put into relative files. Relative files represent distribution library data sets for a function. When SMP/E loads the relative files, it copies in only members from the unloaded data sets; it does not cause aliases to be created for those members.
2. Errors can occur if the DSNTYPE value for a RELFILE is incompatible with the associated SMPTLIB data set or with the system on which you are running. Here are some examples:
   - An SMPTLIB data set was preallocated with a DSNTYPE value that is incompatible with that of the associated RELFILE data set, and the RELFILE data set contains ++MOD elements for load modules or program objects.
   - An SMPTLIB data set was dynamically allocated, SMP/E could not determine the DSNTYPE value of the associated RELFILE data set, and the RELFILE data set contains ++MOD elements for load modules or program objects.
   - The system on which you are running does not support program objects, and RECEIVE is attempting to copy program objects from a RELFILE data set into an SMPTLIB data set.

When a SYSMOD has been received and its relative files have been loaded, SMP/E saves the high-level qualifier for the SMPTLIB data sets as the TLIBPREFIX value in the global zone SYSMOD entry for that SYSMOD. You can change the DSPREFIX value in the DDDEF or OPTIONS entry that was used without affecting any SYSMODs that have already been received.

Sometimes errors occur while the SMPTLIB data sets are being allocated or the relative files are being unloaded. In this case, the return code from IEBCOPY may be higher than the SMP/E default value or the maximum return code value defined in the COPY UTILITY entry that is in effect. If such an error occurs, SMP/E cannot receive the SYSMOD, and it deletes all the SMPTLIB data sets associated with that SYSMOD, even if they were preallocated.

Selecting SYSMODs

A SYSMOD is selected if it meets all the following conditions:

1. You have either explicitly or implicitly selected the SYSMOD.
   - You explicitly select a SYSMOD by specifying it on the SELECT operand (select-mode).
   - You implicitly select a SYSMOD by omitting the SELECT operand (mass-mode) or by specifying the FORFMID operand with an FMID (or FMIDSET) to which this SYSMOD belongs.
2. The SYSMOD is not specified in the EXCLUDE list.
3. If you have implicitly selected the SYSMOD, the SYSMOD will remain selected only if one of the following conditions is true:
   a. No Receive Zone List was specified on either the ZONEGROUP operand of the RECEIVE command or in the RECZGRP and RECEXZGRP subentries of the active OPTIONS entry.
   b. The SYSMOD has not been applied or accepted into any of the zones defined in the Receive Zone List.
   c. The SYSMOD has been applied or accepted into one or more of the Receive Zone List zones, but BYPASS(APPLYCHECK) or BYPASS(ACCEPTCHECK) have been specified on the RECEIVE command.

      Note: The SYSMOD is deemed applied or accepted as long as the SYSMOD is not in error.
4. Either:
   a. The SYSMOD is a base function and is applicable to one of the SRELs (system releases) defined in the global zone.
   b. The SYSMOD is a service SYSMOD or a dependent function and is applicable to one of the SRELs and one of the FMIDs defined in the global zone.

      Note: If BYPASS(FMID) was specified, SMP/E does not check whether the applicable FMID for service is already defined in the global zone. In this case, SMP/E selects all SYSMODs that are applicable to an SREL defined in the global zone and to all exception SYSMOD data.
5. An entry does not already exist in both the SMPPTS and the global zone for the SYSMOD.

   Note: If an entry exists in either the global zone or the SMPPTS, but not both, SMP/E assumes that an error occurred during a previous RECEIVE attempt. In this case, it selects the current SYSMOD and replaces any existing entries for that SYSMOD in the global zone or SMPPTS.

The status of a SYSMOD's requisites has no effect on whether that SYSMOD is selected. These requisites are checked and resolved later when the SYSMOD is applied and accepted.

Generally, a SYSMOD that has already been successfully received cannot be received again unless it is first rejected. However, SMP/E may automatically rereceive a SYSMOD if both of these conditions are met:

- The REWORK operand is coded on the ++PTF, ++FUNCTION, ++APAR, or ++USERMOD statement. Typically, SYSMODs with the REWORK operand have been reworked by IBM for minor changes.
- The REWORK level is higher than the level for the previous version of the SYSMOD.

This saves you from having to reject and receive again the SYSMODs yourself. For more information about the REWORK operand, see the descriptions of these modification control statements in the "SMP/E Modification Control Statements" section in *z/OS SMP/E Reference*.

Note: If a SYSMOD appears more than once in the SMPPTFIN data set, the first occurrence may be received. However, none of the subsequent versions of the SYSMOD are received, even if their REWORK level is higher than the one for the first version of the SYSMOD. (Message GIM40001E is issued for each of the subsequent versions of the SYSMOD.)

Selecting ++PRODUCT and ++FEATURE statements

++PRODUCT and ++FEATURE MCS are selected whenever the RECEIVE command indicates that SYSMODs should be processed. If only HOLDDATA is being processed, ++PRODUCT and ++FEATURE MCS are not selected.

A ++PRODUCT MCS is selected if it meets the following conditions:

1. A PRODUCT entry of the same name does not already exist in the global zone.
2. The ++PRODUCT MCS specifies an SREL value matching an SREL value also listed in the global zone definition.

A ++FEATURE MCS is selected if it meets all these conditions:

1. A FEATURE entry of the same name does not already exist in the global zone.
2. The PRODUCT operand of the ++FEATURE MCS specifies an associated PRODUCT that either already exists as a PRODUCT entry in the global zone or as an ++PRODUCT MCS that is currently being received and that precedes the ++FEATURE MCS in SMPPTFIN.
3. If SELECT is specified (and FORFMID is not), then the ++FEATURE MCS specifies at least one FMID value that matches the SYSMOD ID of a SYSMOD specified on the SELECT operand, is in the FMID subentry of the GLOBALZONE entry, and is not specified in the EXCLUDE list.
4. If FORFMID is specified (and SELECT is not), then the ++FEATURE MCS specifies at least one FMID value that
   - matches the SYSMOD ID of a SYSMOD specified on the FORFMID operand, or matches the SYSMOD ID of a SYSMOD that is included for processing by the FORFMID operand, and
   - is in the FMID subentry of the GLOBALZONE entry, and
   - is not specified in the EXCLUDE list.
5. If both SELECT and FORFMID are specified, then at least one of them selects the ++FEATURE MCS, as previously described.
6. If neither SELECT nor FORFMID is specified, then the FMID operand of the ++FEATURE MCS either:
   a. specifies at least one FMID value that is in the FMID subentry of the GLOBALZONE entry, or
   b. specifies at least one FMID value that is currently being received, is not specified in the EXCLUDE list, and identifies a function SYSMOD that precedes the ++FEATURE MCS in the SMPPTFIN data set, or
   c. was omitted.

Note: If BYPASS(FMID) is specified, SMP/E does not require that the ++FEATURE statement specify an FMID that is in the FMID subentry of the GLOBALZONE entry.

Generally, a ++FEATURE or ++PRODUCT MCS that has already been successfully received cannot be received again unless it is first rejected. However, as with SYSMODs, SMP/E can sometimes automatically rereceive a ++FEATURE or ++PRODUCT MCS. The discussion of the REWORK operand in "Selecting SYSMODs" on page 267 also applies to the ++FEATURE and ++PRODUCT MCS.

Selecting ++HOLD and ++RELEASE statements

SMP/E determines which ++HOLD and ++RELEASE statements (collectively called HOLDDATA) to select from SMPHOLD according to whether the SELECT or FORFMID operand is specified.

- If SELECT or FORFMID is specified, SMP/E selects HOLDDATA that is applicable to the SYSMODs included by these operands.
- If neither SELECT nor FORFMID is specified, SMP/E selects HOLDDATA whose applicable FMID is defined in the global zone.

Processing SYSMODs

For each SYSMOD selected for RECEIVE processing, SMP/E takes these actions: :

- Saves the complete SYSMOD, unchanged and including any inline changes, as a member in the SMPPTS data set. This member is called an *MCS entry*.
- Creates a SYSMOD entry in the global zone. This entry contains information SMP/E needs to determine SYSMOD applicability during later SMP/E processing. For example, if a source ID was specified on the RECEIVE command, SMP/E saves that value in the SYSMOD entry.

  Note:

  1. If a SYSMOD entry already exists, but contains only HOLD reason IDs, those reason IDs are saved in the SYSMOD entry for the SYSMOD that was received.
  2. If a SYSMOD was received again, the existing SYSMOD entry in the global zone is completely replaced, except for the ACCID and APPID subentries. These are saved in the new SYSMOD entry so there is still a record of the zones where the SYSMOD has already been installed.
- Adds the FMID of each function SYSMOD received to the global zone. This enables SMP/E to receive SYSMODs applicable to that function SYSMOD.

modification control statement entries for SYSMODs that were not selected are not saved in the SMPPTS data set.

Processing ++ASSIGN statements

For each ++ASSIGN statement that was successfully processed, SMP/E associates the source ID with the specified SYSMODs. The source ID is assigned only to SYSMODs that have entries in the global zone. If the same SYSMOD is specified on more than one ++ASSIGN statement, all the source IDs are associated with the SYSMOD. A source ID specified on a ++ASSIGN statement is added to any source ID that is assigned to a specified SYSMOD by the RECEIVE command. It is also added to any source IDs currently associated with a specified SYSMOD that has already been received.

Processing ++PRODUCT and ++FEATURE statements

++PRODUCT and ++FEATURE MCS are processed whenever the RECEIVE command indicates that SYSMODs should be processed. If only HOLDDATA is being processed, ++PRODUCT and ++FEATURE MCS are not processed.

For each ++PRODUCT statement successfully received, SMP/E creates a PRODUCT entry in the global zone.

For each ++FEATURE statement successfully received, SMP/E creates a FEATURE entry in the global zone. SMP/E associates the FEATURE with any SYSMODs specified on the FMID operand of the ++FEATURE statement. If the same SYSMOD is specified on more than one ++FEATURE statement, all of the FEATUREs that specify the SYSMOD are associated with the SYSMOD. Specifically:

- If a SYSMOD exists in the global zone, its entry is updated to contain a FEATURE subentry for each FEATURE with which it is to be associated.
- If a SYSMOD does not exist in the global zone, SMP/E creates a SYSMOD entry in the global zone that contains only FEATURE subentries. The SYSMOD entry contains a FEATURE subentry for each FEATURE with which it is to be associated.

Processing ++HOLD and ++RELEASE statements

All modification control statements from SMPHOLD are processed in the order in which they occur. (++HOLD contained within SYSMODs that were received are processed when those SYSMODs are processed.) For each ++HOLD and ++RELEASE statement that was selected, SMP/E takes these actions:

- For ++HOLD statements, SMP/E adds the reason IDs to the associated global zone SYSMOD entry. If no SYSMOD entry exists, SMP/E builds one that contains just the reason IDs. SMP/E also saves the ++HOLD statement in the global zone. This is called a *HOLDDATA entry*.

  Note: For a given SYSMOD, SMP/E does not save multiple entries or subentries for a given reason ID. This is true except for ++HOLDs that are contained within a SYSMOD. For ++HOLDs that are contained

within a SYSMOD, unique HOLDDATA can be created for the same reason ID as long as the SYSMOD ID specified on each ++HOLD is different.

Therefore, uniqueness for a HOLDDATA entry is determined by reason ID and SYSMOD ID specified on the HOLD. For example, assume SMP/E has already received the following ++HOLD statement:

```
++HOLD (UZ12345) FMID(FXY1040) SYSTEM REASON(DOC)
        COMMENT(message XXX123 was changed. enter U to reply.).
```

Later, SMP/E receives another ++HOLD statement for the same SYSMOD, HOLD type, and reason ID, but the comment is different:

```
++HOLD (UZ12345) FMID(FXY1040) SYSTEM REASON(DOC)
        COMMENT(default for xyz command changed to NO.).
```

Information from the second ++HOLD statement replaces the information from the first ++HOLD statement.

- For ++HOLD statements with a hold category of FIXCAT, SMP/E translates fix category values into SOURCEIDs and assigns them to the resolving (fixing) PTFs identified on the HOLDDATA, if those PTFs are already received. A PTF must have been previously received, either during a previous RECEIVE command or the current RECEIVE, in order for the FIXCAT SOURCEID to be assigned.
- For ++RELEASE statements, SMP/E removes the specified reason ID from the global zone SYSMOD entry and deletes the ++HOLD statement for that reason ID from the global zone.

Compaction of inline data

The RECEIVE command automatically compacts inline data within SYSMODs when copying members to an SMPPTS data set whenever the COMPACT subentry in the active OPTIONS entry indicates compaction is to be performed (this is the default) and the driving system supports compression and expansion services. Otherwise, the RECEIVE command does not compact SMPPTS members.

When SMPPTS members are compacted, the modification control statements, inline JCLIN data, and inline ZAP data remain in their original, uncompacted state. All other inline data associated with each of the following modification control statements are compacted before being written to the SMPPTS data set member:

All data elements
All hierarchical file system elements
++MAC
++MACUPD
++MOD
++SRC
++SRCUPD

Processing for RECEIVE FROMNETWORK

When processing a RECEIVE FROMNETWORK command to receive a package from a network location, SMP/E performs the following steps:

1. SMP/E reads the information in the SERVER data set and, optionally, the CLIENT data set. The SERVER data set contains information about the FTP or HTTP(S) server on which resides the package to be received, as well as information about the package itself, such as its location on the server, an SHA-1 hash value for it, and an identifier for the package. The CLIENT data set contains information such as identifying which download mechanism to use, local fire wall navigation details, whether file transfer operations should be retried if necessary, and whether debugging information should be generated (see “Defining data sets and files for RECEIVE FROMNETWORK and RECEIVE ORDER processing” on page 246 for more information on the SERVER and CLIENT data sets).

2. Using the information found in the SERVER and CLIENT data sets, SMP/E spawns the appropriate client program based on the specified download method (FTP or HTTP(S)) and logs in to the FTP or HTTP(S) server that contains the package to be received.
3. SMP/E downloads the package attribute file for the package and stores it in the package directory of the SMPNTS. The package identifier value found in the SERVER data set is used as the package directory in the SMPNTS. If the `signaturekeyring` attribute is specified in the CLIENT data set, then SMP/E will validate the digital signature for a signed package. In this case the package attribute file to download is the GIMPAF2.XML file. Otherwise the GIMPAF.XML file is downloaded.
4. If the package is not digitally signed, or SMP/E is not validating the signature for a signed package, then SMP/E computes the SHA-1 hash value for the transferred file. If the computed hash value matches the hash value found in the SERVER data set information, then the file was transferred properly. If the hash values do not match, then the file may have been corrupted during its transfer (another possibility is that the value in the SERVER data set is incorrect). If a nonzero RETRY value was specified in the CLIENT data set information, SMP/E retries the transfer operation. Otherwise, RECEIVE processing will fail. See "Restarting RECEIVE FROMNETWORK" on page 245 for information on restarting after a failure.

   If the package is digitally signed and SMP/E is validating the signature, then SMP/E verifies the package signer is known and trusted by validating the signing certificate from the GIMPAF2.XML file against trusted Certificate Authority root certificates in the truststore identified by the signaturekeyring attribute in the CLIENT data set. If the signing certificate is trusted, the public key contained in the signing certificate is used to verify the signature for the GIMPAF2.XML file.

5. The package attribute file (either GIMPAF.XML or GIMPAF2.XML) can be thought of as a "packing list" for the package to be received. It contains a list of all the files that, combined, make up the entire package. SMP/E reads the package attribute file (if successfully stored) to determine the files that make up the package. SMP/E then transfers each file defined in the package attribute file from the FTP or HTTP(S) server and stores it in the package directory of the SMPNTS.

   SMP/E creates subdirectories as needed in the package directory and stores files in the subdirectories according to their `filetype` or `subdir` attributes indicated in the package attribute file. SMPPTFIN archive files are stored in the /SMPPTFIN subdirectory, SMPHOLD archive files are stored in the /SMPHOLD subdirectory, and SMPRELF archive files are stored in the /SMPRELF subdirectory. All other files are stored in the subdirectory specified on the `subdir` attribute or, if the `subdir` attribute is not present, in the package directory with no subdirectory.
6. SMP/E computes either a SHA-1 or SHA-256 hash value for each file it stores. If the computed hash value matches the known hash value indicated in the package attribute file, the file was transferred properly. If the hash values do not match, then the file has been corrupted during its transfer. If a nonzero RETRY value was specified in the CLIENT data set information, SMP/E retries the transfer operation. Otherwise, RECEIVE processing will fail. See "Restarting RECEIVE FROMNETWORK" on page 245 for information on restarting after a failure.
7. SMP/E closes the connection with the FTP or HTTP(S) server after all files of the package have been transferred successfully or when no further retry operations will be performed.
8. SMP/E processes the package contents (unless TRANSFERONLY has been specified on the RECEIVE command) as directed by the applicable RECEIVE command operands. If SYSMODs or HOLDDATA are to be processed, SMP/E transforms the archive files in the /SMPPTFIN, /SMPHOLD, and /SMPRELF subdirectories of the package directory into images of the original data. SMP/E then processes this data as if it had been read from the SMPPTFIN and SMPHOLD data sets, as described in "Processing input from SMPPTFIN and SMPHOLD" on page 264.

   If more than one archive file exists in the /SMPPTFIN or /SMPHOLD subdirectories, SMP/E reads and processes the archives in the sequence in which they are defined in the GIMPAF.XML and GIMPAF2.XML files. The GIMPAF.XML and GIMPAF2.XML files are created by the GIMZIP service routine, and the archives are defined in the those files in the order in which they are specified by the input ARCHDEF tags.

Processing for RECEIVE FROMNTS

Use the RECEIVE FROMNTS command to process a package that already exists in the SMPNTS directory. Such a package may have been downloaded previously using either the RECEIVE FROMNETWORK or RECEIVE ORDER commands with the TRANSFERONLY operand. If RECEIVE ORDER was used to create and download the package, then you can use the ORDER entry name to access the package. Otherwise, you must use the package directory name itself. If you specify an ORDER entry name, SMP/E reads the PKGID subentry in the ORDER entry to obtain the package directory value.

Once the package directory is known, SMP/E processes the package contents, as directed by the applicable RECEIVE command operands. If SYSMODs or HOLDDATA are to be processed, then SMP/E transforms the archive files in the /SMPPTFIN, /SMPHOLD, and /SMPRELF subdirectories of the package directory into images of the original data. SMP/E then processes this data as if it had been read from the SMPPTFIN and SMPHOLD data sets, as described in "Processing input from SMPPTFIN and SMPHOLD" on page 264.

In addition, if the package was identified using an ORDER entry name, as SYSMODs are stored in the global zone a source ID value corresponding to the ORDER entry name will be added to each SYSMOD entry. This source ID will be added in addition to a source ID value that may be specified on the SOURCEID operand of the RECEIVE command, and in addition to any source ID values found on ++ASSIGN statements being processed.

Processing for RECEIVE ORDER

RECEIVE ORDER processing includes these steps:

- ORDER request
- Query ORDER status
- Network transfer
- RECEIVE processing
- Termination

ORDER request

The ORDER Request phase supports the RECEIVE ORDER command. In this phase of processing, SMP/E deletes expired ORDER entries from the global zone, and then either requests a new ORDER or Processes a PENDING ORDER.

Deleting expired ORDER entries

SMP/E queries each existing ORDER entry in the global zone to determine if there are any expired ORDER entries. If there is an ORDER RETENTION subentry value in the active OPTIONS entry, then SMP/E uses that value for comparisons. Otherwise SMP/E uses the default ORDER RETENTION value of 180 days for comparisons.

SMP/E deletes an ORDER entry from the global zone if either of the following conditions occurs:

- The ORDER entry has a status of DOWNLOADED, and the difference between the current date and the ORDER entry's DOWNLDATE subentry value is greater than the ORDER RETENTION value, or
- The ORDER entry has a status of ERROR or PENDING and the difference between the current date and the ORDER entry's ORDERDATE subentry value is greater than the ORDER RETENTION value.

When an ORDER entry is deleted from the global zone, SMP/E also deletes the order package stored in the SMPNTS.

Requesting a new ORDER

If the PENDING operand was not specified on the RECEIVE ORDER command, this implies a new HOLDDATA or PTF order request is to be created and sent to the IBM Server.

Creating a new ORDER involves the following steps:

1. If CONTENT(HOLDDATA) is not specified on the RECEIVE ORDER command, then SMP/E constructs a software inventory (bitmap) and provides it to the IBM Server so an order can be tailored to the specific SMP/E environment. The FORTGTZONES operand on the RECEIVE ORDER command allows a user to specify the target zones used to define the scope of that software inventory. If FORTGTZONES is not specified, SMP/E uses all target zones defined by a ZONEINDEX subentry in the global zone as the target zone list. The server does not currently send back superseded PTFs, if the superseding PTF is already on the system as indicated in the bitmap.

   SMP/E uses the target zone list to generate the software inventory. The format of the software inventory is the same as that generated by the service routine GIMXSID except that the Feature records are not included in the inventory file.

   SMP/E writes the software inventory to a file within a temporary work directory in the UNIX file system. The naming convention for this work directory is `/smpwkdir/smpedate_tod/` where smpwkdir is either the directory allocated to SMPWKDIR or the system `/tmp` directory, and `date_tod` is a unique date and time-of-day value of the form YYYYDDDHHMMSSthmiju.
2. SMP/E sends the request to the IBM server using the information specified in the ORDERSERVER data set. For PTF orders, the request includes the software inventory produced in the previous step. Communication with the server is performed using the HTTP protocol and SSL (Secure Sockets Layer), also referred to as HTTPS.
3. If the IBM server accepts the request, it returns a unique order identifier to SMP/E. SMP/E then creates an ORDER entry in the global zone to describe this order. SMP/E generates the name for the ORDER entry in the form ORD*nnnnn*, where *nnnnn* is a decimal number with leading zeros from 00001 to 99999. The unique order identifier returned by the server becomes the ORDERID subentry of the ORDER entry.

Processing pending ORDERs

If the RECEIVE ORDER command specified the PENDING operand, this instructs SMP/E to process an existing, pending, order. A pending order is one that has been sent to the IBM Server but whose package has not yet been downloaded.

SMP/E searches the global zone for the ORDER entry specified on the PENDING operand. If SMP/E finds the ORDER entry in the global zone and it has a status of PENDING, SMP/E uses the server information saved in the ORDERSERVER subentry of the ORDER entry to communicate with the server regarding the PENDING order.

Query ORDER status

During this phase of RECEIVE ORDER processing, SMP/E sends requests for order status information to the server and waits for the order's package to be ready for download. The server responds differently depending on the status of the order:

The order's package is ready for download.
The server responds with the information required to download the order's package from an FTP or HTTP(S) server.

The order's package is not yet ready for download.
The server provides an estimate for the amount of time it will take to fulfill the order. SMP/E will then either wait to recheck the status of the order or stop processing.

No package resulted for this order.
No PTFs satisfied the selection criteria specified for this order. Therefore, no PTF package is produced. However, to ensure the most up to date HOLDDATA is obtained, SMP/E will modify and resend the request to the IBM server to request only HOLDDATA and no PTFs.

There is an error condition regarding the order.
The server provides error information that SMP/E interprets. Such error conditions cause the RECEIVE command to fail.

Network transfer

During the Network Transfer phase of the RECEIVE command, package files are downloaded and stored in the SMPNTS directory. Each package is stored in a unique package directory within the SMPNTS directory. For packages to be downloaded for the RECEIVE ORDER command, SMP/E creates a package directory in the SMPNTS with a name based on the ORDER entry name and current date and time of the form **ORD***nnnnn-DayMonthYear-HH.MM.SS.thm*. For example, `ORD00035-29January2006-15.02.47.496`.

After the package directory has been created, SMP/E uses the information provided by the order server to spawn the appropriate client program based on the specified download method (FTP or HTTP(S)) and download the package files. For the most part the FTP or HTTP(S) processing of the package files for RECEIVE ORDER is the same as that described in "Processing for RECEIVE FROMNETWORK" on page 270.

After the package files have been downloaded, the ORDER entry in the global zone is updated with the latest status of the order, DOWNLOADED. In addition, SMP/E sends a request to the server to close processing for this order. This enables the server to cleanup and complete processing for the order.

RECEIVE processing

SMP/E processes the contents of the order package (unless TRANSFERONLY has been specified on the RECEIVE command) as directed by the applicable RECEIVE command operands. If SYSMODs or HOLDDATA are to be processed, SMP/E transforms the archive files in the /SMPPTFIN, /SMPHOLD, and /SMPRELF subdirectories of the package into images of the original data. SMP/E then processes this data as if it had been read from the SMPPTFIN and SMPHOLD data sets, as described in "Processing input from SMPPTFIN and SMPHOLD" on page 264.

In addition, as SYSMODs are stored in the global zone, a source ID value corresponding to the ORDER entry name is added to each SYSMOD entry. This source ID is added in addition to a source ID value that may be specified on the SOURCEID operand of the RECEIVE command, and in addition to any source ID values found on ++ASSIGN statements being processed.

Zone and data set sharing considerations

The following indicates the phases of RECEIVE processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, "Sharing SMP/E data sets," on page 543.

1. Initialization

   Global zone
   Read without enqueue.

2. ORDER initialization

   Global zone
   Read with shared enqueue.

   Target zone
   Read with shared enqueue.

3. Build software inventory

   Global zone
   Read with shared enqueue.

   Target zone
   Read with shared enqueue.

4. Order request

   Global zone
   Update with exclusive enqueue.

5. Query order status

Global zone
Read with shared enqueue.

6. Network transfer

Global zone
Update with exclusive enqueue (for ORDER processing).

Read with shared enqueue (for FROMNETWORK processing).

Package id value
Exclusive enqueue.

7. RECEIVE processing

Global zone
Update with exclusive enqueue.

Target zones
Read with shared enqueue

Distribution zones
Read with shared enqueue

SMPPTS
Update with exclusive enqueue.

8. Termination

All resources are freed.

Chapter 15. The REJECT command

With the REJECT command, you can clean up the global zone, SMPPTS, and associated entries and data sets. REJECT is helpful if the SMPPTS is being used as a permanent database for all SYSMODs, including those that have been installed. You can also use it to purge old data from the global zone and SMPPTS.

To use the REJECT command, you must first determine which processing mode of the command you want to use. The mode you choose depends on the data you want to delete. (REJECT command processing modes are mutually exclusive. No two modes can be used together.) These are the modes of REJECT processing:

- **Mass mode:** SMP/E rejects all SYSMODs that have been received but not installed. Generally, SMP/E rejects these SYSMODs only if they have been neither accepted nor applied. However, you can specify operands to prevent SMP/E from checking where the SYSMODs have been installed.
- **Select mode:** SMP/E rejects specific SYSMODs that have been received but not installed. Generally, SMP/E rejects these SYSMODs only if they have been neither accepted nor applied. However, you can specify operands to prevent SMP/E from checking where the SYSMODs have been installed.
- **PURGE mode:** SMP/E rejects all SYSMODs that have been accepted into the specified distribution zones. PURGE mode can be used when SYSMODs were not automatically deleted once they were accepted. This is the case if NOPURGE was coded in the OPTIONS entry used to process the distribution zone.
- **NOFMID mode:** SMP/E rejects all SYSMODs applicable to functions that are not part of the system. NOFMID mode can be used to delete service for all functions that have been deleted from the global zone. NOFMID mode can also be used to delete FEATURE and PRODUCT entries for all functions that have been deleted from the global zone.

For each eligible SYSMOD, SMP/E deletes the following, regardless of the processing mode:

- The SMPPTS MCS entry
- The global zone SYSMOD entry
- The associated FMID subentry in the GLOBALZONE entry, as appropriate (see “Processing the SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA” on page 291 for details)
- The associated SMPTLIB data sets, if the SYSMOD was packaged in RELFILE format (see “Processing the SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA” on page 291 for details)

Eligible HOLDDATA entries are deleted only during Select mode and NOFMID mode processing, and only if the HOLDDATA operand is specified. During Mass mode and Purge mode processing, and when HOLDDATA is not specified for Select mode and NOFMID mode processing, all HOLDDATA entries are retained including internal SYSTEM HOLDDATA entries. (See “Selecting the eligible SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA” on page 288 for details.)

Zones for SET BOUNDARY

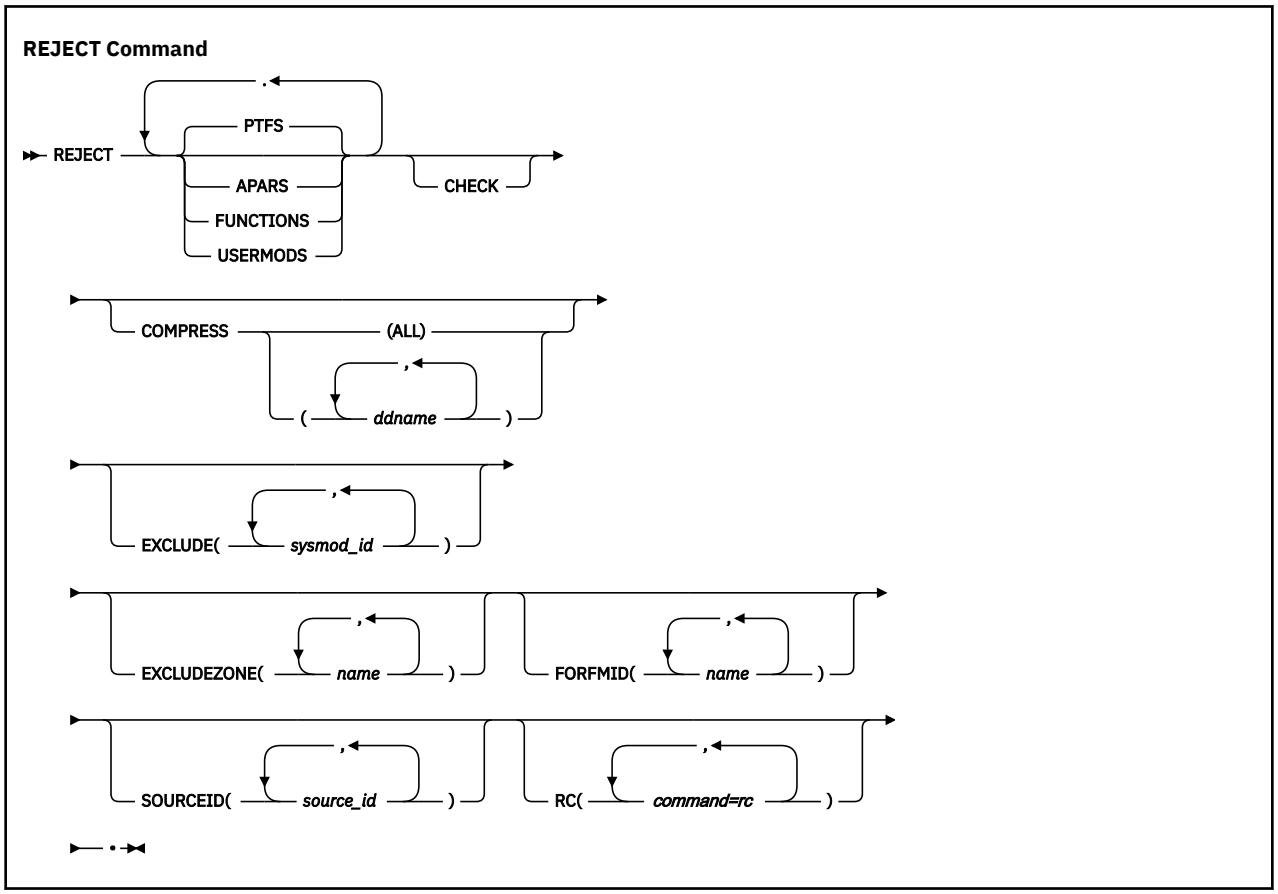
For the REJECT command, the SET BOUNDARY command must specify the global zone.

Syntax

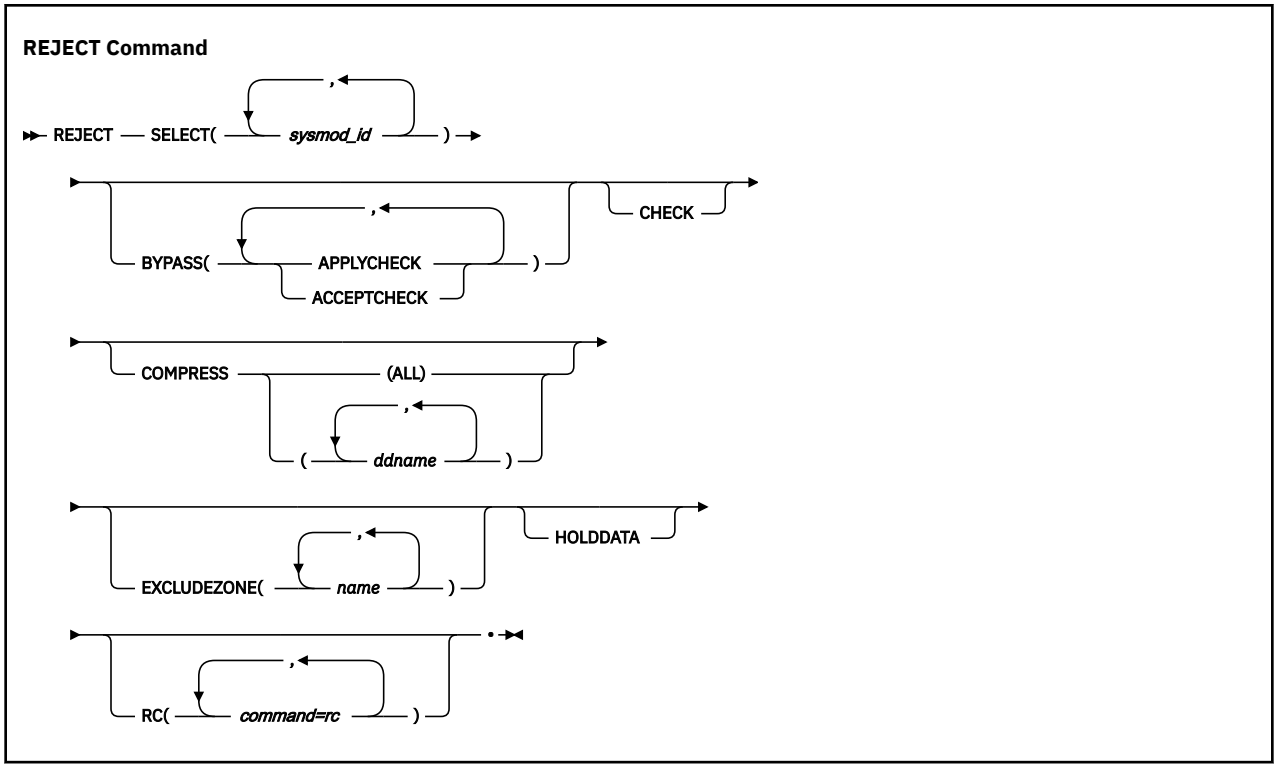
This section shows the syntax for the modes of REJECT processing:

- Mass mode
- Select mode
- PURGE mode
- NOFMID mode

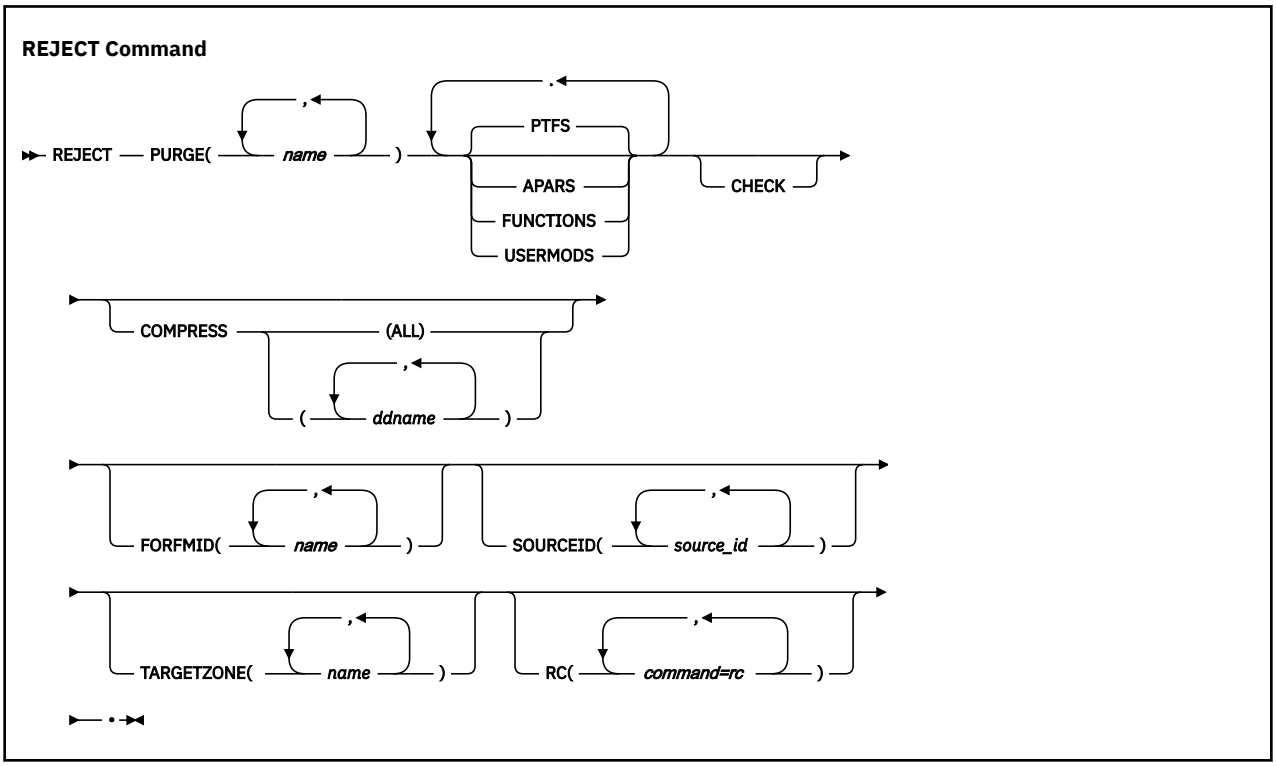
Mass mode syntax



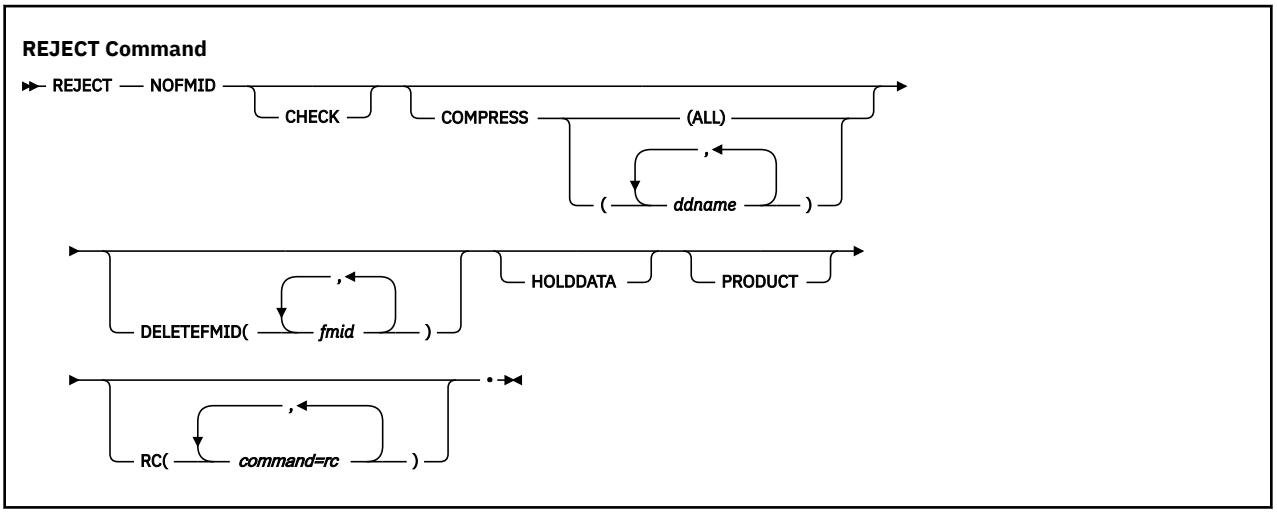
Select mode syntax



PURGE mode syntax



NOFMID mode syntax



Operands

APARS
indicates that APARs should be rejected.

Note:

1. APARS is allowed only in mass mode and PURGE mode.
2. APARS can also be specified as APAR.

BYPASS
indicates that SMP/E should reject SYSMODs that have been installed.

APPLYCHECK
indicates that selected SYSMODs should be rejected, even if they have been applied.

ACCEPTCHECK
indicates that selected SYSMODs can be rejected, even if they have been accepted.

Note:

1. BYPASS is allowed only in select mode.
2. APPLYCHECK can also be specified as APPCHK.
3. ACCEPTCHECK can also be specified as ACCCHK.
4. To reject a superseded SYSMOD, you must either specify the appropriate BYPASS operands, or you must use the EXCLUDEZONE operand to specify the zones in which the SYSMOD is superseded.

CHECK
CHECK indicates that SMP/E should not actually update any libraries. Rather it should just take these actions:

- Test for errors other than those that could occur when the libraries are actually updated.
- Produce a REJECT Summary Report indicating what would have happened if CHECK had not been specified.

COMPRESS
indicates which partitioned data sets should be compressed.

- If you specify ALL, any partitioned data sets that were updated are compressed. In addition, the SMPPTS data set is compressed regardless of whether it was updated.
- If you specify one or more specific ddnames, only the data sets they apply to are compressed. Those data sets are compressed regardless of whether they were updated.

Note:

1. COMPRESS is allowed in all modes of REJECT processing.
2. COMPRESS can also be specified as C.

DELETEFMID
specifies one or more FMID subentries that are to be deleted from the GLOBALZONE entry before SMP/E selects the SYSMODs or HOLDDATA to be rejected.

Note:

1. DELETEFMID is allowed only in NOFMID mode.
2. DELETEFMID can also be specified as DFMID.
3. DELETEFMID does **not** cause SMP/E to reject the function SYSMODs associated with the specified FMID values.

EXCLUDE
specifies one or more SYSMODs that should not be rejected.

Note:

1. EXCLUDE is allowed only in mass mode.
2. EXCLUDE can also be specified as E.

EXCLUDEZONE
indicates that SMP/E should not check whether SYSMODs have been installed in the specified zones or ZONESETs.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named SYS1 and a zone named SYS1. If you specify SYS1 on this operand, SMP/E assumes you want to use the zones defined in ZONESET SYS1 (which might or might not include zone SYS1), and not the individual zone SYS1.

Note:

1. EXCLUDEZONE is allowed only in mass mode and select mode.
2. EXCLUDEZONE can also be specified as EZONE.
3. EXCLUDEZONE cannot specify all the zones defined by ZONEINDEX subentries. If you are doing select-mode processing and you do not want SMP/E to check any zones, you can specify BYPASS(APPLYCHECK,ACCEPTCHECK). If you are doing mass-mode processing, there is no way to have SMP/E ignore all the zones.
4. You should be careful when using the EXCLUDEZONE operand. A SYSMOD that is installed in one of the excluded zones may be rejected, even if you were later going to install it in another zone. Because the rejected SYSMOD is no longer in the SMPPTS, it cannot be installed in any more zones.
5. To reject a superseded SYSMOD, you must either specify the appropriate BYPASS operands, or you must use the EXCLUDEZONE operand to specify the zones in which the SYSMOD is superseded.

FORFMID
indicates that only SYSMODs and HOLDDATA for the specified FMIDs or FMIDSETs should be rejected.

Note:

1. FORFMID is allowed only in mass mode and PURGE mode.
2. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.

FUNCTIONS
indicates that functions should be rejected.

Note:

1. FUNCTIONS is allowed only in mass mode and PURGE mode.
2. FUNCTIONS can also be specified as FUNCTION.

HOLDDATA
indicates that SMP/E should reject HOLDDATA entries. There are two types of HOLDDATA entries: those that have an associated SYSMOD entry, and those that have no associated SYSMOD entry. (For more information, see *z/OS SMP/E Reference* .) How SMP/E processes HOLDDATA entries depends on the mode of REJECT processing you choose.

Note: HOLDDATA is allowed for all modes of REJECT processing, but it will be ignored during Mass mode and Purge mode processing. Eligible HOLDDATA entries are deleted only during REJECT Select mode and REJECT NOFMID mode processing when the HOLDDATA operand is specified.

- **Mass mode:** The HOLDDATA operand is ignored when specified for Mass mode REJECT processing. All HOLDDATA entries are retained during Mass mode processing. This includes internal SYSTEM HOLDs even though the containing SYSMOD may be deleted.
- **Select mode:** If you specify HOLDDATA, SMP/E deletes HOLDDATA entries associated with the eligible SYSMOD IDs, regardless of whether an associated SYSMOD entry exists.

  If you do not specify HOLDDATA, SMP/E does not delete any HOLDDATA entries, including internal SYSTEM HOLDDATA entries, even though the containing SYSMOD may be deleted.
- **PURGE mode:** The HOLDDATA operand is ignored when specified for Purge mode REJECT processing. All HOLDDATA entries are retained during Purge mode processing. This includes internal SYSTEM HOLDs even though the containing SYSMOD may be deleted.
- **NOFMID mode:** If you specify HOLDDATA, SMP/E deletes HOLDDATA entries associated with the SYSMOD entries that are also being rejected. In addition, SMP/E deletes HOLDDATA entries whose associated FMID is not defined in the global zone.

  If you do not specify HOLDDATA, SMP/E does not delete any HOLDDATA entries, including internal SYSTEM HOLDDATA entries, even though the containing SYSMOD may be deleted.

NOFMID
indicates that SMP/E is to reject SYSMODs applicable to functions that are not part of the system. (The FMID they apply to is not in the GLOBALZONE entry.)

If DELETEFMID is also specified, SMP/E deletes the specified FMIDs from the GLOBALZONE entry before determining which SYSMODs and HOLDDATA to delete.

Note: NOFMID is allowed only in NOFMID mode.

PRODUCT
indicates that SMP/E should reject PRODUCT and FEATURE entries.

Note: PRODUCT is allowed only in NOFMID mode.

PTFS
indicates that PTFs should be rejected. This is the default SYSMOD type operand. In mass or PURGE mode processing, if no SYSMOD type is specified, only PTFs are rejected.

Note:

1. PTFS is allowed only in mass mode and PURGE mode.
2. PTFS can also be specified as PTF.

PURGE
indicates that SMP/E should reject only SYSMOD entries for SYSMODs that have been accepted into the specified distribution zones or ZONESETs.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named SYS1 and a zone named SYS1. If you specify SYS1 on this operand, SMP/E assumes you want to use the zones defined in ZONESET SYS1 (which might or might not include zone SYS1), not the individual zone SYS1.

Any individual zones you specify must be distribution zones. Among all the zones and ZONESETs you specify, there must be at least one distribution zone.

- If you specify a single zone, SMP/E rejects entries only for SYSMODs that have been accepted into that zone.
- If you specify more than one zone (including zones in a ZONESET), SMP/E rejects entries only for SYSMODs that have been installed in at least one distribution zone and have been installed in all the distribution zones they apply to.

If TARGETZONE is also specified, SMP/E rejects entries only for SYSMODs that have also been installed in the specified target zones where they are applicable.

HOLDDATA entries are not deleted during Purge mode processing. The HOLDDATA operand is ignored when specified with the PURGE operand.

Note:

1. PURGE is only allowed in PURGE mode.
2. PURGE cannot be used to reject **specific** SYSMODs that have been accepted. To do this you must specify BYPASS(ACCCHK) in select mode.
3. PURGE uses a SYSMOD's REWORK value to determine whether the SYSMOD should be rejected. If the REWORK value of a SYSMOD in the global zone is higher than the REWORK value of the SYSMOD in the distribution or target zone, then the SYSMOD is **not** rejected.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the REJECT command.

Before SMP/E processes the REJECT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the REJECT command. Otherwise, the REJECT command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand **must be the last** operand specified on the command.
2. RC is allowed in all modes of REJECT processing.
3. If you do specify the RC operand, return codes for commands not specified do not affect processing for the REJECT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

SELECT
specifies one or more SYSMODs that should be rejected.

Note:

1. SELECT is only allowed in select mode.
2. SELECT can also be specified as S.

SOURCEID
indicates that only entries for SYSMODs associated with the specified source IDs should be rejected.

Note:

1. SOURCEID is allowed only in mass mode and PURGE mode.
2. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID value (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are selected.
   - Implicitly, by partially specifying a source ID using asterisks (*) as global characters and percent signs (%) as placeholders.
     – A single asterisk, for example, RSU*, *0711, or RSU*1, indicates that zero or more characters can occupy that position.
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU are selected.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are selected.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are selected.
     – A single percent sign, for example, RSU0%11, indicates that any one single character can occupy that position. In this case, SYSMODS that contain any of the following source IDs are selected: RSU0711, RSU0211, and RSU0311. SYSMODs that contain RSU00711 are not selected.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source ID specifications:

   ```
   RSU0709
   RSU*
   IBM.Device.20%4
   IBM.Device.*.zAAP
   ```

3. A given source ID can be explicitly specified **only once** on the SOURCEID operand.
4. If no SYSMOD types are specified, only PTFs are processed. To process other types of SYSMODs, you must specify the desired SYSMOD types.
5. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

TARGETZONE
indicates that SMP/E should only reject SYSMOD entries for SYSMODs that have been applied to the specified target zones or ZONESETs.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named SYS1 and a zone named SYS1. If you specify SYS1 on this operand, SMP/E assumes you want to use the zones defined in ZONESET SYS1 (which might or might not include zone SYS1), not the individual zone SYS1.

Any individual zones you specify must be target zones. Among all the zones and ZONESETs you specify, there must be at least one target zone.

SMP/E rejects entries for SYSMODs that have been installed in the specified target zones where they are applicable. If a SYSMOD is not applicable to any of the specified target zones, it may still be rejected if it is installed in the specified distribution zones where it is applicable.

Note:

1. TARGETZONE is allowed only in PURGE mode.
2. TARGETZONE cannot be used to reject **specific** SYSMODs that have been applied. To do this, you must specify BYPASS(APPCHK) in select mode.

USERMODS
indicates that USERMODs should be rejected.

Note:

1. USERMODS is allowed only in mass mode and PURGE mode.
2. USERMODS can also be specified as USERMOD.

Data sets used

The following data sets might be needed to run the REJECT command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPRPT	SYSPRINT
SMPCSI	SMPOUT	SMPSNAP	SYSUT1
SMPLOG	SMPPTS	SMPTLIB	*zone*

Note: *zone* represents the DD statements required for each distribution zone used by this command. If the DD statements are not specified, the ZONEINDEX information in the GLOBALZONE entry is used to dynamically allocate the data sets. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

zone is required if the PURGE operand is specified.

Output

Output from the REJECT command includes reports, as well as statistics written to SMPOUT and SMPLOG.

Reports

Two reports are produced during REJECT processing:

- File Allocation report
- REJECT Summary report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

Statistics

SMP/E writes statistics to SMPOUT and SMPLOG to summarize what happened during REJECT processing. These statistics follow the message issued at the completion of REJECT processing. Figure 9 on page 285 shows the format of the statistics.

```
REJECT STATISTICS

  SYSMODS REJECTED         - nnnnnn  SYSMODS NOT REJECTED - nnnnnn
  FMIDS DELETED            - nnnnnn  FMIDS NOT DELETED    - nnnnnn
  HOLDDATA DELETED         - nnnnnn
  FEATURE ENTRIES REJECTED - nnnnnn
  PRODUCT ENTRIES REJECTED - nnnnnn
```

Figure 9. REJECT Statistics

SYSMODS REJECTED
is the number of SYSMODs that were rejected.

SYSMODS NOT REJECTED
is the number of SYSMODs that were candidates but were not rejected. The reason appears in the REJECT Summary report.

FMIDS DELETED
is the number of FMIDs that were deleted. This includes FMIDs specified on the DELETEFMID operand in NOFMID mode or FMIDs that were deleted from the GLOBALZONE entry in other modes when functions were rejected.

FMIDS NOT DELETED
is the number of FMIDs specified on the DELETEFMID operand that were not deleted.

HOLDDATA DELETED
is the number of HOLDDATA entries that were deleted.

FEATURE ENTRIES REJECTED
is the number of FEATURE entries that were rejected.

PRODUCT ENTRIES REJECTED
is the number of PRODUCT entries that were rejected.

Examples

The following examples are provided to help you use the REJECT command.

Example 1: Rejecting all SYSMODs that have not been installed (mass mode)

Assume that you want to delete all SYSMODs that have been received but not installed. Also assume that you want to CHECK the results of the REJECT command before proceeding with it. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   CHECK              /* Check results only        */
         APARS              /* Reject all received-only */
         FUNCTIONS          /* SYSMODs.                 */
         PTFS
         USERMODS.
```

Note: Be careful when you use this format for REJECT. It may reject SYSMODs you wanted to keep, and you would have to receive them again. Use the CHECK operand to verify that the REJECT command will do what you expect, then re-run the command after removing the CHECK operand.

If you want to reject only PTFs that have been received but not installed, you can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT                      /* Reject all received-only
                               PTFs.                    */.
```

If no SYSMOD types are specified on a REJECT command for mass mode, SMP/E rejects only PTFs.

Example 2: Rejecting all SYSMODs for a specific function (mass mode)

Assume that you have received a new function (HMX1100), as well as service for that function. You have now decided to delete that function and all the associated service for it. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   APARS              /* Reject all APARs,        */
         FUNCTIONS          /* functions,               */
         PTFS               /* PTFs, and                */
         USERMODS           /* USERMODs for             */
         FORFMID(HMX1100)   /* HMX1100.                 */.
```

Example 3: Rejecting selected SYSMODs that have been applied (select mode)

Assume that you have applied a specific user modification but have not accepted it. You want to reject the current version, update the SYSMOD, and then reapply it. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   S(MYMOD01)         /* Reject this SYSMOD       */
         BYPASS(            /* even though it was       */
                APPCHK)     /* applied.                 */.
```

Example 4: Rejecting selected SYSMODs that have been accepted and applied (select mode)

Assume that you have applied a user modification and accepted it (with NOPURGE in the OPTIONS entry). You want to reject the current version, update the SYSMOD, and then reapply and reaccept it. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   S(MYMOD01)         /* Reject this SYSMOD       */
         BYPASS(            /* even though it was       */
                ACCEPTCHECK /* accepted and             */
                APPLYCHECK) /* applied.                 */.
```

Example 5: Rejecting HOLDDATA that has no SYSMOD entry (select mode)

Assume that you have received a ++HOLD statement for PTF UZ04356 from SMPHOLD, but you have not yet received the PTF itself. There is a HOLDDATA entry but no SYSMOD entry. If you do not plan to install that PTF, you may want to delete the HOLDDATA entry. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   S(UZ04356)         /* For this SYSMOD,         */
         HOLDDATA           /* reject the HOLDDATA.     */.
```

Note: If there had been a SYSMOD entry for UZ04356, these commands would have deleted the SYSMOD entry along with the HOLDDATA entry.

Example 6: Rejecting SYSMODs that have been accepted (PURGE mode)

Assume that you have been using your SMPPTS as a database for all SYSMODs (by using NOPURGE in the OPTIONS entry). You have been receiving service through ESO tapes, which assign the SYSMODs source IDs to identify the service levels you have installed. Now you want to purge PTFs from service levels

0707 through 0710 that have been accepted into distribution zone DLIB1. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   PURGE(DLIB1)       /* Reject SYSMODs installed
                               in this DLIB zone        */
         SOURCEID(PUT0707,  /* for these service levels.*/
                  PUT0708,  /*                          */
                  PUT0709,  /*                          */
                  PUT0710)  /*                          */.
```

Note:

1. Because no SYSMOD types were specified, only PTFs are rejected.
2. Without the SOURCEID operand, SMP/E rejects the SYSMOD entries for **all** the PTFs that had been accepted into DLIB1.
3. PURGE uses a SYSMOD's REWORK value to determine whether the SYSMOD should be rejected. If the REWORK value of a SYSMOD in the global zone is higher than the REWORK value of the SYSMOD in the distribution or target zone, then the SYSMOD is **not** rejected.

Example 7: Rejecting SYSMODs that have been accepted and applied (PURGE mode)

Assume that you have a system with three target zones (TMV1, TMV2, and TMV3) and two DLIB zones (DMVA and DMVB). You have set up two ZONESETs to make it easier to maintain these zones. MVSSET contains TMV1, TMV2, and DMVA, and MVSTEST contains TMV3 and DMVB.

Assume that you want to reject all the PTFs that were installed in the zones contained in the MVSSET ZONESET. You can use these commands:

```
SET    BDY(GLOBAL)        /* Process global zone.    */.
REJECT PURGE(MVSSET)      /* Reject PTFs in MVSSET   */
       TZONE(MVSSET)      /* DLIB and target zones.  */.
```

Example 8: Rejecting SYSMODs for undefined functions (NOFMID mode)

Assume that you had received, applied, and accepted function HMX1101. The function was automatically deleted from the global zone and SMPPTS when it was accepted. You have also received service for the function.

Assume that you have now decided to install an updated version of the function. To prepare for this, you want to delete the FMID of the current function from the GLOBALZONE entry, as well as delete the service and associated HOLDDATA that were received for that function. You can use these commands:

```
SET    BDY(GLOBAL)        /* Process global zone.    */.
REJECT DFMID(HMX1101)     /* Delete FMID and reject  */
       NOFMID HOLDDATA    /* for FMIDs not in GZONE. */.
```

Note:

1. This deletes SYSMODs and associated HOLDDATA for **all** functions that are not defined in the GLOBALZONE entry. It is not limited to entries for HMX1101. To limit the REJECT command to specific functions, use the FORFMID operand in another REJECT mode.
2. Rather than manually determine which FMIDs to delete, you can use the sample programs GIMCRSAM and GIMPRSAM (provided in SYS1.SAMPLIB) to help you create a REJECT NOFMID command for the FMIDs that are superseded or deleted in the DLIB zones you specify.

Example 9: Deleting service for a group of source IDs

Assume that you are maintaining two systems, and you want to delete from the global zone and SMPPTS all service from 2008 service levels already applied and accepted on both of your systems.

The distribution zones associated with these two systems are SYS1DLIB and SYS2DLIB. You can use the following commands:

```
SET      BDY(GLOBAL)        /* Process the global zone. */.
REJECT                      /* Reject from              */
         PURGE(SYS1DLIB     /* zones SYS1DLIB           */
               SYS2DLIB)    /* and SYS2DLIB             */
         PTFS               /* PTFs accepted            */
         SOURCEID(PUT07*)   /* from 2008 service levels.*/.
```

Example 10: Rejecting selected SYSMODs that have been superseded (select mode)

Assume that you have applied but not yet accepted PTF UZ45678, which supersedes a previous PTF, UZ01234. You had received the superseded PTF, but had never installed it. For cleanup purposes, you want to reject the superseded PTF. The simplest way to do this is by specifying the BYPASS(APPLYCHECK) operand. That way, you do not need to indicate the specific zones in which the SYSMOD is superseded:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   S(UZ01234)         /* Reject this SYSMOD, which*/
         BYPASS(            /* was superseded by a      */
                APPCHK)     /* SYSMOD that was applied. */.
```

Processing

REJECT processing includes these steps:

1. Selecting the eligible SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA
2. Deleting the entries and related data sets for the selected SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA

Selecting the eligible SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA

SMP/E checks the operands specified on the REJECT command to determine which SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA are eligible. First, SMP/E determines what type of REJECT processing was requested:

- If neither SELECT, PURGE, nor NOFMID was specified, it does mass-mode processing.
- If SELECT was specified, it does select-mode processing.
- If PURGE was specified, it does PURGE-mode processing.
- If NOFMID was specified, it does NOFMID-mode processing.

SMP/E selects the eligible SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA according to the mode of processing.

Mass-mode processing

In mass-mode processing, SMP/E selects only SYSMODs that have not been applied or accepted anywhere. First, SMP/E checks each SYSMOD to see if it meets the requirements defined by the operands specified on the REJECT command.

- If you specify the EXCLUDE operand, SMP/E makes sure the SYSMOD was not specified in the exclude list.
- If any SYSMOD types were specified (APARS, FUNCTIONS, PTFS, or USERMODS), SMP/E selects only SYSMODs that match one of the specified types. If no SYSMOD types were specified, only PTFs are selected.
- If you specify the FORFMID operand, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values contained in a specified FMIDSET.

- If you specify the SOURCEID operand, SMP/E makes sure one of the SOURCEIDs of the SYSMOD matches a source ID that you have specified, either explicitly or implicitly, on the SOURCEID operand.

Next, SMP/E determines which of these SYSMODs have not been applied or accepted anywhere. To do this, it checks all the target and distribution zones defined by zone index subentries, minus any zones or ZONESETs specified on the EXCLUDEZONE operand. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a SYSMOD is not installed in any of the zones that were checked, it may be rejected.

Note: A SYSMOD is considered installed if the ERROR indicator in its entry is off, or if it has been superseded. A deleted SYSMOD is not considered installed.

Each SYSMOD that meets **all** the specified conditions is eligible to be rejected.

Note: Superseded SYSMODs are rejected in mass mode only if the EXCLUDEZONE operand specifies the zone where the SYSMOD is superseded. EXCLUDEZONE cannot exclude all target and distribution zones from processing.

Deleted SYSMODs are rejected in mass mode provided they are eligible, regardless of whether EXCLUDEZONE is specified.

No HOLDDATA entries are deleted during Mass-mode processing. If the HOLDDATA operand is specified, it is ignored. When a SYSMOD entry is deleted, its internal SYSTEM HOLDDATA entries are not deleted, but are retained in the global zone.

Select-mode processing

In select-mode processing, SMP/E selects only SYSMODs that were specified on the SELECT operand. Generally, SMP/E chooses only SYSMODs that have not been applied or accepted anywhere. To determine this, it checks all the target and distribution zones defined by zone index subentries, minus any zones or ZONESETs specified on the EXCLUDEZONE operand. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a SYSMOD is not installed in any of the zones that were checked, it may be rejected.

Note: A SYSMOD is considered installed if the ERROR indicator in its entry is off, or if it has been superseded. A deleted SYSMOD is not considered installed.

However, if BYPASS was also specified, SMP/E can select SYSMODs that have been installed.

- If BYPASS(APPLYCHECK) was specified, SMP/E selects the specified SYSMODs, even if they have been applied, but not if they have been accepted.
- If BYPASS(ACCEPTCHECK) was specified, SMP/E selects the specified SYSMODs, even if they have been accepted, but not if they have been applied.
- If BYPASS(APPLYCHECK, ACCEPTCHECK) was specified, SMP/E selects the specified SYSMODs, even if they have been accepted, applied, or both.

Note: Superseded SYSMODs are rejected in select mode only in these cases:

- The appropriate BYPASS operand is specified.
- The EXCLUDEZONE operand specifies the zones where the SYSMODs are superseded. EXCLUDEZONE cannot exclude all target and distribution zones from processing.

Deleted SYSMODs are rejected in select mode provided they are eligible, regardless of whether BYPASS or EXCLUDEZONE is specified.

If HOLDDATA was specified, HOLDDATA entries associated with eligible SYSMOD IDs are also eligible to be rejected, regardless of whether an associated SYSMOD entry exists.

If HOLDDATA was not specified, no HOLDDATA entries are eligible to be rejected. When a SYSMOD entry is deleted, its internal SYSTEM HOLDDATA entries are not deleted, but are retained in the global zone.

PURGE-mode processing

In PURGE-mode processing, SMP/E selects only SYSMODs that have been installed in the specified distribution zones and target zones to which they are applicable.

First, SMP/E checks whether a SYSMOD type, FORFMID, or SOURCEID was specified. If so, SYSMODs must meet the requirements defined by those operands:

- If you specify one or more of the SYSMOD-type operands (that is, FUNCTIONS, PTFS, APARS, or USERMODS), SMP/E checks to make sure the SYSMOD type was one of those specified.

  If you do not specify a SYSMOD-type operand, the default is for SMP/E to process only PTF SYSMODs.
- If you specify the FORFMID operand, SMP/E makes sure that either the FMID value on one of the ++VER statements within the SYSMOD or the SYSMOD ID itself matches either an FMID specified in FORFMID or one of the FMID values contained in a specified FMIDSET.
- If you specify the SOURCEID operand, SMP/E makes sure one of the SOURCEIDs of the SYSMOD matches a source ID that you have specified, either explicitly or implicitly, on the SOURCEID operand.

A SYSMOD is eligible to be rejected if it meets all of these conditions:

- It is installed in at least one of the distribution zones specified on the PURGE operand.
- It is successfully installed, superseded, or deleted in all the specified distribution zones to which it is applicable.
- The REWORK value of the SYSMOD in the global zone is equal to or less than the REWORK value of the SYSMOD in the distribution and target zones.

To determine applicability, SMP/E checks the specified distribution zones. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a ZONESET was specified, SMP/E uses only the distribution zones defined in the ZONESET and ignores the target zones. If there are no distribution zones among all the zones and ZONESETs specified, REJECT processing fails. SMP/E determines whether a SYSMOD is applicable to a zone according to the type of SYSMOD.

- A base function is applicable to a zone if its SREL matches an SREL defined for the zone.
- Other types of SYSMODs are applicable to a zone if they meet either of the following sets of conditions:
  - The SREL matches an SREL defined for the zone and the SYSMOD is for an FMID that is installed in the zone.
  - The SREL matches an SREL defined for the zone and the SYSMOD is for an FMID that has been received but has not yet been installed in the zone.

If TARGETZONE was specified, the selected SYSMODs must also be installed in the specified target zones to which they are applicable. If a SYSMOD is not applicable to any of the specified target zones, it may still be eligible if it is installed in the specified distribution zones where it is applicable.

To determine applicability, SMP/E checks the specified target zones. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a ZONESET was specified, SMP/E uses only the target zones defined in the ZONESET and ignores the distribution zones. If there are no target zones among all the zones and ZONESETs specified, REJECT processing fails. (SMP/E determines applicability to target zones in the same way as it determines applicability to distribution zones.)

Each SYSMOD that meets all the specified conditions is eligible to be rejected.

No HOLDDATA entries are deleted during Purge-mode processing. If the HOLDDATA operand is specified, it is ignored. When a SYSMOD entry is deleted, its internal SYSTEM HOLDDATA entries are not deleted, but are retained in the global zone.

NOFMID-mode processing

In NOFMID-mode processing, SMP/E selects only SYSMODs that are applicable to FMIDs not defined in the GLOBALZONE entry. Before selecting the eligible SYSMODs, SMP/E first checks whether DELETEFMID

was specified. If so, it deletes the specified FMIDs from the GLOBALZONE entry. It then compares the SYSMOD entries in the global zone with the FMIDs in the GLOBALZONE entry.

- A base function is eligible to be rejected if its FMID does not match an FMID in the GLOBALZONE entry.
- A dependent function is eligible to be rejected if its FMID does not match an FMID in the GLOBALZONE entry and if either of these conditions is met:
  - The FMID specified on the ++VER statement does not match an FMID in the GLOBALZONE entry.
  - The FMIDs match, but the SREL specified on the ++VER statement does not match an SREL in the GLOBALZONE entry.

  If all the ++VER statements in the function meet these conditions, the function is eligible to be rejected.
- For other types of SYSMODs, SMP/E checks whether either of these conditions is met:
  - The FMID specified on the ++VER statement does not match an FMID in the GLOBALZONE entry.
  - The FMIDs match, but the SREL specified on the ++VER statement does not match an SREL in the GLOBALZONE entry.

  If all the ++VER statements in the SYSMOD meet these conditions, the SYSMOD is eligible to be rejected.

If HOLDDATA was specified, HOLDDATA entries associated with eligible SYSMOD entries are also eligible to be rejected. In addition, HOLDDATA entries whose associated FMID is not defined in the global zone are eligible for rejection.

If HOLDDATA was not specified, no HOLDDATA entries are eligible to be rejected. When a SYSMOD entry is deleted, its internal SYSTEM HOLDDATA entries are not deleted, but are retained in the global zone.

If **PRODUCT** was specified and NOFMID mode is in effect, FEATURE and PRODUCT entries are eligible to be rejected.

- A FEATURE entry is eligible to be rejected if no FMIDs associated with the FEATURE match any FMID in the GLOBALZONE entry. If even one FMID associated with the FEATURE exists in the GLOBALZONE FMID list, the FEATURE entry will not be rejected.

  Note: A FEATURE entry that does not contain an FMID subentry cannot be deleted with the REJECT command. The UCLIN command must be used instead.
- A PRODUCT entry is eligible to be rejected if no FEATURE entries associated with the PRODUCT are in the global zone. A PRODUCT is associated with a FEATURE when the FEATURE entry specifies that PRODUCT on the PRODUCT subentry within the FEATURE entry. If even one FEATURE associated with the PRODUCT exists in the global zone, the PRODUCT entry is not rejected.

If **PRODUCT** was not specified, or if any mode other than NOFMID is in effect, no PRODUCT or FEATURE entries are rejected.

The PRODUCT operand has no effect on which SYSMOD entries will be rejected. This includes SYSMODs that contain only FEATURE subentries.

Processing the SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA

Once SMP/E has selected all the eligible SYSMODs, FEATUREs, PRODUCTs, and HOLDDATA, it rejects the appropriate entries and associated SMPTLIB data sets. For each eligible SYSMOD, SMP/E deletes the following entries:

- The SMPPTS MCS entry.
- The global zone SYSMOD entry.
- The associated FMID subentry in the GLOBALZONE entry.

  The FMID subentry is deleted for a function SYSMOD if both of these conditions are met:
  - Mass-mode or select-mode processing was done.
  - BYPASS was not specified.

Note: Once the FMID subentry is deleted, service for that function is no longer received. If you intend to install a more recent copy of the function, you can use UCLIN to add the FMID subentry back to the GLOBALZONE entry. This way, no service is lost in the meantime.

- The eligible HOLDDATA entries.

- The associated SMPTLIB data sets, if the SYSMOD was packaged in RELFILE format. SMP/E determines the number of data sets to delete from the FILES operand of the header MCS.

  SMP/E locates the SMPTLIB data sets to be deleted by checking the following sources in the order shown. If SMP/E finds one of the data sets, it assumes that all of the SMPTLIB data sets can be found the same way.

  1. If the SMPTLIB data sets are cataloged, they are deleted according to the catalog.
  2. If there is an SMPTLIB DD statement, the data sets are deleted from the volumes specified on the DD statement.
  3. If there is an SMPTLIB DDDEF entry, the data sets are deleted from the volumes specified in the DDDEF entry.

  If the SMPTLIB data sets are not located by the catalog and there is no DD statement or DDDEF entry for them, the SYSMOD is not rejected. If there is a DD statement or DDDEF entry but the data sets are not found, SMP/E issues a warning message and continues REJECT processing for that SYSMOD.

  Note: If any elements were packaged in either LKLIB or TXLIB format, no action is taken on those data sets.

Zone and data set sharing considerations

The following identifies the phases of REJECT processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

2. REJECT processing

   Global zone
   Update with exclusive enqueue.

   SMPPTS
   Update with exclusive enqueue.

   DLIB zone
   Read with shared enqueue.

   Target zone
   Read with shared enqueue.

   Note:

   a. The distribution zones are accessed in mass mode, select mode, and PURGE mode.

   b. The target zones are accessed in mass mode and select mode, and in PURGE mode if the TARGETZONE operand was specified.

3. Termination

   All resources are freed.

Chapter 16. The REPORT CROSSZONE command

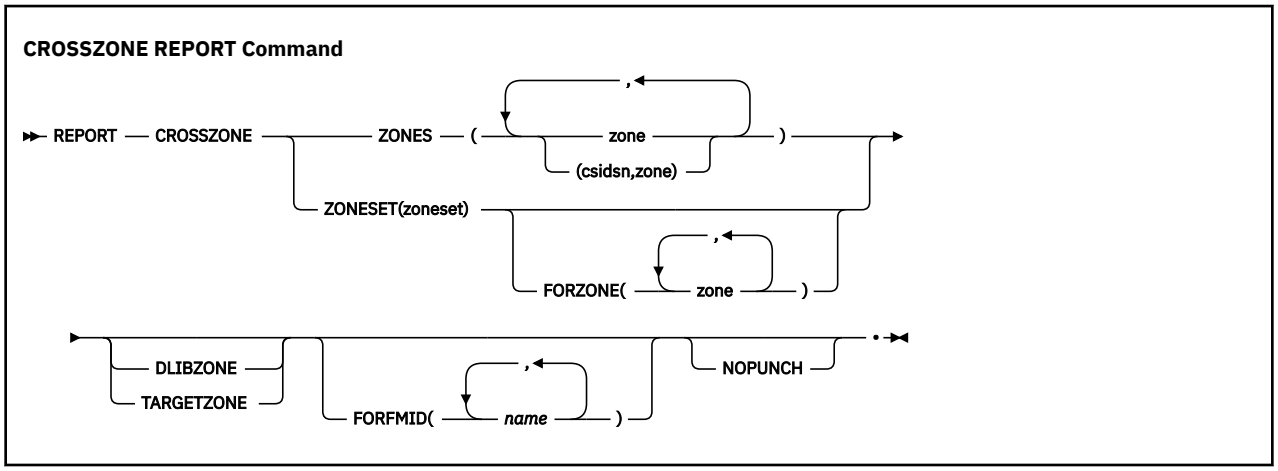
This command helps you synchronize the service levels of different products when the products are installed in different zones. It allows for the specification of target and DLIB zones that are defined either in the same global zone or in different global zones. Specifically, REPORT CROSSZONE lists conditional requisites that must be installed in certain zones because of SYSMODs that are installed in other zones. Information about these requisites is provided in the Cross-Zone Requisite SYSMOD report. The commands needed to install the requisites are written to the SMPPUNCH data set.

For example, suppose you have two versions of a product: one for z/OS and one for OS/390®. Each version is installed on a different system and in different target and distribution zones. To keep the two versions synchronized, the service for one version may specify service for the other version as a conditional requisite. However, because the versions are not in the same target or distribution zones, neither zone contains information about conditional requisites defined in the other zone. Therefore, SMP/E cannot use the information to keep the two versions synchronized. Instead, you can use the REPORT CROSSZONE command to obtain a summary of the requisite information and have SMP/E generate the commands needed to install the requisites into the appropriate zones.

Zones for SET BOUNDARY

For the REPORT CROSSZONE command, the SET BOUNDARY command must specify the global zone.

Syntax



Operands

CROSSZONE
requests a report about cross-zone requisites.

CROSSZONE is a required operand for the REPORT CROSSZONE command.

Note: CROSSZONE is mutually exclusive with CALLLIBS, ERRSYSMODS, SOURCEID, and SYSMODS.

DLIBZONE
indicates that SMP/E should report only on SYSMODs accepted into the distribution zones identified by the ZONESET or ZONES operand.

DLIBZONE is required if the ZONESET or ZONES operand being used contains both target and distribution zones, because the REPORT CROSSZONE command processes only zones of the same type. It is not required when the ZONESET or ZONES operand identifies only distribution zones.

Note:

1. DLIBZONE is allowed only on the REPORT CROSSZONE command.
2. DLIBZONE is mutually exclusive with TARGETZONE.
3. DLIBZONE can also be specified as DZONE.

FORFMID
specifies the FMIDs used to limit which SYSMODs are included in the report. This list can include FMIDs, FMIDSET names, or both.

If FORFMID is specified, SMP/E lists only the SYSMODs that are needed for these FMIDs.

If FORFMID is not specified, SMP/E reports on requisites based on the CIFREQ subentries in the SYSMOD entries for **all** the functions in the ZONESET zones.

Note: CIFREQ subentries list requisites for the function that were specified on ++IF statements in other SYSMODs. They also list the causer SYSMOD that contained the ++IF statement. For more information about CIFREQ subentries and conditional requisites, see "Conditional requisites (IFREQ)" on page 31.

FORZONE
specifies the zones to be reported on. SMP/E lists only the SYSMODs that are needed in these zones. All these zones must be part of the ZONESET.

If FORZONE is not specified, SMP/E reports on requisites for all the zones identified by the ZONESET operand.

Note:

1. FORZONE is allowed only on the REPORT CROSSZONE command.
2. FORZONE is mutually exclusive with ZONES.

NOPUNCH
indicates that SMP/E should not write any output to SMPPUNCH. If NOPUNCH is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

Note: The output produced by REPORT CROSSZONE processing contains commands for installing cross-zone requisites.

TARGETZONE
indicates that SMP/E should report only on SYSMODs applied to the target zones identified by the ZONESET or ZONES operand. TARGETZONE is required if the ZONESET or ZONES operand being used contains both target and distribution zones, because the REPORT CROSSZONE command processes only zones of the same type. It is not required when the ZONESET or ZONES operand identifies only target zones.

Note:

1. TARGETZONE is allowed only on the REPORT CROSSZONE command.
2. TARGETZONE is mutually exclusive with DLIBZONE.
3. TARGETZONE can also be specified as TZONE.

ZONES
a zone name, ZONESET or a global zone CSI data set name and zone (or ZONESET). A global zone CSI data set name and zone can be specified to compare zones defined in a different global zone. SMP/E will check each specified zone for information on conditional requisites that have been installed.

Note:

1. ZONES is mutually exclusive with ZONESET and FORZONE.
2. Either the ZONES or ZONESET operand is required for the REPORT CROSSZONE command.'
3. The maximum number of global zone CSIs that can be processed by a single REPORT CROSSZONE command is 253.

ZONESET
is the name of the global zone ZONESET entry that is used to report on cross-zone requisites. SMP/E checks all the zones in the ZONESET for information on conditional requisites that have been installed.

Note:

1. Either the ZONESET or ZONES operand is required for the REPORT CROSSZONE command.
2. ZONESET is mutually exclusive with ZONES.

For more information about defining a ZONESET, see *z/OS SMP/E Reference*.

Data sets used

The following data sets might be needed to run the REPORT CROSSZONE command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	*zone*
SMPLOG			

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

If you use the REPORT CROSSZONE command, keep these considerations in mind:

- After installing the requisite SYSMODs identified by the REPORT CROSSZONE command, you should run the command again to see whether that installation causes any new requisites. You should repeat this process until there are no new requisites.
- You should always check the Cross-Zone Requisite SYSMOD report for unreceived requisites before using the SMPPUNCH output from REPORT CROSSZONE. You may want to receive these SYSMODs in order to run the commands in SMPPUNCH, or you may prefer to delete them from SMPPUNCH and receive them later.

Output

Output from the REPORT CROSSZONE command includes reports, as well as data written to SMPPUNCH.

Reports

The following reports are produced during REPORT CROSSZONE processing:

- Cross-Zone Requisite SYSMOD report
- File Allocation report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

SMPPUNCH output

To make it easier for you to install cross-zone requisite SYSMODs, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no requisite SYSMODs for the specified zone.
- NOPUNCH was specified on the REPORT CROSSZONE command.

When a global zone CSI data set name is specified for one or more zones on the ZONES operand, SMPCSI and SMPCNTL DD statements will be included in the SMPPUNCH output. These DD statements will appear before the SET BDY command and must be copied into the users job before the output can be used. When all of the zones in the report are defined in the SET global zone, the SMPCSI and SMPCNTL DD statements will not appear in the SMPPUNCH output.

Figure 10 on page 296 shows the format of the SMPPUNCH output from the REPORT CROSSZONE command.

```
//* DD statement for zone zone1
//SMPCSI   DD DSN=globalcsi1,DISP=SHR
//SMPCNTL  DD  *
SET BDY    (zone1  ).
RESETRC.
command SELECT(
              sysmod1      /* REQUIRED DUE TO sysmod2 IN zone2 */
              )
      BYPASS(HOLDSYSTEM)
        CHECK
        GROUP.
/*

//* DD statement for zone zone3
//SMPCSI   DD DSN=globalcsi2,DISP=SHR
//SMPCNTL  DD  *
SET BDY  (zone3  ).
RESETRC.
command SELECT(
              sysmod3      /* REQUIRED DUE TO sysmod4 IN zone4 */
              )
       BYPASS(HOLDSYSTEM)
         CHECK
         GROUP.
/*
```

Figure 10. REPORT CROSSZONE: Format of SMPPUNCH Output

globalcsi1
the global CSI data set in which zone1 is defined.

globalcsi2
the global CSI data set in which zone3 is defined.

zone1, zone3
are the names of the zones where the requisites are to be installed.

command
is the command to be used to install the requisites: ACCEPT for a distribution zone, and APPLY for a target zone.

sysmod1, sysmod3
are the IDs of the requisite SYSMODS.

sysmod2, sysmod4
are the IDs of the SYSMODs that contained the CIFREQ data (the causer SYSMODs).

zone2, zone4
are the names of the zones that contained the CIFREQ data (the causer zones).

Note:

1. If there is more than one causer SYSMOD or causer zone for a selected SYSMOD, a comment is written for each of the causers. These comments refer to the previous SYSMOD in the select list.
2. If there are requisites for more than one zone, a set of commands is written for each zone.
3. You can edit the SMPPUNCH output before using it. For example, you may want to install SYSMODs for only one of the zones, or you may want to delete SYSMODs that have not yet been received.

Examples

The following examples are provided to help you use the REPORT CROSSZONE command.

Example 1: Using REPORT CROSSZONE with zones controlled by the same global zone

Assume that you have a system that supports z/OS and OS/390. There is one target zone (BASEZOS) for base z/OS functions and another target zone (PRODZOS) for a dependent function. Likewise, there is a target zone (OS390) for base OS/390 functions and another target zone (PROD90) for a dependent function. Table 19 on page 297 shows some of the functions and PTFs contained in each zone.

Table 19. REPORT CROSSZONE example: SYSMOD installed in each zone

Zone	Functions	PTFs
BASEZOS	HBB3310	UZ00005 UZ00009 UZ00011 UZ00013 UZ00031 UZ00032
PRODZOS	HJS3311	UZ00006 UZ00022 UZ00025 UZ00026
OS390	HBB2102 JBB2220	UZ00030 UZ00031
PROD90	HJS2220	UZ00032 UZ00033

You have also received the following SYSMODs but have not yet applied or accepted them:

UZ00023
UZ00024
UZ00027

Assume some of the PTFs specify conditional requisites. Table 20 on page 297 shows some of the statements in these PTFs (causer SYSMODs), along with the zones where they were installed (causer zones), the functions they are applicable to (causer FMIDs), the functions specified on the ++IF statements (IFREQ FMIDs), the zones where these functions are installed (IFREQ zones), and the requisites.

Table 20. REPORT CROSSZONE example: required SYSMODs

Causer SYSMODs	Causer zones and FMIDs	IFREQ zones and FMIDs	Required SYSMODs
++PTF(UZ00011). ++VER (Z038) FMID(HBB3310). ++IF FMID(HJS3311) REQ(UZ00023).	BASEZOS – HBB3310	PRODZOS – HJS3311	UZ00023

Table 20. REPORT CROSSZONE example: required SYSMODs (continued)

Causer SYSMODs	Causer zones and FMIDs	IFREQ zones and FMIDs	Required SYSMODs
++PTF(UZ00013). ++VER (Z038) FMID(HBB3310). ++IF FMID(HJS3311) REQ(UZ00024).	BASEZOS – HBB3310	PRODZOS – HJS3311	UZ00024
++PTF(UZ00006). ++VER (Z038) FMID(HJS3311). ++IF FMID(HBB3310) REQ(UZ00009).	PRODZOS – HJS3311	BASEZOS – HBB3310	UZ00009
++PTF(UZ00022). ++VER (Z038) FMID(HJS3311). ++IF FMID(HBB3310) REQ(UZ00005).	PRODZOS – HJS3311	BASEZOS – HBB3310	UZ00005
++PTF(UZ00025). ++VER (Z038) FMID(HJS3311). ++IF FMID(HJS2220) REQ(UZ00027).	PRODZOS – HJS3311	PROD90 – HJS2220	UZ00027
++PTF(UZ00026). ++VER (Z038) FMID(HJS3311). ++IF FMID(HJS2220) REQ(UZ00028).	PRODZOS – HJS3311	PROD90 – HJS2220	UZ00028
++PTF(UZ00030). ++VER (Z038) FMID(HBB2102). ++IF FMID(HJS2220) REQ(UZ00032).	OS390 – HBB2102	PROD90 – HJS2220	UZ00032
++PTF(UZ00031). ++VER (Z038) FMID(HBB2102). ++IF FMID(HJS2220) REQ(UZ00033).	OS390 – HBB2102	PROD90 – HJS2220	UZ00033

The dependent functions are different versions of the same product. They must be synchronized with each other and with their base functions. You can set up two ZONESETs (ZOSZSET and S390) to help keep these products at the same service level. Table 21 on page 298 shows the zones contained in each ZONESET:

Table 21. REPORT CROSSZONE example: ZONESETs to be used

ZONESET	Zones
ZOSZSET	BASEZOS PRODZOS PROD90
S390	OS390 PROD90 PRODZOS

Assume that you want to find out whether there are any cross-zone requisites for the zones in ZONESET ZOSZSET. You can use the following commands:

```
SET    BDY(GLOBAL)           /* Process global zone.       */.
REPORT CROSSZONE             /* Report on requisites       */
       ZONESET(ZOSZSET)      /* for ZONESET ZOSZSET.       */.
```

SMP/E checks zones BASEZOS, PRODZOS, and PROD90 because they are the zones defined in ZONESET ZOSZSET. Because F0RFMID was not specified, SMP/E checks the SYSMOD entries for **all** the functions installed in those zones. It makes a list of the CIFREQ subentries for all the functions. Then, because F0RZONE was not specified, SMP/E reports on requisites needed in **all** the zones in the ZONESET.

Figure 11 on page 299 shows an example of the report SMP/E produces:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

                 CROSSZONE REQUISITE SYSMOD REPORT FOR APPLY

 ZONE
REQUIRES                      CAUSER
 NAME     FMID     SYSMOD    RECEIVED  SYSMOD   FMID     ZONE


 BASEZOS            NONE


 PROD90   HJS2220  UZ00027   YES       UZ00025  HJS3311  PRODZOS
          HJS2220  UZ00028   NO        UZ00026  HJS3311  PRODZOS


 PRODZOS  HJS3311  UZ00023   YES       UZ00011  HBB3310  BASEZOS
          HJS3311  UZ00024   YES       UZ00013  HBB3310  BASEZOS
```

Figure 11. Example of a cross-zone requisite SYSMOD report

SMP/E also writes the commands shown in Figure 12 on page 299 to the SMPPUNCH data set:

```
SET BDY (PROD90 ).
RESETRC.
APPLY   SELECT(
               UZ00027   /* REQUIRED DUE TO UZ00025 IN PRODZOS */
               UZ00028   /* REQUIRED DUE TO UZ00026 IN PRODZOS */
              )
        BYPASS(HOLDSYSTEM
        CHECK
        GROUP.
SET BDY (PRODZOS).
RESETRC.
APPLY   SELECT(
               UZ00023   /* REQUIRED DUE TO UZ00011 IN BASEZOS  */
               UZ00024   /* REQUIRED DUE TO UZ00013 IN BASEZOS  */
              )

        BYPASS(HOLDSYSTEM
        CHECK
        GROUP.
```

Figure 12. Example of SMPPUNCH output for REPORT CROSSZONE

After getting the Cross-Zone Requisite SYSMOD report, you can take these actions:

1. Receive SYSMOD UZ00028 so you can install it in the PROD90 zone.
2. Use the SMPPUNCH output to install the requisite SYSMODs listed in the report.
3. Rerun the REPORT CROSSZONE command for the same ZONESET (ZOSZSET) to check for any additional requisites, and install any that are found.
4. Run the REPORT CROSSZONE command for the S390 ZONESET to keep zones PROD90 and OS390 synchronized.

5. Receive and install SYSMODs as needed for the zones in ZONESET S390.
6. Rerun the REPORT CROSSZONE command for S390, and install additional SYSMODs as needed.

Example 2: Using REPORT CROSSZONE with zones controlled by different global zones

Assume that you have the same setup as in example 1, except that the z/OS zones are controlled by a different global zone than the OS/390 zones. That is, target zones BASEZOS and PRODZOS are controlled by global zone SYS1.ZOS.CSI, and target zones OS390 and PROD90 are controlled by global zone SYS1.OS390.CSI. Also assume that:

- the functions and PTFs contained in each zone are the same as those in table 18
- the conditional requisites are the same as those shown in table 19, and
- you have received the following SYSMODs in both global zones but have not yet applied or accepted them:
  - UZ00023
  - UZ00024
  - UZ00027

Assume that SMPCSI is allocated to SYS1.ZOS.CSI and you want to find out whether there are any cross-zone requisites for zones BASEZOS and PRODZOS defined in global zone SYS1.ZOS.CSI and PROD90 defined in global zone SYS1.OS390.CSI. You can use the following commands:

```
SET             BDY(GLOBAL)    /* Process global zone. */.
REPORT          CROSSZONE      /* Report on requisites */
                      ZONES(BASEZOS, PRODZOS,
                           (SYS1.OS390.CSI,PROD90)).
```

SMP/E checks zones BASEZOS, PRODZOS and PROD90 because they are the specified zones. Because **FORFMID** was not specified, SMP/E checks the SYSMOD entries for all the functions installed in those zones. It makes a list of the CIFREQ subentries for all the functions. Then, because **FORZONE** was not specified, SMP/E reports on requisites that are needed in all three zones.

Figure 13 on page 300 shows an example of the report SMP/E produces. It includes the global zone CSI data set name when it was specified for zones on the ZONES operand. A footnote and key notation are used where a footnote is placed next to a zone name in the report and an entry for the CSI data set name is included in a key at the bottom of the report. If all of the zones in the report are defined in the SET TO global zone, the CSI global zone names are not included in the report.

```
PAGE nnnn - NOW SET TO xxxxxx ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST OUTPUT

                              CROSSZONE REQUISITE SYSMOD REPORT FOR APPLY

ZONE                      REQUISITE                                  CAUSER
NAME          FMID        SYSMODS       RECEIVED      SYSMOD       FMID          ZONE

BASEZOS(1)                NONE

PROD90(2)     HJS2220     UZ00027       YES           UZ00025      HJS3311       PRODZOS(1)
              HJS2220     UZ00028       NO            UZ00026      HJS3311       PRODZOS(1)

PRODZOS(1)    HJS3311     UZ00023       YES           UZ00011      HBB3310       BASEZOS(1)
              HJS3311     UZ00024       YES           UZ00013      HBB3310       BASEZOS(1)

GLOBAL ZONE CSI DATA SET KEY:
1 - SYS1.ZOS.CSI
2 - SYS1.OS390.CSI
```

Figure 13. Example of a cross-zone requisite SYSMOD report

SMP/E also writes the commands shown in Figure 14 on page 301 to the SMPPUNCH data set. Since the reported zones are defined in separate global CSI data sets, the SMPCSI and SMPCNTL DD statements

are included in the SMPPUNCH output. These DD statements must be copied to your job before the output is used.

```
//* DD statement for zone PROD90
//SMPCSI  DD  DSN=SYS1.OS390.CSI,DISP=SHR
//SMPCNTL DD  *
SET BDY(PROD90).
RESETRC.
APPLY  SELECT(
              UZ00027   /* REQUIRED DUE TO UZ00025 IN BASEZOS */
              UZ00028   /* REQUIRED DUE TO UZ00026 IN BASEZOS */
              )
       BYPASS(HOLDSYSTEM)
       CHECK
       GROUP.
/*

//* DD statement for zone PRODZOS
//SMPCSI  DD  DSN=SYS1.ZOS.CSI,DISP=SHR
//SMPCNTL DD  *
SET BDY(PRODZOS).
RESETRC.
APPLY  SELECT(
              UZ00023   /* REQUIRED DUE TO UZ00011 IN PRODZOS */
              UZ00024   /* REQUIRED DUE TO UZ00013 IN PRODZOS */
              )
       BYPASS(HOLDSYSTEM)
       CHECK
       GROUP.
/*
```

Figure 14. Example of SMPPUNCH output for REPORT CROSSZONE

Processing

The REPORT CROSSZONE command checks across zones identified by the ZONESET or ZONES operand for conditional requisites that need to be installed.

SMP/E first verifies that each zone identified by the ZONESET or ZONES operand is defined in the global zone and that all zones specified on the FORZONE operand are identified in the ZONESET operand. Then SMP/E determines which zones will be used.

- If TZONE was specified, SMP/E checks that the ZONESET or ZONES operand contains target zones and uses those target zones to process the REPORT CROSSZONE command.
- If DZONE was specified, SMP/E checks that the ZONESET or ZONES operand contains distribution zones and uses those distribution zones to process the REPORT CROSSZONE command.
- If neither DZONE nor TZONE was specified, SMP/E checks that all the zones in the ZONESET or ZONES operand are the same type. If so, it uses all the zones in the ZONESET or ZONES operand to process the REPORT CROSSZONE command.

The zones are opened for read access. If NOPUNCH was **not** specified, the SMPPUNCH data set is also opened. In addition, if FORFMID was specified, the FMIDs and FMIDSETs specified on the FORFMID operand are used to create a list of FMIDs to be reported on.

Next, SMP/E makes a list of all the CIFREQ subentries contained in the zones being used for this REPORT CROSSZONE command. If the FORFMID operand was specified, SMP/E uses only the CIFREQ subentries from SYSMOD entries for functions specified in the FMID list. Otherwise, it lists all the CIFREQ subentries.

Note: CIFREQ subentries are only in SYSMOD entries for functions. They list requisites for the function that were specified on an ++IF statement in another SYSMOD. They also list the causer SYSMOD that contained the ++IF statement. For more information about CIFREQ subentries and conditional requisites, see “Conditional requisites (IFREQ)” on page 31.

SMP/E then checks the CIFREQ list and functions against each of the FORZONE zones (or against all of the zones being used if FORZONE was not specified). If the function is installed in the zone (and has not

been deleted or superseded) and the causer SYSMOD is not installed there, SMP/E checks whether the requisite is installed in the zone. For each requisite that is not installed, SMP/E writes information for that requisite in the Cross-Zone Requisite SYSMOD report. In addition, commands to install the requisite SYSMODs are written to the SMPPUNCH data set. SMP/E does this processing zone by zone until all the zones to be reported on have been checked. It then notes which of the requisites have not yet been received and writes this information to the SMPRPT data set as well. Finally, it closes all the data sets.

If **any** of the following occurs, SMP/E issues an error message, and REPORT CROSSZONE processing fails:

- A zoneset or zone specified in the ZONES or ZONESET operand is not defined in the global zone.
- The zone type in the ZONEINDEX subentry does not match the zone type in the zone definition entry.
- The zones specified by the ZONES or ZONESET operand are not all the same type, and neither DZONE nor TZONE was specified.
- TZONE was specified, but there were no target zones specified by the ZONES or ZONESET operand.
- DZONE was specified, but there were no distribution zones specified by the ZONES or ZONESET operand.
- A zone on the FORZONE operand is not among the zones identified by the ZONESET operand.
- TZONE was specified, and a zone on the FORZONE operand is not a target zone.
- DZONE was specified, and a zone on the FORZONE operand is not a distribution zone.
- NOPUNCH was not specified, but there is no definition for the SMPPUNCH data set.

Zone and data set sharing considerations

The following identifies the phases of REPORT CROSSZONE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

Global zone (multiple as required)	Read without enqueue.
Target zones (as required)	Read without enqueue.
DLIB zones (as required)	Read without enqueue.

2. Processing

Global zone (multiple as required)	Read with shared enqueue.
Target zones (as required)	Read with shared enqueue.
DLIB zones (as required)	Read with shared enqueue.

3. Termination

   All resources are freed.

Chapter 17. The REPORT ERRSYSMODS command

This command helps you determine whether any SYSMODs you have already processed are now exception SYSMODs. It also helps you determine whether any resolving SYSMODs are available for held SYSMODs.

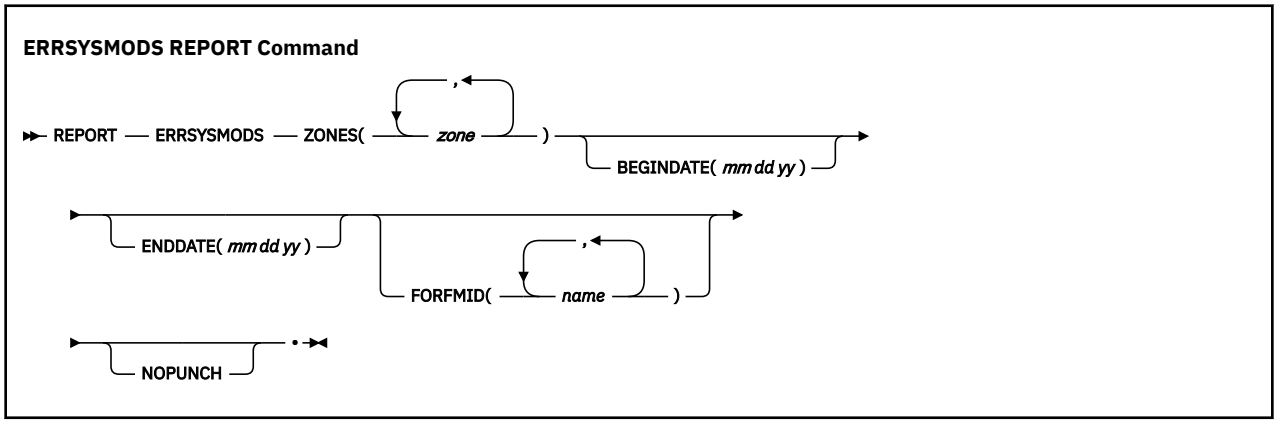
- For target and distribution zones, REPORT ERRSYSMODS lists installed SYSMODs for which ++HOLD statements were subsequently received and whose error reason IDs have not yet been resolved.
- For the global zone, REPORT ERRSYSMODS lists received SYSMODs for which ++HOLD statements with error reason IDs have been received.

Information about the held SYSMODs and any resolving SYSMODs is provided in the Exception SYSMOD report. The commands needed to install the resolving SYSMODs are written to the SMPPUNCH data set.

Zones for SET BOUNDARY

For the REPORT ERRSYSMODS command, the SET BOUNDARY command must specify the global zone.

Syntax



Operands

BEGINDATE
indicates that you should use ++HOLD statements received by SMP/E on or after the specified date for REPORT ERRSYSMODS processing. BEGINDATE can be used alone or with ENDDATE to define the range of the ++HOLD statements to be used.

The date is specified as *mm dd yy*, where *mm* is the month (01–12), *dd* is the day (01–31), and *yy* is the year (00–99). Blanks separate the month, day, and year.

Note:

1. BEGINDATE is allowed only on the REPORT ERRSYSMODS command.
2. If BEGINDATE is specified without ENDDATE, SMP/E uses either the DATE parameter on the GIMSMP EXEC statement or the current date as the end date.

ENDDATE
indicates that you should use ++HOLD statements received by SMP/E on or before the specified date for REPORT ERRSYSMODS processing. ENDDATE can be used alone or with BEGINDATE to define the range of the ++HOLD statements to be used.

The date is specified as *mm dd yy*, where *mm* is the month (01–12), *dd* is the day (01–31), and *yy* is the year (00–99). Blanks separate the month, day, and year.

Note:

1. ENDDATE is only allowed on the REPORT ERRSYSMODS command.
2. If ENDDATE is specified without BEGINDATE, SMP/E processes all HOLDDATA processed by SMP/E on or before the specified end date.

ERRSYSMODS
requests a report about SYSMODs for which ++HOLD modification control statements with error reason IDs have been received.

- For target and distribution zones, SMP/E reports on SYSMODs that meet **all** these conditions:
  - They have been installed in any of the specified target and distribution zones. (They are not installed in error and have not been deleted.)
  - They are in HOLDERROR status in the global zone. (++HOLD statements with error reason IDs have been received for the SYSMODs.)
  - The error reason IDs are not resolved. (That is, the error reason ID has not been installed, and no SYSMODs that supersede the error reason ID have been installed.)
- For the global zone, SMP/E reports on SYSMODs that meet these conditions:
  - They have been received.
  - They are in HOLDERROR status in the global zone. (++HOLD statements with error reason IDs have been received for the SYSMODs.)

ERRSYSMODS is a required operand for the REPORT ERRSYSMODS command.

Note:

1. ERRSYSMODS is mutually exclusive with CALLLIBS, CROSSZONE, SOURCEID, and SYSMODS.
2. ERRSYSMODS can also be specified as ERRSYS.

FORFMID
specifies the FMIDs used to limit which SYSMODs are included in the report. This list can include FMIDs, FMIDSET names, or both.

If FORFMID is specified, SMP/E lists only the SYSMODs that have ++HOLD statements with these FMIDs.

If FORFMID is not specified, SMP/E reports on all the affected SYSMODs in the specified zones.

NOPUNCH
indicates that SMP/E should not write any output to SMPPUNCH. If NOPUNCH is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

Note: The output produced by REPORT ERRSYSMODS processing contains commands for installing SYSMODs that resolve the error hold reason IDs for exception SYSMODs.

ZONES
specifies which zones SMP/E should report on. This list can include zone names, ZONESET names, or both. You can specify the global zone, target zones, distribution zones, or any combination of these. SMP/E produces a separate report for each zone it checks.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named SYS1 and a zone named SYS1. If you specify SYS1 on this operand, SMP/E assumes you want to use the zones defined in ZONESET SYS1 (which might or might not include zone SYS1), and not the individual zone SYS1.

ZONES is a required operand on the REPORT ERRSYSMODS command.

Data sets used

The following data sets might be needed to run the REPORT ERRSYSMODS command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPCSI	SMPPUNCH	SMPSNAP
SMPLOGA	SMPLOG	SMPRPT	*zone*
SMPOUT			

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

After you run REPORT ERRSYSMODS for target or distribution zones, the Exception SYSMOD report may indicate that some of the resolving SYSMODs are themselves held. In this case, the Exception SYSMOD report will also show any resolving SYSMODs for the additional holds. If you want to install the additional resolving SYSMODs indicated in the report, you can use the SMPPUNCH output produced for the applicable target or distribution zones as a starting point for creating the necessary SMP/E commands.

Output

Output from the REPORT ERRSYSMODS command includes reports, as well as data written to SMPPUNCH.

Reports

The following reports are produced during REPORT ERRSYSMODS processing:

- Exception SYSMOD report
- File Allocation report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

SMPPUNCH output

To make it easier for you to install resolving SYSMODs for exception SYSMODs, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RECEIVE (for unreceived resolving SYSMODs), RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no exception SYSMODs for the specified zone.
- There are no resolving SYSMODs for any of the exception SYSMODs or all resolving SYSMODs identified are held.
- The specified zone is the global zone.
- NOPUNCH was specified on the REPORT ERRSYSMODS command.

Figure 15 on page 306 shows the format of the SMPPUNCH output from the REPORT ERRSYSMODS command.

```
SET BDY (GLOBAL).  /* REMOVE COMMENT IF DOING RECEIVE
RECEIVE SELECT(
        sysmod0
              )
        SYSMODS.
                   REMOVE COMMENT IF DOING RECEIVE */
RESETRC.
SET BDY (zone   ).
command SELECT(
       sysmod1   /* type RESOLVES reason  for sysmod2 FMID(sysmod3) */
      *** OR ***
    /* sysmod1      type RESOLVES reason  for sysmod2 FMID(sysmod3) */
              )
        GROUP CHECK
        BYPASS(HOLDSYSTEM, HOLDCLASS(ERREL)).
```

Figure 15. REPORT ERRSYSMODS: Format of SMPPUNCH Output

zone
is the name of the target or distribution zone in which the exception SYSMOD is currently installed.

command
is the command to be used to install the resolving SYSMODs: ACCEPT for a distribution zone, APPLY for a target zone.

sysmod0
is the ID of a resolving SYSMOD that has not been received.

sysmod1
is the ID of a resolving SYSMOD for the error reason ID.

A SYSMOD is commented out in the following cases:

- The command is ACCEPT and the SYSMOD is an APAR fix. This is to avoid accepting APAR fixes into a distribution zone.
- The SYSMOD has not been received.
- The SYSMOD is held for an error reason ID other than ERREL.
- The SYSMOD was already listed in the SMPPUNCH output.
- The SYSMOD is a function. This is to avoid inadvertently installing a function.

type
is the SYSMOD type of the resolving SYSMOD.

reason
is the APAR that caused the exception SYSMOD to be held.

sysmod2
is the ID of the exception SYSMOD.

sysmod3
is the FMID of the function to which the exception SYSMOD applies.

You can edit the SMPPUNCH output before using it. For example, if you need to receive a resolving SYSMOD, you can remove the comments from the RECEIVE command, after verifying the SYSMODs in the SELECT list.

Example: Using REPORT ERRSYSMODS

Assume that you have just received some HOLDDATA, and you need to know whether it affects any of the SYSMODs already accepted into distribution zone DZONE1. You can use the following commands:

```
SET    BDY(GLOBAL)         /* Process global zone.    */.
REPORT ERRSYSMODS          /* Report on exception     */
       ZONES(DZONE1)       /* SYSMODs in this zone    */
```

```
        BEGINDATE(07 01 07) /* for HOLDDATA received   */
        ENDDATE(08 01 07)   /* between these dates.    */.
```

Figure 16 on page 307 is an example of the report SMP/E produces:

```
PAGE 0001  - NOW SET TO GLOBAL ZONE          DATE 08/02/07  TIME 16:08:43  SMP/E 25.00 SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE DZONE1   DATE: 07/01/98 - 08/01/98

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

HMJ4102  HMJ4102  AN78422  AN78422 GOOD   YES        HIPER   IPL,FAILS WITH E37 ABEND
                           UW31189 HELD   YES
                           UW32001 GOOD   YES
                  AN80332  AN80332 GOOD   YES        HIPER   DAL,PRV,FUL
                           UW37822 GOOD   YES
                  AN80501  UW38922 HELD   YES        HIPER   PRV
HQA5140  HQA5140  AN90012  AN90012 GOOD   YES        HIPER   DAL
                           UW42146 HELD   YES

-------------------------------------------------------------------------------------------------------

PAGE 0002  - NOW SET TO GLOBAL ZONE          DATE 08/02/07  TIME 16:08:43  SMP/E 25.00 SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE DZONE1   DATE: 07/01/98 - 08/01/98

FIXES FOR HELD RESOLVING SYSMODS

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

HMJ4102  UW31189  AN80203  UW32213 GOOD   YES        PE
                           UW36378 HELD   NO
                           UW36402 GOOD   YES
         UW36378  AN81345  AN81345 GOOD   YES        PE
                           UW37011 GOOD   NO
         UW38922  AN81401  UW39013 ERREL  YES        PE
HQA5140  UW42146  AN90025  UW43610 GOOD   NO         PE


-------------------------------------------------------------------------------------------------------

PAGE 0003  - NOW SET TO GLOBAL ZONE          DATE 08/02/07  TIME 16:08:43  SMP/E 25.00 SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT SUMMARY           DATE: 07/01/98 - 08/01/98

ZONE     FMID        TOTAL APARS        TOTAL RESOLVING
                    AGAINST FMID      SYSMODS AGAINST FMID

DZONE1   HMJ4120            6                 12
         HQA5140            2                  3
```

Figure 16. Example of an exception SYSMOD report

SMP/E also writes the commands shown in Figure 17 on page 308 to the SMPPUNCH data set:

```
SET BDY(GLOBAL  ). /* REMOVE COMMENT IF DOING RECEIVE
RECEIVE SELECT(
        UW36378
        UW37011
        UW43610
             )
         SYSMODS.
                  REMOVE COMMENT IF DOING RECEIVE */
RESETRC.
SET BDY(TGT1    ).
APPLY  SELECT(
       AN78422  /* APAR     RESOLVES AN78422 FOR HMJ4102 FMID(HMJ4102) */
    /* UW31189     PTF      RESOLVES AN78422 FOR HMJ4102 FMID(HMJ4102) */
       UW32213  /* PTF      RESOLVES AN80203 FOR UW31189 FMID(HMJ4102) */
    /* UW36378     PTF      RESOLVES AN80203 FOR UW31189 FMID(HMJ4102) */
       UW36402  /* PTF      RESOLVES AN80203 FOR UW31189 FMID(HMJ4102) */
       UW32001  /* PTF      RESOLVES AN78422 FOR HMJ4102 FMID(HMJ4102) */
       AN80332  /* APAR     RESOLVES AN80332 FOR HMJ4102 FMID(HMJ4102) */
       UW37822  /* PTF      RESOLVES AN80332 FOR HMJ4102 FMID(HMJ4102) */
    /* UW38922     PTF      RESOLVES AN80501 FOR HMJ4102 FMID(HMJ4102) */
       UW39013  /* PTF      RESOLVES AN81211 FOR UW39013 FMID(HMJ4102) */
       AN81345  /* APAR     RESOLVES AN81345 FOR UW36378 FMID(HMJ4102) */
    /* UW37011     PTF      RESOLVES AN81345 FOR UW36378 FMID(HMJ4102) */
       AN90012  /* APAR     RESOLVES AN90012 FOR HQA5140 FMID(HQA5140) */
    /* UW42146     PTF      RESOLVES AN90012 FOR HQA5140 FMID(HQA5140) */
    /* UW43610     PTF      RESOLVES AN90025 FOR UW41246 FMID(HQA5140) */
             )
        GROUP CHECK
        BYPASS(HOLDSYSTEM, HOLDCLASS(ERREL)).
```

Figure 17. Example of SMPPUNCH output for REPORT ERRSYSMODS

After getting the Exception SYSMOD report, you can take these actions:

1. Examine the SMPPUNCH output and edit as necessary.
2. Use the SMPPUNCH output to install the resolving SYSMODs in DZONE1.

Processing

The REPORT ERRSYSMODS command checks the zones specified on the ZONES operand to determine whether SYSMODs that have been processed are now exception SYSMODs.

SMP/E first verifies that each target and distribution zone specified on the ZONES operand is defined in the global zone. The zones and ZONESETs specified on the ZONES operand (including the global zone) are used to create a list of zones to be reported on. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name.

The zones are opened for read access. If NOPUNCH was **not** specified, the SMPPUNCH data set is also opened. In addition, if FORFMID was specified, the FMIDs and FMIDSETs specified on the FORFMID operand are used to create a list of FMIDs to be reported on.

Next, SMP/E makes a list of all the HOLDDATA entries for error reason IDs. If the FORFMID operand was specified, this HOLDERROR list contains only HOLDDATA entries containing ++HOLD statements with an FMID value that is specified in the FMID list. If BEGINDATE was specified, the HOLDERROR list contains entries only for HOLDDATA received by SMP/E on or after the specified date. Likewise, if ENDDATE was specified, the HOLDERROR list contains entries only for HOLDDATA received by SMP/E on or before the specified date. If neither BEGINDATE or ENDDATE was specified, the HOLDERROR list contains all the HOLDDATA entries.

For each zone in the zone list, SMP/E processes the HOLDERROR list to determine the SYSMODs to be reported on.

- For target and distribution zones, SMP/E reports on SYSMODs that meet these conditions:
  - They have been installed in any of the specified target and distribution zones. (They are not installed in error and have not been deleted.)

  - They are in HOLDERROR status in the global zone. (++HOLD statements with error reason IDs have been received for the SYSMODs.)
  - The error reason ID is not resolved. (That is, the error reason ID has not been installed, and no SYSMODs that supersede the error reason ID have been installed.)
- For the global zone, SMP/E reports on SYSMODs that meet these conditions:
  - They have been received.
  - They are in HOLDERROR status in the global zone. (++HOLD statements with error reason IDs have been received for the SYSMODs.)

After determining which SYSMODs to report on for the specified zone, SMP/E checks the global zone for any resolving SYSMODs. A SYSMOD resolves an error reason ID if it either matches or specifically supersedes that reason ID. When Enhanced HOLDDATA is used, a resolving SYSMOD can also be identified by FIX information in the COMMENT operand of the ++HOLD statement. For each exception SYSMOD in the specified zone, SMP/E writes information about that SYSMOD and any resolving SYSMODs in the Exception SYSMOD report. In addition, for target and distribution zones, commands to install the resolving SYSMODs are written to the SMPPUNCH data set. SMP/E does this processing zone by zone until all the zones to be reported on have been checked. Then SMP/E closes all the data sets.

If any of the following occurs, SMP/E issues an error message, and REPORT ERRSYSMODS processing fails:

- A zone, ZONESET, or zone in a ZONESET specified on the ZONES operand (other than the global zone) is not defined in the global zone.
- NOPUNCH was not specified, but there is no definition for the SMPPUNCH data set.
- The date range specified by the BEGINDATE and ENDDATE operands is invalid (for example, if the beginning date is later than the ending date).

Zone and data set sharing considerations

The following identifies the phases of REPORT ERRSYSMODS processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zones (as required)
   Read without enqueue.

   DLIB zones (as required)
   Read without enqueue.

2. Processing

   Global zone
   Read with shared enqueue.

   Target zones (as required)
   Read with shared enqueue.

   DLIB zones (as required)
   Read with shared enqueue.

3. Termination

   All resources are freed.

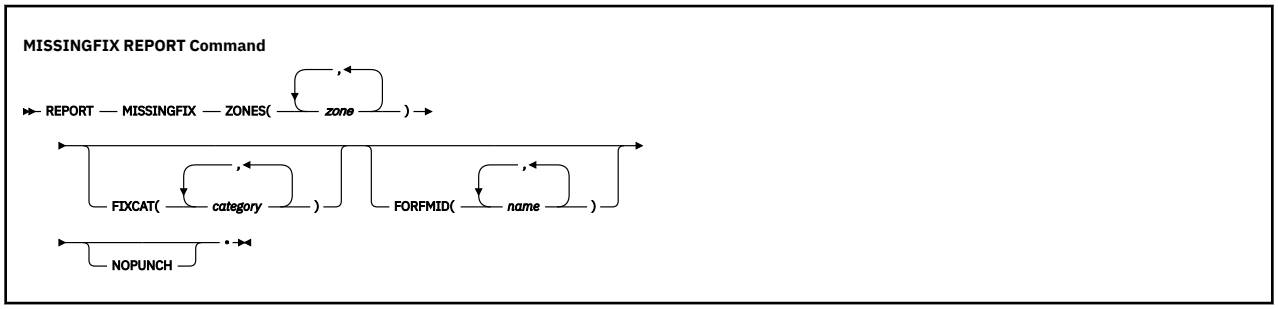
Chapter 18. The REPORT MISSINGFIX command

This command helps you determine whether any FIXCAT APARs exist that are applicable and have not yet been installed, and whether any SYSMODs are available to satisfy the missing FIXCAT APARs. More specifically, the REPORT MISSINGFIX command reports on SYSMODs that have been applied or accepted for which subsequent FIXCAT HOLDs have been received and whose reason IDs have not yet been resolved. The actual FIXCAT HOLDs analyzed during report processing are determined by the fix categories of interest specified by the user.

Zones for SET BOUNDARY

For the REPORT MISSINGFIX command, the SET BOUNDARY command must specify the global zone.

Syntax



Operands

MISSINGFIX
requests a report about missing FIXCAT APARs. That is, fixes for APARs that have an applicable and interesting fix category value.

MISSINGFIX is a required operand for the REPORT MISSINGFIX command.

FIXCAT
identifies the list of fix categories of interest to be reported on. This list determines which missing FIXCAT APARs are to be included in the report. SMP/E reports only the missing FIXCAT APARs identified by FIXCAT type ++HOLD statements with these fix categories.

The values specified on the FIXCAT operand will override the list of values, if any, defined by the FIXCAT subentry in the active OPTIONS entry.

Fix category values can be 1 to 64 characters in length and contain any nonblank character in the range X'41' - X'FE'except single quotation marks, commas, left parentheses, and right parentheses. The value can be specified in one of two ways:

- Explicitly, by fully specifying a particular fix category value, such as IBM.Device.zIIP, for example. In this case, all HOLDDATA associated with this fix category will be applicable to command processing.
- Implicitly, by partially specifying a fix category value using any number of asterisks (*) as global characters and percent signs (%) as placeholders.
  - A single asterisk indicates that zero or more characters can occupy that position. For example, IBM.Device* implies that all HOLDDATA associated with a fix category that begins with the character string IBM.Device are applicable *z/OS implies that all HOLDDATA associated with a fix category that ends with the character string z/OS are applicable. Finally, IBM*z/OS implies that all HOLDDATA associated with a fix category that begins with the character string IBM and ends with the character string z/OS are applicable.

- A single percent sign indicates that any one single character can occupy that position. For example, IBM.Device.20%4 implies that HOLDDATA associated with any of the following fix categories is applicable: IBM.Device.2084, IBM.Device.2094, and IBM.Device.20t4. HOLDDATA associated with the fix category IBM.Device.20914, however, is not applicable.

Fix category values can contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified Fix category value. For example, a specified value of IBM.FUNCTION.HEALTHCHECKER matches the value of IBM.Function.HealthChecker.

The following examples are acceptable fix category values:

```
IBM.Device.zIIP
*
IBM.Function*
IBM.Device.20%4.*
IBM.HealthChecker
```

FORFMID
specifies the FMIDs used to limit which missing FIXCAT APARs are included in the report. This list can include FMIDs, FMIDSET names, or both. If FORFMID is specified, SMP/E reports only the missing APARs identified by ++HOLD statements with these FMIDs. If FORFMID is not specified, SMP/E reports all the missing FIXCAT APARs in the specified zones.

NOPUNCH
indicates that SMP/E should not write any output to SMPPUNCH. If NOPUNCH is not specified, sample SMP/E commands for installing the missing SYSMODs are written to SMPPUNCH. In addition, a sample RECEIVE command is created for ordering and receiving resolving SYSMODs that are not found in the global zone.

ZONES
specifies which zones SMP/E should report on. This list can include zone names, ZONESET names, or both. You can specify target zones, distribution zones, or any combination of these. SMP/E produces a separate report for each zone it checks.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name.

Data sets used

The following data sets might be needed to run the REPORT MISSINGFIX command. They can be defined by DD statements or, usually, by DDDEF entries.For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPCSI	SMPPUNCH	SMPSNAP
SMPLOGA	SMPLOG	SMPRPT	*zone*
SMPOUT			

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

After you run REPORT MISSINGFIX for target or distribution zones, the Missing FIXCAT SYSMOD report might indicate that some of the resolving SYSMODs are held. This means one or more ERROR holds exists for them. In this case, the second part of the Missing FIXCAT SYSMOD report will also show any resolving SYSMODs for the additional holds. If you want to install the additional resolving SYSMODs indicated in the report, you can use the SMPPUNCH output produced for the applicable target or distribution zones as a starting point for creating the necessary SMP/E commands.

Output

Output from the REPORT MISSINGFIX command includes reports, as well as data written to SMPPUNCH.

Reports

The following reports are produced during REPORT MISSINGFIX processing:

- Missing FIXCAT SYSMOD Report
- File Allocation report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

SMPPUNCH output

To make it easier for you to install resolving SYSMODs for exception SYSMODs, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RECEIVE (for unreceived resolving SYSMODs), RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no exception SYSMODs for the specified zone.
- There are no resolving SYSMODs for any of the exception SYSMODs or all resolving SYSMODs identified are held.
- The specified zone is the global zone.
- NOPUNCH was specified on the REPORT MISSINGFIX command.

Figure 18 on page 313 shows the format of the SMPPUNCH output from the REPORT MISSINGFIX command.

```
SET BDY(GLOBAL).
    RECEIVE ORDER(
    CONTENT(ALL)/*
    CONTENT(PTFS(
                aaaaaaa bbbbbbb ccccccc ddddddd
                  ))
                 */
    ORDERSERVER(xxxxxxxx)
    CLIENT(yyyyyy)
                  )
    DELETEPKG.
SET BDY(tgtzone).
    APPLY CHECK
    SELECT(
    /*fix category*/
           eeeeeee
           fffffff
         )
   BYPASS(HOLDSYSTEM)
    GROUPEXTEND.
```

Figure 18. REPORT MISSINGFIX: Format of SMPPUNCH Output

zone
is the name of the target or distribution zone in which the exception SYSMOD is currently installed.

command
is the command to be used to install the resolving SYSMODs: ACCEPT for a distribution zone, APPLY for a target zone.

aaaaaaabbbbbbbccccccc
is the ID of a resolving SYSMOD that has not been received.

Example: Using REPORT MISSINGFIX

Assume that you have just received some HOLDDATA and you want to know whether there are any new FIXCAT HOLDs related to the following categories

- IBM.Device.2094
- z/OS IBM.Device.zIIP
- IBM.Function.DataSharing.MVS/ESA
- IBM.HealthChecker

which affect any of the SYSMODs already applied into target zone TGT. You can use the following commands:

```
SET BDY(GLOBAL).                                    /* Process a GLOBAL zone             */
REPORT MISSINGFIX                                   /* report on FIXCAT APARs          */
       ZONES(TGT)                                   /* missing from this zone          */
       FIXCAT(IBM.Device.2094.z/OS,       /* related to these categories     */
              IBM.Device.zIIP,
              IBM.Function.DataSharing.MVS/ESA,
              IBM.HealthChecker).
```

Figure 19 on page 314 is an example of the report that SMP/E produces:

```
PAGE nnnn  - NOW SET TO GLOBAL ZONE   DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

MISSING FIXCAT SYSMOD REPORT FOR ZONE TGT

                               HOLD       MISSING        HELD          ______RESOLVING SYSMOD______
FIX CATEGORY      FMID         CLASS      APAR           SYSMOD        NAME      STATUS    RECEIVED
------------      -------      -----      -------        -------       -------   ------    --------
IBM.Device.2094.z/OS
                  HBB7730      PSP        AA13644        HBB7730       UA27033   GOOD      NO
                                          AA14941        HBB7730       UA26443   GOOD      NO
                                          AA15968        HBB7730       UA27113   GOOD      YES
                                          AA16005        HBB7730       UA26745   GOOD      NO
                                          AA16529        HBB7730       UA26741   HELD      NO
                                          AA17268        HBB7730       UA27861   GOOD      NO
                  HRM7730      PSP        AA16458        HRM7730       UA28236   GOOD      NO

IBM.Device.zIIP
                  HBB7730      PSP        AA15968        HBB7730       UA27113   GOOD      NO
                                          AA16005        HBB7730       UA26475   GOOD      NO
                  HRM7730      PSP        AA16458        HRM7730       UA28236   GOOD      NO
                                          AA18772        HRM7730       ***NONE

IBM.Function.DataSharing.MVS/ESA
                  HBB7730      PSP        AA13335        HBB7730       UA24231   GOOD      NO
                                          AA16416        HBB7730       UA26375   GOOD      NO
                                          AA16534        HBB7730       UA29059   HELD      NO
                                                                       UA30114   GOOD      YES
                                          AA16640        HBB7730       UA27891   GOOD      NO
                                          AA17188        HBB7730       UA29175   GOOD      NO

IBM.HealthChecker
                  HBB7730      PSP        AA15593        HBB7730       UA28094   GOOD      NO
                                          AA16687        HBB7730       UA27678   GOOD      NO
                  HRF7730      PSP        AA15290        HRF7730       UA26196   GOOD      NO
                                          AA17429        HRF7730       UA28412   GOOD      NO
```

Figure 19. Missing FIXCAT SYSMOD report: sample part 1

```
PAGE nnnn  - NOW SET TO GLOBAL ZONE   DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

MISSING FIXCAT SYSMOD REPORT FOR ZONE TGT                              - FIXES FOR HELD RESOLVING SYSMODS

HOLD           HELD                        ______RESOLVING SYSMOD______          HOLD
FMID           SYSMOD        APAR      NAME      STATUS     RECEIVED             CLASS
-------        -------       -------   -------   -------    ---------            -------
HBB7730        UA26741       AA22874   UA34271   GOOD       NO                   PE
                                       UA36122   GOOD       YES                  PE
               UA29059       AA24875   UA36625   HELD       NO                   PE
               UA36625       AA43771   ***NONE                                   PE
                             AA45962   ***NONE                                   PE
```

Figure 20. Missing FIXCAT SYSMOD report, sample part 2

Here is an example of the SMPPUNCH output for REPORT MISSINGFIX:

```
SET BDY(GLOBAL). /*

   THE FOLLOWING SMP/E COMMANDS WERE GENERATED BY A REPORT MISSINGFIX COMMAND ON
mm/dd/yy AT hh:mm:ss.

       */.

RECEIVE ORDER(
        CONTENT(ALL) /*

          SMP/E RECOMMENDS ORDERING AND RECEIVING ALL APPLICABLE PTFS.
IF YOU CHOOSE NOT TO ORDER ALL, THEN ORDER ONLY THE RESOLVING PTFS:

                    CONTENT(PTFS(
                UA24231 UA26196 UA26375 UA26443 UA26741 UA26745 UA27033
                      UA27678 UA27861 UA27891 UA28084 UA28236 UA28412 UA29059
              )    )
          */
        ORDERSERVER(SMPOSRVR) /* SPECIFY THE ORDERSERVER DDNAME.   */
        CLIENT     (SMPCLNT ) /* SPECIFY THE CLIENT DDNAME.       */
              )
        DELETEPKG.
SET BDY(tgtzone).
  APPLY CHECK
        SELECT(
  /* IBM.Device.2094.z/OS                                          */
                UA26443
                UA26741
                UA26745
                UA27033
                UA27113
                UA27861
                UA28236
  /*
IBM.Device.zIIP
             */
              /* UA26745 */
              /* UA27113 */
              /* UA28236 */
  /*
IBM.Function.DataSharing.MVS/ESA                                                */
                UA24231
                UA26375
                UA27891
                UA29059
                UA29175
  /* IBM.HealthChecker                                             */
                UA26196
                UA27678
                UA28084
                UA28412
              )
BYPASS(HOLDSYSTEM)
GROUPEXTEND.
```

Figure 21. Example of SMPPUNCH output for REPORT MISSINGFIX

Processing

1. The REPORT MISSINGFIX command checks the zones specified on the ZONES operand to determine if there are any missing fixes based on the fix categories of interest.
2. SMP/E then verifies that each target and distribution zone specified on the ZONES operand is defined in the global zone as either a ZONESET entry or as a zone with a ZONEINDEX subentry. The zones and ZONESETs specified on the ZONES operand (including the global zone) are used to create a list of zones to be reported on. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a ZONESET and a zone both exist with the same name, then the ZONESET is used.

   Only target and dlib zones may be reported on. The global zone is ignored if specified within a ZONESET.

3. If NOPUNCH was **not** specified, the SMPPUNCH data set is allocated. In addition, if FORFMID was specified, the FMIDs and FMIDSETs specified on the FORFMID operand are used to create a list of FMIDs to be reported on.
4. The purpose of the REPORT MISSINGFIX command is to analyze only the FIXCAT type HOLDDATA that has a fix category of interest. If the FIXCAT operand was specified on the command, the values specified are used as the active fix category list. Otherwise, the values in the FIXCAT subentry of the active OPTIONS entry are used.
5. FIXCAT HOLDDATA entries identify APARs associated with one or more fix categories for a held SYSMOD. SMP/E makes a list of FIXCAT type HOLDDATA entries to be reported on by analyzing all FIXCAT type HOLDDATA entries found in the global zone. HOLDDATA entries will be considered for the report if one or more of the entry's fix category values matches one or more category values in the active interest list, and the FORFMID operand was not specified, or the FORFMID operand was specified and the entry's FMID matches a value in the FMID list.
6. For each target or dlib zone in the zone list, SMP/E processes the list of FIXCAT type HOLDs to determine which APARs to report as missing. For each FIXCAT type HOLD, SMP/E determines if the held SYSMOD is installed in the target or dlib zone. If the held SYSMOD is installed, then SMP/E determines if the APAR reason ID is installed or superseded. If the APAR SYSMOD is not installed or superseded, then it is "missing" and included in the report.
7. After SMP/E identifies all missing APARs, SMP/E must then find in the global zone all SYSMODs that may resolve the missing APAR. A resolving SYSMOD is the fixing PTF identified by the RESOLVER operand on the FIXCAT ++HOLD statement (if specified), and any SYSMOD in the global zone that supersedes the missing APAR. SMP/E includes each resolving SYSMOD it finds in the report.

   In addition, for each resolving SYSMOD, SMP/E determines if the SYSMOD is received in the global zone and if it is held for one or more ERRORs. If a SYSMOD entry exists in the global zone and the entry is not in Error status, then SMP/E considers the resolving SYSMOD as received and reports it as such. If the SYSMOD has one or more HOLDERR subentries in the global zone, SMP/E reports its status as HELD. Otherwise, SMP/E reports SYSMOD status as GOOD.
8. If any resolving SYSMODs are reported with a status of HELD, SMP/E produces the second part of the Missing FIXCAT SYSMOD reportt, Fixes for Held Resolving SYSMODs.

   More specifically, SMP/E reports each unique ERROR HOLD for the held resolving SYSMODs (each unique APAR). For each ERROR HOLD, SMP/E must find in the global zone all SYSMODs that may resolve the ERROR HOLD. Resolving SYSMODs are the fixing PTF identified by the FIX operand in the SMRTDATA of the ++HOLD ERROR statement (if specified), and any SYSMOD in the global zone that supersedes the ERROR HOLD reason ID. SMP/E includes each resolving SYSMOD found in the report.

   In addition, for each resolving SYSMOD, SMP/E determines if the SYSMOD is received in the global zone and if it is also held for one or more ERRORs. If a SYSMOD entry exists in the global zone and the entry is not in Error status, then the resolving SYSMOD is considered received and reported as such. If the SYSMOD has one or more HOLDERR subentries in the global zone, then its status in the report is HELD. Otherwise, its status in the report is GOOD. If a resolving SYSMOD is reported with a status of HELD, then it too will be reported as a HELD SYSMOD in this part 2 of the report.
9. After all reports are produced and if the NOPUNCH operand is **NOT** specified, then SMP/E generates the sample commands and writes them to the SMPPUNCH data set.

Zone and data set sharing considerations

The following list identifies the phases of REPORT MISSINGFIX processing and the zones and data sets SMP/E might require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zones (as required)
   Read without enqueue.

DLIB zones (as required)
Read without enqueue.

2. Processing

Global zone
Read with shared enqueue.

Target zones (as required)
Read with shared enqueue.

DLIB zones (as required)
Read with shared enqueue.

3. Termination

All resources are freed.

Chapter 19. The REPORT SOURCEID command

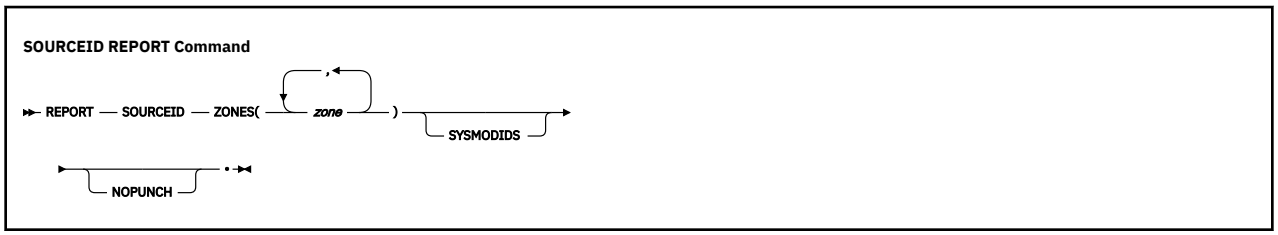
This command helps you determine which source IDs have been assigned to SYSMODs in a given zone or ZONESET.

Information about these source IDs is provided in the SOURCEID report. The commands needed to list the entries for SYSMODs associated with these source IDs are written to the SMPPUNCH data set.

Zones for SET BOUNDARY

For the REPORT SOURCEID command, the SET BOUNDARY command must specify the global zone.

Syntax



Operands

NOPUNCH
indicates that SMP/E should not write any output to SMPPUNCH. If NOPUNCH is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

Note: The output produced by REPORT SOURCEID processing contains commands for listing SYSMODs associated with the source IDs that were found in the specified zones.

SOURCEID
requests a report listing all the source IDs assigned to SYSMODs in a given zone or ZONESET.

SOURCEID is a required operand for the REPORT SOURCEID command.

Note: SOURCEID is mutually exclusive with CALLLIBS, CROSSZONE, ERRSYSMODS, and SYSMODS.

SYSMODIDS
indicates that in the SOURCEID report, SMP/E should also include the IDs of the SYSMODs to which each source ID is assigned.

If SYSMODIDS is not specified, the SOURCEID report contains the source IDS assigned to SYSMODs in the zones being reported on, but not the SYSMOD IDs associated with those source IDS.

Note: SYSMODIDS is allowed only on the REPORT SOURCEID command.

ZONES
specifies which zones SMP/E should report on. This list can include zone names, ZONESET names, or both. You can specify the global zone, target zones, distribution zones, or any combination of these. SMP/E produces a separate report for each zone it checks.

For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. For example, suppose you have a ZONESET named ZOSZSET and a zone named ZOSZSET. If you specify Z0SZSET on this operand, SMP/E assumes you want to use the zones defined in ZONESET ZOSZSET (which might or might not include zone ZOSZSET), and not the individual zone ZOSZSET.

ZONES is a required operand on the REPORT SOURCEID command.

Data sets used

The following data sets might be needed to run the REPORT SOURCEID command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	*zone*
SMPLOG			

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statementscan be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

Output from the REPORT SOURCEID command includes reports, as well as data written to SMPPUNCH.

Reports

The following reports are produced during REPORT SOURCEID processing:

- SOURCEID report
- File Allocation report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

SMPPUNCH output

To make it easier for you to list the SYSMODs associated with the SOURCEIDs named in the SOURCEID report, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and LIST SYSMOD SOURCEID(...). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no SYSMOD entries in the zone.
- There are no source IDs assigned to SYSMODs in the zone.
- NOPUNCH was specified on the REPORT SOURCEID command.

Figure 22 on page 320 shows the format of the SMPPUNCH output from the REPORT SOURCEID command.

```
SET BDY(zonename).
RESETRC.
LIST SYSMOD SOURCEID(
                     sourceid
                     sourceid
                    ).
SET BDY(zonename).
RESETRC.
LIST SYSMOD SOURCEID(
                     sourceid
                     sourceid
                    ).
```

Figure 22. REPORT SOURCEID: format of SMPPUNCH output

zonename
is the name of a zone specified on the ZONES operand as an individual zone or a member of a ZONESET.

sourceid
is the source ID assigned to a SYSMOD in the specified zone.

You can edit the SMPPUNCH output before using it.

Examples

The following examples are provided to help you use the REPORT SOURCEID command:

Example 1: REPORT SOURCEID (SYSMODIDS operand specified)

Assume that you want to find out which source IDs are associated with SYSMODs in target zone TGT1, and you want to know which SYSMODs each source ID is assigned to. You can use the following commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
       ZONES(TGT1)
       SYSMODIDS.
```

Figure 23 on page 321 is an example of the report SMP/E writes:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

          SOURCEID REPORT FOR TARGET ZONE TGT1

NOTE: * - INDICATES THE SYSMOD HAS MULTIPLE SOURCEIDS

SOURCEID    _________________SYSMOD NAMES_________________

PUT0703     UZ00023*  UZ00024   UZ00035   UZ00037   UZ00039
            UZ00052   UZ00073   UZ00076   UZ00077

SMCREC      UZ00015   UZ00023*  UZ00044
```

Figure 23. Example of a SOURCEID report (SYSMODIDS operand specified)

SMP/E also writes the commands shown in Figure 24 on page 321 to the SMPPUNCH data set:

```
SET BDY(TGT1).
RESETRC.
LIST SYSMOD SOURCEID(
                     PUT0703
                     SMCREC
                    ).
```

Figure 24. Example of SMPPUNCH output for REPORT SOURCEID (SYSMODIDS operand specified)

Example 2: REPORT SOURCEID (SYSMODIDS operand not specified)

Assume that you want to find out the source IDs associated with SYSMODs in target zone TGT2, but you are not interested in knowing the specific SYSMODs that each SOURCEID is assigned to. You can use the following commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
       ZONES(TGT2).
```

Figure 25 on page 322 is an example of the report SMP/E writes:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

          SOURCEID REPORT FOR TARGET ZONE TGT2


_________________________SOURCEIDS_____________________________

PUT0606   PUT0607   PUT0608   PUT0701   PUT0702   PUT0703
PUT0704
```

Figure 25. Example of a SOURCEID report (SYSMODIDS operand not specified)

SMP/E also writes the commands shown in Figure 26 on page 322 to the SMPPUNCH data set:

```
SET BDY(TGT2).
RESETRC.
LIST SYSMOD SOURCEID(
                     PUT0606
                     PUT0607
                     PUT0608
                     PUT0701
                     PUT0702
                     PUT0703
                     PUT0704
                    ).
```

Figure 26. Example of SMPPUNCH output for REPORT SOURCEID (SYSMODIDS operand not specified)

Example 3: REPORT SOURCEID (ZONES operand specified)

Assume that you want to find out the source IDs associated with SYSMODs in DLB3, plus all the zones defined by ZONESET MYPROD3. (MYPROD3 defines two zones: DLB3 and TGT3.) You are not interested in knowing the specific SYSMODs each source ID is assigned to. You can use the following commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
       ZONES(DLB3, MYPROD3).
```

SMP/E writes one report for each zone specified by the ZONES operand. Even though DLB3 is specified individually and also included by ZONESET MYPROD3, it is reported on only once.

Figure 27 on page 323 shows examples of the reports SMP/E would write for DLB3 and TGT3:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

          SOURCEID REPORT FOR DLIB   ZONE DLB3


_________________________SOURCEIDS_____________________________

PUT0606   PUT0607   PUT0608   PUT0701   PUT0702   PUT0703
PUT0704

PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

          SOURCEID REPORT FOR TARGET ZONE TGT3


_________________________SOURCEIDS_____________________________

PUT0606   PUT0607   PUT0608   PUT0701   PUT0702
```

Figure 27. Example of SOURCEID reports (ZONES operand specified)

SMP/E also writes the commands shown in Figure 28 on page 323 to the SMPPUNCH data set:

```
SET BDY(DLB3).
RESETRC.
LIST SYSMOD SOURCEID(
                     PUT0606
                     PUT0607
                     PUT0608
                     PUT0701
                     PUT0702
                     PUT0703
                     PUT0704
                    ).
SET BDY(TGT3).
RESETRC.
LIST SYSMOD SOURCEID(
                     PUT0606
                     PUT0607
                     PUT0608
                     PUT0701
                     PUT0702
                    ).
```

Figure 28. Example of SMPPUNCH output for REPORT SOURCEID (ZONES operand specified)

Processing

The REPORT SOURCEID command checks the SYSMOD entries in a zone and reports on any source IDs found in those entries. The resulting output can be used to list the SYSMOD entries associated with the source IDs that were found.

SMP/E first verifies that each target and distribution zone specified on the ZONES operand is defined in the global zone. The zones and ZONESETs specified on the ZONES operand (including the global zone) are used to create a list of zones to be reported on. For each specified value, SMP/E first checks for a ZONESET with the same name. If none is found, SMP/E checks for a zone with the same name. If a zone is specified separately and is also part of a ZONESET, it is only reported on once.

The zones are opened for read access. If NOPUNCH was **not** specified, the SMPPUNCH data set is also opened.

For each zone to be reported on, SMP/E checks all the SYSMOD entries in that zone to see if they contain source IDs.

- For each SYSMOD entry containing a source ID, SMP/E saves the SOURCEID value and the SYSMOD ID.
- If a SYSMOD entry contains multiple source IDs, SMP/E saves all its SOURCEID values. It also saves an indicator so the SOURCEID report can note that the SYSMOD has multiple source IDs.
- If a SYSMOD is in error, it is not considered installed. As a result, SMP/E ignores any source IDs associated with that SYSMOD.

SMP/E then writes a SOURCEID report for each zone specified by the ZONES operand.

- If SYSMODIDS was specified on the REPORT SOURCEID command, the report includes the IDs of the SYSMODs associated with each SOURCEID value found in the zone.
- If SYSMODIDS was not specified on the REPORT SOURCEID command, the report includes all the SOURCEID values found in the zone, but not the IDs of the SYSMODs associated with those source IDs.

In addition, if NOPUNCH was **not** specified, SMP/E writes commands to the SMPPUNCH data set to list the SYSMODs associated with the source IDs that were found. Finally, SMP/E closes all the data sets.

If any of the following occurs, SMP/E issues an error message, and REPORT SOURCEID processing fails:

- A zone, ZONESET, or zone in a ZONESET specified on the ZONES operand (other than the global zone) is not defined in the global zone.
- NOPUNCH was not specified, but there is no definition for the SMPPUNCH data set.

Zone and data set sharing considerations

The following identifies the phases of REPORT SOURCEID processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zones (as required)
   Read without enqueue.

   DLIB zones (as required)
   Read without enqueue.

2. Processing

   Global zone
   Read without enqueue.

   Target zones (as required)
   Read without enqueue.

   DLIB zones (as required)
   Read without enqueue.

3. Termination

   All resources are freed.

Chapter 20. The REPORT SYSMODS command

The REPORT SYSMODS command helps you compare the SYSMODs installed in two zones. These are the types of zones you can compare:

- A DLIB zone to a DLIB zone
- A target zone to a target zone
- A DLIB zone to a target zone
- A target zone to a DLIB zone

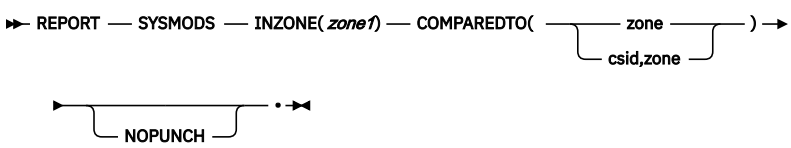
Information about the SYSMODs is provided in the SYSMOD Comparison report. This report identifies the SYSMODs that are installed in the input zone, but not found in the comparison zone. The commands needed to install the SYSMODs that are not found in the comparison zone are written to the SMPPUNCH data set. Information about the SYSTEM and USER HOLDs that must be resolved before these SYSMODs can be installed is provided in the SYSMOD Comparison HOLDDATA report.

Zones for SET BOUNDARY

For the REPORT SYSMODS command, the SET BOUNDARY command must specify the global zone.

Syntax

REPORT SYSMODS Command



Operands

COMPAREDTO
specifies the zone (called the *comparison zone*) that SMP/E should compare against the input zone for SYSMOD content. You can specify a single target zone or distribution zone name. In addition, a global zone CSI data set and zone name can be specified if the comparison zone is defined in a different global zone. The COMPAREDTO zone must **not** be the same as the zone specified on the INZONE operand. A report is issued to indicate which SYSMODs were in the input zone but not in the comparison zone.

COMPAREDTO is a required operand on the REPORT SYSMODS command.

Note: COMPAREDTO is allowed only on the REPORT SYSMODS command.

INZONE
specifies the input zone that SMP/E should use to compare against another zone (called the *comparison zone*) for SYSMOD content. You can specify a single target zone or distribution zone name. This must **not** be the same as the zone specified on the COMPAREDTO zone. A report is issued to indicate which SYSMODs were in the input zone but not in the comparison zone.

INZONE is a required operand on the REPORT SYSMODS command.

Note: INZONE is allowed only on the REPORT SYSMODS command.

NOPUNCH
indicates that SMP/E should not write any output to SMPPUNCH. If NOPUNCH is **not** specified, JCL or commands are written to SMPPUNCH. This output contains JCL or commands that can be used for further processing.

Note: The output produced by REPORT SYSMODS processing contains commands for installing SYSMODs from the input zone that are applicable to the comparison zone.

SYSMODS
requests a report comparing the SYSMOD content of two zones.

SYSMODS is a required operand for the REPORT SYSMODS command.

Note: SYSMODS is mutually exclusive with CALLLIBS, CROSSZONE, SOURCEID, and ERRSYSMODS.

Data sets used

The following data sets might be needed to run the REPORT SYSMODS command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	*zone*
SMPLOG			

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

Output from the REPORT sysmods command includes reports, as well as data written to SMPPUNCH.

Reports

The following reports are produced during REPORT SYSMODS processing:

- SYSMOD Comparison report
- SYSMOD Comparison HOLDDATA report
- File Allocation report

When a global zone CSI data set name is specified for the COMPAREDTO zone, the CSI data set name will appear in the heading of the SYSMOD Comparison report.

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

SMPPUNCH output

To make it easier for you to install the applicable SYSMODs named in the SYSMOD Comparison report, SMP/E writes the necessary commands to the SMPPUNCH data set: SET BOUNDARY, RESETRC, and either ACCEPT (for distribution zones) or APPLY (for target zones). Nothing is written to SMPPUNCH for a specified zone in the following cases:

- There are no applicable SYSMODs in the input zone.
- NOPUNCH was specified on the REPORT SYSMODS command.

When a global zone CSI data set name is specified on the COMPAREDTO operand, SMPCSI and SMPCNTL DD statements will be included in the SMPPUNCH output. These DD statements will appear before the

SET BDY command and must be copied into the users job before the output can be used. When the global zone CSI data set name is not specified on the COMPAREDTO operand, the SMPCSI and SMPCNTL DD statements will not be included in the SMPPUNCH output.

The code sample below shows the format of the SMPPUNCH output from the REPORT SYSMODS command.

```
//* DD statement for zone zone2
//SMPCSI    DD  DSN=globalcsi,DISP=SHR
//SMPCNTL   DD *
SET BDY (zone2).
RESETRC        /*

  THE  FOLLOWING command COMMAND(S) WERE GENERATED BY A REPORT SYSMODS
  COMMAND ON yy.ddd AT hh:mm:ss.  THE  SELECTED SYSMODS WERE  FOUND IN
  THE INPUT  ZONE (zone1_) BUT   NOT IN THE   COMPARISON ZONE (zone2_)
  AND APPEAR APPLICABLE TO THE COMPARISON ZONE.

  SYSMODS IN THE  SELECT LIST ARE  GROUPED BY FMID.   FUNCTIONS APPEAR
  FIRST FOLLOWED BY SERVICE SYSMODS (PTFS, APARS, AND USERMODS IN THAT
  ORDER).

  FUNCTIONS AND  PTFS THAT WERE  AVAILABLE IN THE  GLOBAL ZONE AND ARE
  APPLICABLE  TO THE  COMPARISON ZONE  APPEAR AS  NORMAL  SELECT  LIST
  VALUES WITH AN  APPROPRIATE COMMENT.   FUNCTIONS AND  PTFS THAT WERE
  NOT AVAILABLE  IN THE  GLOBAL  ZONE OR  HAVE  UNKNOWN  APPLICABILITY
  APPEAR  AS SMP  COMMENTS.   PTFS  FOR AN  APPLICABLE  FUNCTION,  NOT
  AVAILABLE IN THE  GLOBAL ZONE,  APPEAR AS COMMENTS  EVEN IF THEY ARE
  AVAILABLE.   SUCH COMMENTED SYSMODS CAN BE  CHANGED TO NORMAL SELECT
  LIST VALUES WHEN THEY OR THEIR APPLICABLE FUNCTIONS ARE RECEIVED  OR
  ARE DETERMINED TO BE APPLICABLE.

  APARS AND USERMODS APPEAR AS SMP COMMENTS.   THESE COMMENTED SYSMODS
  CAN  BECOME NORMAL  SELECT LIST VALUES IF YOU DESIRE TO INSTALL THEM
  IN THE zone2__ ZONE AND THEY ARE AVAILABLE IN THE GLOBAL ZONE.   THE
  COMMENTARY  TEXT  INDICATES  THEIR  AVAILABILITY  AND  APPLICABILITY
  STATUS.

  THE COMMAND(S) WERE GENERATED WITH THE BYPASS(HOLDSYSTEM), GROUP AND
  CHECK OPTIONS.  GROUP CAUSES REQUISITE SYSMODS TO BE  INCLUDED ALONG
  WITH THE  SELECTED SYSMODS.  CHECK  CAUSES A  DRYRUN OF THE  COMMAND
  BEFORE  LIBRARIES  ARE UPDATED  SO THAT ANY NEEDED CORRECTIVE ACTION
  CAN BE TAKEN BEFORE REAL   INSTALLATION.  BYPASS(HOLDSYSTEM)  CAUSES
  ALL SYSTEM HOLD REASON IDS TO BE BYPASSED.

  REDO WAS  GENERATED IF A FUNCTION IS TO BE  REINSTALLED.  IF REDO IS
  USED,  ONLY THE  FUNCTION  BEING  REINSTALLED  AND  ITS  SERVICE ARE
  SELECTED ON THAT COMMAND.

                       **** WARNING ****

  IT IS   POSSIBLE  THAT  ALL  SYSMODS  SELECTED  WILL  APPEAR  AS SMP
  COMMENTS. IF THIS IS THE CASE, SMP WILL ISSUE A SYNTAX  ERROR IF YOU
  RUN THE GENERATED COMMAND(S) AS IS.

                       **** WARNING ****
                                                                    */
             .
```

```
command
 SELECT(
    sysmdid    /* smdtype FOR fmid RECEIVED                          */
 /* sysmdid       smdtype FOR fmid NOT RECEIVED                      */
 /* sysmdid       smdtype FOR fmid RECEIVED, fmid NOT RECV'D         */
 /* sysmdid       smdtype FOR fmid NOT RECEIVED, fmid NOT RECV'D     */
 /* sysmdid       smdtype FOR fmid RECEIVED,  APPLICABILITY UNKNOWN  */
 /* sysmdid       smdtype FOR fmid NOT RECV'D, APPLICABILITY UNKNOWN */
 /* sysmdid       APAR    FOR fmid RECEIVED                          */
 /* sysmdid       APAR    FOR fmid NOT RECEIVED                      */
 /* sysmdid       USERMOD FOR fmid RECEIVED                          */
 /* sysmdid       USERMOD FOR fmid NOT RECEIVED                      */
             )
 [REDO]
 GROUP
 BYPASS (HOLDSYSTEM)
 CHECK.
/*
```

globalcsi
the global zone CSI data set in which *zone2* is defined.

zone2
is the name of the target or distribution zone specified on the COMPAREDTO operand (the comparison zone).

command
is the command to be used to install the SYSMODs: ACCEPT for a distribution zone, and APPLY for a target zone.

yy.ddd
is the year and Julian date that the command was generated by the REPORT SYSMODS command.

hh:mm:ss
is the time of day at which the command was generated by the REPORT SYSMODS command.

zone1
is the name of the target or distribution zone specified on the INZONE operand (the input zone).

sysmdid
is the ID of an applicable SYSMOD that appears in the SYSMOD Comparison report.

A SYSMOD is commented out in the following cases:

- The SYSMOD is an APAR fix or a USERMOD.
- The SYSMOD or its owning function (FMID) is not in the global zone.
- SMP/E cannot determine whether the SYSMOD is applicable to the comparison zone.

smdtype
is the SYSMOD type of the indicated SYSMOD.

fmid
is the SYSMOD ID of the function that owns the indicated SYSMOD.

You can edit the SMPPUNCH output before using it. For example, if any applicable SYSMODs were not in the global zone at the time of the report and you have since received them, you may want to change the comments for those SYSMODs to normal SELECT list values and install them.

Note: There may have been SYSMODs whose applicability was unknown (for example, they were not in the global zone), and which you have determined to be applicable. You may want to receive these SYSMODs and change them from comments to normal select list values so they can be installed.

You can determine whether a SYSMOD is applicable to the comparison zone by checking the program directory or installation manual for the FMID to find out the SREL, or you can receive the SYSMOD into the global zone and run the REPORT SYSMODS command again.

Multiple APPLY or ACCEPT commands may be generated if one or more functions are being reinstalled. Each function SYSMOD to be reinstalled appears on a separate APPLY or ACCEPT command along with its own service. Other SYSMODs belonging to functions not being reinstalled appear on a separate APPLY or ACCEPT command.

Example 1: Using REPORT SYSMODS with zones controlled by the same global zone

Assume that you have two systems. The target zones controlling these systems are TGZONE1 and TGZONE2, and they are serviced from the same global zone. You want to determine which SYSMODs are installed in TGZONE1 and are not installed in, but are applicable to, TGZONE2. The following chart shows the SYSMODs installed in the TGZONE1 and TGZONE2 zones and the SYSMODs available in the global zone at the time the REPORT SYSMODS command is run.

Note: All the zones have a single SREL value of Z038. The SYSMODs for all the zones are annotated with information about their FMID, source IDs, and installation status. These are the meanings of the annotations:

- DELBY indicates that the SYSMOD has not been installed, but its SYSMOD entry contains the DELBY subentry. The entry was created when a function was installed that deleted the indicated SYSMOD.
- ERR0R indicates that the SYSMOD has only been partially installed. Errors occurred during APPLY or ACCEPT processing.
- SUP0NLY indicates that the SYSMOD is a superseded-only SYSMOD.
- F= is followed by the SYSMOD's FMID.
- S= is followed by the SYSMOD's source IDs.

Table 22. REPORT SYSMODS example: SYSMODs installed in each zone

Zone	Functions	PTFs	APARs	USERMODs
TGZONE1	HAA1202 HBB1102 DELBY HBB1202 JBB1122 SUPONLY JBB1222 F=HBB1202	UZ00001 F=HBB1202 S=PUT0706 UZ00002 F=HBB1202 S=PUT0707 PDO0001 UZ00011 F=HAA1202 S=PUT0706 UZ00012 F=HAA1202 S=PUT0707 UZ00013 F=HAA1202 S=PUT0707 ERROR UZ00021 F=JBB1222 S=PUT0706 UZ00022 F=JBB1222 S=PUT0707	AZ00001 SUPONLY AZ00002 SUPONLY AZ00011 SUPONLY AZ00012 SUPONLY AZ00013 F=HAA1202 S=RETAIN AZ00021 SUPONLY AZ00022 SUPONLY	TZ00001 F=HAA1202
TGZONE2	HBB1202 JBB1122 SUPONLY JBB1222 SUPONLY JBB1322 F=HBB1202	UZ00001 F=HBB1202 S=PUT0706 UZ00031 F=JBB1322 S=PUT0706 UZ00032 F=JBB1322 S=PUT0707	AZ00001 SUPONLY AZ00031 SUPONLY AZ00032 SUPONLY	None

Table 22. REPORT SYSMODS example: SYSMODs installed in each zone (continued)

Zone	Functions	PTFs	APARs	USERMODs
GLOBAL	HAA1202 JBB1322 F=HBB1202	UZ00002 F=HBB1202 S=PUT0707 UZ00012 F=HAA1202 S=PUT0707 UZ00022 F=JBB1222 S=PUT0707 UZ00031 F=JBB1322 S=PUT0706 UZ00032 F=JBB1322 S=PUT0707	None	TZ00001 F=HAA1202

Assume that you want to find out which SYSMODs are installed in zone TGZONE1 and are not installed in zone TGZONE2, but are applicable to it. You can use the following commands:

```
SET    BDY(GLOBAL)          /* Process global zone.    */.
REPORT SYSMODS              /* Report on SYSMODs.      */
       INZONE(TGZONE1)      /* Input zone TGZONE1.     */
       COMPAREDTO(TGZONE2)  /* Comparison zone TGZONE2.*/.
```

SMP/E compares the SYSMOD content of zone TGZONE1 to that of zone TGZONE2. Any SYSMODs that are in zone TGZONE1 (with the exceptions noted under “Processing” on page 333) and are not in zone TGZONE2 appear in the resulting report.

Figure 29 on page 330 shows the report SMP/E produces:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

SYSMOD COMPARISON REPORT FOR TARGET ZONE TGZONE1 AND TARGET ZONE TGZONE2

MATCHING SREL(S) - Z038

FMID___  SYSMOD_  TYPE____  APPLICABLE  RECEIVED  SOURCEIDS

HAA1202  HAA1202  FUNCTION  YES         YES
         UZ00011  PTF       YES         NO        PUT0706
         UZ00012  PTF       YES         YES       PUT0707
         AZ00013  APAR      YES         NO        RETAIN
         TZ00001  USERMOD   YES         YES
HBB1202  UZ00002  PTF       YES         YES       PUT0707
                                                   PDO0001
JBB1222  UZ00021  PTF       NO          NO        PUT0706
         UZ00022  PTF       NO          YES       PUT0707
```

Figure 29. Example of a SYSMOD comparison report

If a SYSMOD is identified in the SYSMOD Comparison report, and SYSTEM or USER HOLDDATA entries exist for that SYSMOD in the global zone for the input zone, those HOLDs are reported in the SYSMOD Comparison HOLDDATA report. See “SYSMOD comparison HOLDDATA report” on page 523 for an example of the report.

SMP/E also writes the commands shown in the code sample below to the SMPPUNCH data set:

```
SET BDY (TGZONE2).
RESETRC        /*

  THE  FOLLOWING APPLY  COMMAND(S) WERE  GENERATED BY A REPORT SYSMODS
  COMMAND ON yy.ddd AT hh:mm:ss.  THE  SELECTED SYSMODS WERE  FOUND IN
  THE INPUT  ZONE (TGZONE1) BUT  NOT IN THE  COMPARISON ZONE (TGZONE2)
  AND APPEAR APPLICABLE TO THE COMPARISON ZONE.

  SYSMODS IN THE  SELECT LIST ARE  GROUPED BY FMID.   FUNCTIONS APPEAR
  FIRST FOLLOWED BY SERVICE SYSMODS (PTFS, APARS, AND USERMODS IN THAT
  ORDER).

  FUNCTIONS AND  PTFS THAT WERE  AVAILABLE IN THE  GLOBAL ZONE AND ARE
  APPLICABLE  TO THE  COMPARISON ZONE  APPEAR AS  NORMAL  SELECT  LIST
  VALUES WITH AN  APPROPRIATE COMMENT.   FUNCTIONS AND  PTFS THAT WERE
  NOT AVAILABLE  IN THE  GLOBAL  ZONE OR  HAVE  UNKNOWN  APPLICABILITY
  APPEAR  AS SMP  COMMENTS.   PTFS  FOR AN  APPLICABLE  FUNCTION,  NOT
  AVAILABLE IN THE  GLOBAL ZONE,  APPEAR AS COMMENTS  EVEN IF THEY ARE
  AVAILABLE.   SUCH COMMENTED SYSMODS CAN BE  CHANGED TO NORMAL SELECT
  LIST VALUES WHEN THEY OR THEIR APPLICABLE FUNCTIONS ARE RECEIVED, OR
  ARE DETERMINED TO BE APPLICABLE.

  APARS AND USERMODS APPEAR AS SMP COMMENTS.   THESE COMMENTED SYSMODS
  CAN  BECOME NORMAL  SELECT LIST VALUES IF YOU DESIRE TO INSTALL THEM
  IN THE TGZONE2 ZONE AND THEY ARE AVAILABLE IN THE GLOBAL ZONE.   THE
  COMMENTARY  TEXT  INDICATES  THEIR  AVAILABILITY  AND  APPLICABILITY
  STATUS.

  THE COMMAND(S)  WERE  GENERATED  WITH THE BYPASS(HOLDSYSTEM), GROUP,
  AND CHECK OPTIONS.  GROUP CAUSES REQUISITE  SYSMODS  TO BE  INCLUDED
  ALONG  WITH THE  SELECTED SYSMODS.  CHECK  CAUSES  A  DRYRUN OF  THE
  COMMAND BEFORE LIBRARIES ARE  UPDATED SO THAT  ANY NEEDED CORRECTIVE
  ACTION CAN BE  TAKEN BEFORE  REAL INSTALLATION.   BYPASS(HOLDSYSTEM)
  CAUSES ALL SYSTEM HOLD REASON IDS TO BE BYPASSED.

  REDO WAS GENERATED  IF A  FUNCTION IS TO  BE REINSTALLED. IF REDO IS
  USED, ONLY  THE  FUNCTION BEING  REINSTALLED AND  ITS  SERVICE   ARE
  SELECTED ON THAT COMMAND.

                       **** WARNING ****

  IT IS   POSSIBLE  THAT  ALL  SYSMODS  SELECTED  WILL  APPEAR  AS SMP
  COMMENTS.  IF THIS IS THE CASE, SMP WILL ISSUE A SYNTAX ERROR IF YOU
  RUN THE GENERATED COMMAND(S) AS IS.

                       **** WARNING ****
                                                                    */
            .
```

```
APPLY
 SELECT(
    HAA1202     /* FUNCTION FOR HAA1202 RECEIVED                    */
 /* UZ00011        PTF      FOR HAA1202 NOT RECEIVED                */
    UZ00012     /* PTF      FOR HAA1202 RECEIVED                    */
 /* AZ00013        APAR     FOR HAA1202 NOT RECEIVED                */
 /* TZ00001        USERMOD  FOR HAA1202 RECEIVED                    */
    UZ00002     /* PTF      FOR HBB1202 RECEIVED                    */
              )
 GROUP
 BYPASS(HOLDSYSTEM)
 CHECK.
```

After getting the SYSMOD Comparison report, you can take these actions:

1. Research the report to determine which of the identified SYSMODs you want to install into the comparison zone.
2. Find and receive any applicable SYSMODs that were not available and that you want to install. The source ID in the report identifies some possible sources for obtaining the SYSMODs.
3. Tailor the SMPPUNCH output to install the set of SYSMODs that you deem appropriate, and run the commands to install the desired SYSMODs.

Example 2: Using REPORT SYSMODS with zones controlled by different global zones

Assume that you have the same set up as in Example 1, except that TGZONE1 is controlled by a different global zone than TGZONE2. That is, target zone TGZONE1 is controlled by global zone SYS1.GLOBAL1.CSI, and target zone TGZONE2 is controlled by global zone SYS1.GLOBAL2.CSI. Further assume that the functions and PTFs contained in each zone are the same as those in Table 22 on page 329.

Also assume that SMPCSI is allocated to SYS1.GLOBAL1.CSI and you want to find out which sysmods are installed in TGZONE1 and are not installed in TGZONE2. You can use the following commands:

```
SET        BDY(GLOBAL)          /* Process global zone.    */.
REPORT     SYSMODS              /* Report on SYSMODs       */
           INZONE(TGZONE1)      /* Input zone TGZONE2      */
           COMPAREDTO(SYS1.GLOBAL2.CSI,TGZONE2)           */
                                /* Comparison zone TGZONE2 */.
```

SMP/E compares the SYSMOD content of zone TGZONE1 to that of zone TGZONE2. Any SYSMODs that are in zone TGZONE1 (with the exceptions noted in "Processing" on page 333) and are not in zone TGZONE2 appear in the resulting report.

Figure 30 on page 332 shows the report produced by SMP/E. The report includes the global zone CSI data set name that was specified on the COMPAREDTO operand.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPLIST
OUTPUT

SYSMOD COMPARISON REPORT FOR TARGET ZONE TGZONE1 AND TARGET ZONE TGZONE2
IN THE GLOBAL ZONE IN CSI SYS1.GLOBAL2.CSI

MATCHING SREL(S) - Z038

FMID___  SYSMOD_  TYPE____  APPLICABLE  RECEIVED  SOURCEIDS

HAA1202  HAA1202  FUNCTION  YES         YES
         UZ00011  PTF       YES         NO        PUT0706
         UZ00012  PTF       YES         YES       PUT0707
         AZ00013  APAR      YES         NO        RETAIN
         TZ00001  USERMOD   YES         YES
HBB1202  UZ00002  PTF       YES         YES       PUT0707
                                                   PDO0001
JBB1222  UZ00021  PTF       NO          NO        PUT0706
         UZ00022  PTF       NO          YES       PUT0707
```

Figure 30. Example of a SYSMOD comparison report

If a SYSMOD is identified in the SYSMOD Comparison report, and SYSTEM or USER HOLDDATA entries exist for that SYSMOD in the global zone for the input zone, those HOLDs are reported in the SYSMOD Comparison HOLDDATA report. See "SYSMOD comparison HOLDDATA report" on page 523 for an example of the report.

SMP/E also writes the commands shown in the code example below to the SMPPUNCH data set. Since the INZONE and COMPAREDTO zones are defined in different global CSI data sets, the SMPCSI and SMPCNTL DD statements are included in the SMPPUNCH output. These DD statements must be copied to your job before the output is used.

```
//* DD statement for zone TGZONE2  */
//SMPCSI   DD   DSN=SYS1.GLOBAL2.CSI
//SMPCNTL  DD  *
SET BDY (TGZONE2).
RESETRC        /*

 THE  FOLLOWING APPLY  COMMAND(S) WERE  GENERATED BY A REPORT SYSMODS
 COMMAND ON yy.ddd AT hh:mm:ss.  THE  SELECTED SYSMODS WERE  FOUND IN
 THE INPUT  ZONE (TGZONE1) BUT  NOT IN THE  COMPARISON ZONE (TGZONE2)
 AND APPEAR APPLICABLE TO THE COMPARISON ZONE.

 SYSMODS IN THE  SELECT LIST ARE  GROUPED BY FMID.   FUNCTIONS APPEAR
```

```
  FIRST FOLLOWED BY SERVICE SYSMODS (PTFS, APARS, AND USERMODS IN THAT
  ORDER).

  FUNCTIONS AND  PTFS THAT WERE  AVAILABLE IN THE  GLOBAL ZONE AND ARE
  APPLICABLE  TO THE  COMPARISON ZONE  APPEAR AS  NORMAL  SELECT  LIST
  VALUES WITH AN  APPROPRIATE COMMENT.   FUNCTIONS AND  PTFS THAT WERE
  NOT AVAILABLE  IN THE  GLOBAL  ZONE OR  HAVE  UNKNOWN  APPLICABILITY
  APPEAR  AS SMP  COMMENTS.   PTFS  FOR AN  APPLICABLE  FUNCTION,  NOT
  AVAILABLE IN THE  GLOBAL ZONE,  APPEAR AS COMMENTS  EVEN IF THEY ARE
  AVAILABLE.   SUCH COMMENTED SYSMODS CAN BE  CHANGED TO NORMAL SELECT
  LIST VALUES WHEN THEY OR THEIR APPLICABLE FUNCTIONS ARE RECEIVED, OR
  ARE DETERMINED TO BE APPLICABLE.

  APARS AND USERMODS APPEAR AS SMP COMMENTS.   THESE COMMENTED SYSMODS
  CAN  BECOME NORMAL  SELECT LIST VALUES IF YOU DESIRE TO INSTALL THEM
  IN THE TGZONE2 ZONE AND THEY ARE AVAILABLE IN THE GLOBAL ZONE.   THE
  COMMENTARY  TEXT  INDICATES  THEIR  AVAILABILITY  AND  APPLICABILITY
  STATUS.

  THE COMMAND(S)  WERE  GENERATED  WITH THE BYPASS(HOLDSYSTEM), GROUP,
  AND CHECK OPTIONS.  GROUP CAUSES REQUISITE  SYSMODS  TO BE  INCLUDED
  ALONG  WITH THE  SELECTED SYSMODS.  CHECK  CAUSES  A  DRYRUN OF  THE
  COMMAND BEFORE LIBRARIES ARE  UPDATED SO THAT  ANY NEEDED CORRECTIVE
  ACTION CAN BE  TAKEN BEFORE  REAL INSTALLATION.   BYPASS(HOLDSYSTEM)
  CAUSES ALL SYSTEM HOLD REASON IDS TO BE BYPASSED.

  REDO WAS GENERATED   IF  A  FUNCTION IS TO  BE REINSTALLED.  If REDO
  IS  USED, ONLY  THE FUNCTION BEING  REINSTALLED AND ITS  SERVICE ARE
  SELECTED ON THAT COMMAND.

                       **** WARNING ****

  IT IS   POSSIBLE  THAT  ALL  SYSMODS  SELECTED  WILL  APPEAR  AS SMP
  COMMENTS.  IF THIS IS THE CASE, SMP WILL ISSUE A SYNTAX ERROR IF YOU
  RUN THE GENERATED COMMAND(S) AS IS.

                       **** WARNING ****
                                                                    */
            .
```

```
APPLY
 SELECT(
    HAA1202     /* FUNCTION FOR HAA1202 RECEIVED                    */
 /* UZ00011        PTF      FOR HAA1202 NOT RECEIVED                */
    UZ00012     /* PTF      FOR HAA1202 RECEIVED                    */
 /* AZ00013        APAR     FOR HAA1202 NOT RECEIVED                */
 /* TZ00001        USERMOD  FOR HAA1202 RECEIVED                    */
    UZ00002     /* PTF      FOR HBB1202 RECEIVED                    */
              )
 GROUP
 BYPASS(HOLDSYSTEM)
 CHECK.
/*
```

Processing

The REPORT SYSMODS command compares the SYSMOD content of an input target or distribution zone to that of a comparison target or distribution zone. The resulting output can be used to determine which SYSMODs might need to be installed in the comparison zone to make its function and service content more equivalent to that of the input zone.

SMP/E first checks the REPORT SYSMODS command to determine the zones to be compared and whether SMPPUNCH output should be produced. The zones are then opened for read access along with the global zone. If NOPUNCH was **not** specified, the SMPPUNCH data set is also opened.

Next, SMP/E determines the matching SREL values from the zone definitions of the zones being compared. Matching values are saved to be put in the header of the SYSMOD Comparison report.

SMP/E then determines whether or not the zones to be compared contain SYSMOD entries. If they both contain SYSMOD entries, SMP/E looks for SYSMODs that are installed in the input zone but not in the comparison zone, and checks whether those SYSMODs are applicable to the comparison zone.

1. First, SMP/E reads sequentially through the SYSMOD entries in the input zone to determine which SYSMODs should be included in the report. For each SYSMOD entry that is **not** a superseded-only entry

or deleted-only entry and is not in error, SMP/E checks the comparison zone to see if the SYSMOD also exists there. The SYSMOD exists in the comparison zone if **one** of the following is true:

- There is a SYSMOD entry with the ERROR indicator off.
- There is a SYSMOD entry with the DELBY subentry.
- There is a SYSMOD entry with the SUPBY subentry.

If the SYSMOD does **not** exist in the comparison zone, SMP/E includes it in the SYSMOD Comparison report.

If the SYSMOD **exists** in the comparison zone and is a function, it may still be included in the SYSMOD Comparison report. This can happen when the function in the input zone has been service-updated. A service-updated function may be at a higher service level than the function currently installed in the comparison zone. Therefore, SMP/E does additional checking for a function that exists in both the input and comparison zones.

If a function in the input zone has been service-updated, it supersedes the service integrated into it. To determine whether a service-updated function should be reinstalled in the comparison zone, SMP/E checks whether all the SYSMODs superseded by the function exist in the comparison zone. If any of the superseded SYSMODs do **not** exist in the comparison zone, the service-updated function can be reinstalled and is included in the report.

Note: When SMP/E checks the global zone to determine whether a service-update is available to be reinstalled, it also checks the superseded information in the global zone SYSMOD entry against the input zone SYSMOD entry. This is done to ensure that the function in the global zone is not at a lower service level than the SYSMOD in the input zone.

2. Next, SMP/E determines whether the SYSMODs included in the report are applicable to the comparison zone.
   - If the SYSMOD is a service-updated function that can be reinstalled, it is applicable to the comparison zone.
   - If the SYSMOD is a base function, it is applicable to the comparison zone if it meets **either** of these conditions:
     - All the SRELs supported by the input zone are also supported by the comparison zone.
     - Any of the SRELs defined for the SYSMOD are supported by the comparison zone.

       Note: SMP/E checks the global zone SYSMOD entry to determine which SRELs are defined for the SYSMOD. If there is no entry for the SYSMOD in the global zone, SMP/E cannot determine the SYSMOD's SREL.

       If there is at least one SREL in the input zone that is **not** in the comparison zone and there is no SYSMOD entry in the global zone, the SYSMOD **may** or **may not** be applicable to the comparison zone. In this case, SMP/E indicates in the report that the applicability of the SYSMOD is unknown.
   - If the SYSMOD is not a base function, it is applicable to the comparison zone if its FMID meets **either** of these conditions:
     - It exists in the comparison zone.
     - It has already been deemed applicable during this run of the REPORT SYSMODS command. (If the applicability of the FMID is unknown, the applicability of this SYSMOD is also unknown.)

At this point, SMP/E gathers additional information about the SYSMOD to include in the SYSMOD Comparison report:

- It takes the FMID, SYSMOD type (function, PTF, APAR, or USERMOD), and source IDs from the SYSMOD entry in the input zone.
- It checks whether there is an entry for the SYSMOD in the global zone.

SMP/E saves all the information gathered in the preceding processing and uses it in the SYSMOD Comparison report and SMPPUNCH output after checking all the SYSMOD entries in the input zone.

If a SYSMOD is identified in the SYSMOD Comparison Report, and SYSTEM or USER HOLDDATA entries exist for that SYSMOD in the global zone for the input zone, those HOLDs are reported in the SYSMOD Comparison HOLDDATA Report.

Internal SYSTEM HOLDDATA may be associated with more than one SYSMOD. Therefore, internal SYSTEM HOLDDATA will appear in the SYSMOD Comparison HOLDDATA Report only if the originating SYSMOD does not exist in the comparison zone. The originating SYSMOD exists in the comparison zone if one of the following is true:

- There is a SYSMOD entry with the ERROR indicator off.
- There is a SYSMOD entry with the DELBY subentry.
- There is a SYSMOD entry with the SUPBY subentry.

It is possible for an internal SYSTEM HOLD to appear in the SYSMOD Comparison HOLDDATA Report even though the originating SYSMOD does not appear in the SYSMOD Comparison Report. This situation would occur if the originating SYSMOD is a superseded-only SYSMOD in the input zone and it does not exist in the comparison zone but one of its superseding SYSMODs (which also contains the internal SYSTEM HOLD) is reported in the SYSMOD Comparison Report.

Likewise, it is possible for a SYSMOD to appear in the SYSMOD Comparison Report, but its internal SYSTEM HOLDs are not reported in the SYSMOD Comparison HOLDDATA Report. This situation would occur when the SYSMOD supersedes the originating SYSMOD for an internal SYSTEM HOLD, but the originating SYSMOD already exists in the comparison zone.

SMP/E issues an error message, and REPORT SYSMODS processing fails if any of the following occurs:

- The zone specified on the INZONE or COMPAREDTO operand is not defined in the global zone.
- The zones specified on the INZONE and COMPAREDTO operands have no matching SREL values in their zone definitions.
- The zone specified on the INZONE or COMPAREDTO operand is the global zone.
- The zone specified on the INZONE or COMPAREDTO operand contains no SYSMOD entries.
- The zones specified on the INZONE and COMPAREDTO operands are the same.
- NOPUNCH was not specified, but there is no definition for the SMPPUNCH data set.

Zone and data set sharing considerations

The following identifies the phases of REPORT SYSMODS processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

Global zone (multiple as required)	Read without enqueue.
Target zones (as required)	Read without enqueue.
DLIB zones (as required)	Read without enqueue.

2. Processing

Global zone (multiple as required)	Read with shared enqueue.
Target zones (as required)	Read with shared enqueue.
DLIB zones (as required)	Update with exclusive enqueue.

3. Termination

   All resources are freed.

Chapter 21. The RESETRC command

Many SMP/E commands depend on the successful processing of preceding commands. To help avoid errors resulting from the failure of preceding commands, SMP/E saves the highest return code issued for each command. As it processes each command, it checks these return codes to make sure all the preceding commands succeeded.

At times, you may want to process commands that do not depend on the success of any preceding commands. To do this, you can use the RESETRC command. RESETRC sets the return codes for the preceding commands to zero, thus allowing SMP/E to process the current command.

Zones for SET BOUNDARY

The RESETRC command is used only to determine whether subsequent commands are to be processed. Therefore, you should use the same SET BOUNDARY command for RESETRC as for the subsequent commands.

Syntax

Data sets used

The following data sets might be needed to run the RESETRC command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA		

Usage notes

- You should **not** use RESETRC before commands that depend on the success of preceding commands.
- RESETRC only resets return codes issued for SMP/E commands. It does not affect the maximum return code set when SMP/E itself fails. That return code is always set to the highest return code issued for any command processed during that calling of SMP/E.
- There is no return code for the RESETRC command.

Examples

The following examples are provided to help you use the RESETRC command.

Example 1: Using RESETRC between commands for one zone

The processing of one command may **not** depend on the success of preceding commands if you divide the work that is to be done to a single zone into multiple steps. For example, you might use the source ID and FMIDSET operands to apply SYSMODs for two different functions. In this example, you want to install two groups of PTFs from service level PUT0701. One group is for function EBB1102, and the other is for

an unrelated function, EDM1102. To prevent possible errors from one APPLY command from affecting the processing of the other, you can put a RESETRC command between the two APPLY commands:

```
SET      BDY(MVSTST1)       /* Process MVSTST1 tgt zone.*/.
APPLY    SOURCEID(PUT0701)  /* Apply service level 0701 */
         FORFMID(EBB1102)   /* for function EBB1102.    */.
RESETRC                     /* Next set of commands
                               not dependent on
                               successful apply of
                               EBB1102 service.         */.
APPLY    SOURCEID(PUT0701)  /* Now apply service for    */
         FORFMID(EDM1102)   /* EDM1102.                 */.
```

SMP/E first applies PTFs from service level 0701 that are for function EBB1102. It then applies PTFs from service level 0701 that are for function EDM1102. Without the intervening RESETRC command, the second APPLY runs only if the return code for the first APPLY was less than 12.

Example 2: Using RESETRC between commands for different zones

You might also want to use the RESETRC command when you are installing the same changes on different systems. If the two systems are not identical, errors that might occur during installation on one system may not occur during installation on the other system. For example, Assume that you want to install two PTFs that are applicable to two different systems, MVSTST1 and MVSTST2. To prevent possible errors from the SET or APPLY commands for one system from affecting processing in the other system, you can put a RESETRC command between the two groups of SET and APPLY commands:

```
SET      BDY(MVSTST1)       /* Set to process MVSTST1.  */.
APPLY    S(UR12345,UR12346) /* Apply two PTFs.          */.
RESETRC                     /* Next set of commands is
                               not dependent on successful
                               apply of two PTFs.       */.
SET      BDY(MVSTST2)       /* Set to process MVSTST2.  */.
APPLY    S(UR12345,UR12346) /* Apply two PTFs.          */.
```

SMP/E first applies the two PTFs to MVSTST1, and then applies the same PTFs to MVSTST2. Without the intervening RESETRC command, the second group of SET and APPLY commands runs only if the return code for the first APPLY was less than 12.

Processing

For each command processed in a single invocation of SMP/E, SMP/E keeps a record of the highest return code for that command. As SMP/E processes each command, it checks the saved return codes from all the preceding commands to determine whether it should process the current command.

When the RESETRC command is processed, SMP/E resets the return codes for the preceding commands to 0. This allows the next command after the RESETRC command to be processed, regardless of whether preceding commands succeeded.

Chapter 22. The RESTORE command

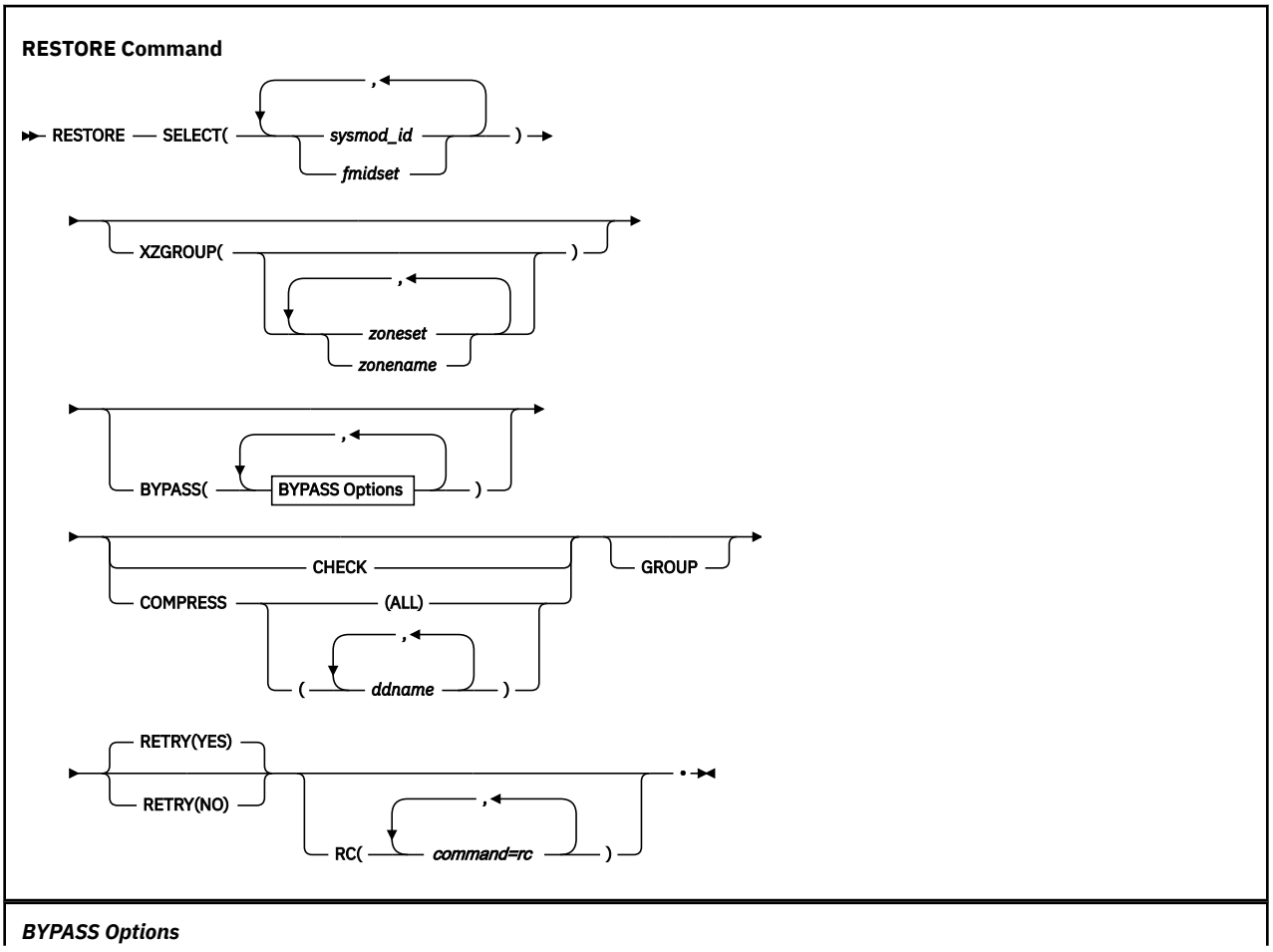
At times, you may want to remove a SYSMOD that has been applied to the target libraries. For example, perhaps you installed a fix for a problem and got unexpected results. If you have not yet accepted the SYSMOD into the distribution libraries, you can use the RESTORE command to remove it from the target libraries.

The RESTORE command replaces the affected elements in the target libraries with the unchanged versions from the distribution libraries. (As a result, once you have accepted a SYSMOD into the distribution libraries, you cannot use RESTORE to remove it from the target libraries.)

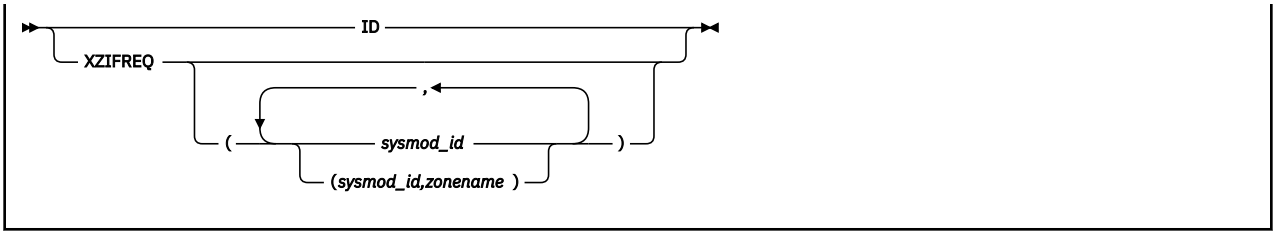
Zones for SET BOUNDARY

For the RESTORE command, the SET BOUNDARY command must specify the target zone from which the SYSMODs should be removed. SMP/E uses the RELATED field in the TARGETZONE entry to determine which distribution zone describes the elements to replace the restored elements.

Syntax



BYPASS Options



Operands

BYPASS
You can specify any of these options:

ID
XZIFREQ
XZIFREQ(*list*)

BYPASS(ID)
indicates that SMP/E should ignore any errors it detects when checking RMID and UMIDs for element entries in the target zone or distribution zone.

BYPASS(XZIFREQ)
indicates that SMP/E should continue RESTORE processing for a SYSMOD even if a SYSMOD in another zone names the SYSMOD being restored as a requisite, regardless of which or how many SYSMODs state the requisite condition or what zones they are in. SMP/E will identify such requisite conditions with a warning message, instead of terminating the RESTORE processing.

Note: CIFREQ conditions within the set-to zone cannot be bypassed.

BYPASS(XZIFREQ(*list*))
indicates that SMP/E should continue RESTORE processing for a SYSMOD even if a SYSMOD in another zone names the SYSMOD being restored as a requisite, provided that the SYSMOD stating the requisite condition is included in the list provided with the XZIFREQ option. For causer SYSMODs identified in the list, SMP/E will identify such requisite conditions with a warning message.

Each entry in the list must be in one of the following formats:

- *sysmod_id*
- (*sysmod_id*,*zone*)

sysmod_id
specifies that any requisite condition stated by SYSMOD *sysmod_id* in any zone (other than the set-to zone) is not to be considered an error condition.

(sysmod_id,zone)
specifies that any requisite condition stated by SYSMOD *sysmod_id* in zone *zone* is not to be considered an error condition.

Each entry in the list must be unique. Also, a SYSMOD ID must not appear both by itself and as part of a SYSMOD/zone pair. However, a SYSMOD ID may appear in multiple SYSMOD/zone pairs, provided each of the pairs is unique.

The list provided must not be a null list; that is, BYPASS(XZIFREQ()) is not allowed.

Note:

1. CIFREQ conditions within the set-to zone cannot be bypassed.
2. If a SYSMOD that is not on the BYPASS XZIFREQ list has stated a requisite condition for the SYSMOD being restored, SMP/E terminates the RESTORE processing.

CHECK
indicates that SMP/E should not actually update any libraries. Instead, it should just take these actions:

- Test for errors other than those that can occur when the libraries are actually updated
- Report on which libraries are affected
- Report on any SYSMOD that would be regressed

COMPRESS
indicates which target libraries should be compressed. SMP/E does **not** compress any libraries that are actually paths in a UNIX file system.

- If you specify ALL, any libraries containing elements that will be updated by this RESTORE command are compressed.
- If you specify particular ddnames, those libraries are compressed regardless of whether they will be updated.

Note:

1. COMPRESS can also be specified as C.
2. If you specify COMPRESS and CHECK, COMPRESS is ignored, because SMP/E does not update any data sets for CHECK.

GROUP
indicates that if SMP/E determines that additional SYSMODs should be restored, other than those specified in the SELECT list, SMP/E should automatically include them.

For example, Assume that you have applied a function and service for that function. When you select the function and specify the GROUP operand, SMP/E also tries to restore the service that was applied for that function.

Likewise, Assume that you have applied two PTFs, and one defines the other as the prerequisite. When you select the prerequisite and specify the GROUP operand, SMP/E also tries to restore the other PTF. On the other hand, if you select the SYSMOD that specifies the prerequisite, SMP/E restores that particular SYSMOD **only** if the prerequisite has been accepted.

However, Assume that you have installed two PTFs that affect the same element but that do not define any relationship to each other. If you select one of the PTFs and specify the GROUP operand, SMP/E does **not** try to restore the other PTF. You have to specify both PTFs on the SELECT operand.

Note: GROUP can also be specified as G.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the RESTORE command.

Before SMP/E processes the RESTORE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the RESTORE command. Otherwise, the RESTORE command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the RESTORE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RETRY
indicates whether SMP/E should try to recover from out-of-space errors for utilities it calls.

YES
indicates that SMP/E should try to recover and retry the utility if a RETRYDDN list is available in the OPTIONS entry that is in effect. RETRY(YES) is the default.

If retry processing does not reclaim sufficient space and input to the utility was batched (copy or link-edit utility only), SMP/E debatches the input and retries the utility for each member separately. If this final attempt fails, the resulting *x*37 abend is treated as an unacceptable utility

return code. In this case, processing continues for SYSMODs containing eligible updates to other libraries, but processing fails for SYSMODs containing unprocessed elements for the out-of-space library (and it fails for any SYSMODs that are dependent on the failed SYSMODs). For guidance on setting up the desired retry processing, see *z/OS SMP/E User's Guide*. For more information about OPTIONS entries, see *z/OS SMP/E Reference*.

If there is no RETRYDDN list, SMP/E does not try to recover from out-of-space errors, even if RETRY(YES) is specified.

NO
indicates that SMP/E should not try to recover from the error.

SELECT
specifies one or more SYSMODs that should be restored.

You may specify any combination of individual SYSMOD IDs and FMIDSET names, provided that there are no duplicate SYSMOD IDs nor any duplicate FMIDSET names. For each FMIDSET specified, all FMIDs defined in the FMIDSET are processed as if they were explicitly specified in the SELECT list.

Note:

1. SELECT is required for RESTORE. This is the only means of specifying which SYSMODs are eligible to be restored.
2. SELECT can also be specified as S.
3. If you use GROUP along with SELECT, make sure to specify the lowest level of service you want restored. For example, if you want to restore PTF1 and PTF2, and PTF1 is a prerequisite for PTF2, specify PTF1 on the SELECT operand.
4. When using FMIDSETs on the SELECT operand, remember that:
   - A value specified in the SELECT list is processed as an FMIDSET if the GLOBAL zone contains an FMIDSET entry by that name.
   - A value specified in the SELECT list is processed as a SYSMOD ID if it is not defined as an FMIDSET in the GLOBAL zone and it is a valid SYSMOD ID.
   - If the value in the SELECT list is valid both as a SYSMOD ID and as an FMIDSET name, it is processed (for SELECT) as an FMIDSET. If you want to select a SYSMOD that has the same name as an FMIDSET, you must define that SYSMOD in an FMIDSET and then include that FMIDSET name in the SELECT list.
   - Any given value (whether it represents a SYSMOD ID, an FMIDSET, or both) may **not** appear more than once in the SELECT list.
   - A SYSMOD ID may be explicitly specified in the SELECT list and also included in an FMIDSET that is also specified in the SELECT list, provided the SYSMOD ID does not have the same name as the FMIDSET. The duplicate SYSMOD ID is ignored.

XZGROUP
indicates that SMP/E's default method for determining the zones to be checked for cross-zone requisites is being overridden. You may specify a list of ZONESETs or zones (or both) that are to be used to establish the zone group for this command execution. Each value in the list must be 1 to 8 alphanumeric or national (@, #, and $) characters. XZGROUP() – a null list – may be specified, which means that SMP/E is to do no cross-zone requisite checking.

Note:

1. If XZGROUP is specified, whatever ZONESETs the user specifies are used to establish the initial zone group, even if the set-to zone is not in a ZONESET and the XZREQCHK subentry is not set.
2. If no XZGROUP operand was specified on the RESTORE command, SMP/E reads all ZONESET entries. If a ZONESET entry has its XZREQCHK subentry set to YES and it contains the set-to zone, then all the other zones within the ZONESET entry become part of the initial zone group for the RESTORE command.

3. After the initial zone group is established, it is culled by removing all distribution zone for RESTORE processing. In other words, only zones having the same type as the set-to zone are left in the final zone group used for cross-zone requisite checking.

Data sets used

The following data sets might be needed to run the RESTORE command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

Distribution library	SMPLOGA	SMPSNAP	SYSPRINT
Link library	SMPLTS	SMPSTS	SYSUT1
SMPCNTL	SMPMTS	SMPTLIB	SYSUT2
SMPCSI	SMPOUT	SMPWRK1	SYSUT3
SMPDATA1	SMPPARM	SMPWRK2	SYSUT4
SMPDATA2	SMPPTS	SMPWRK3	Target library
SMPJHOME	SMPRPT	SMPWRK4	Text library
SMPLOG	SMPSCDS	SYSLIB	*zone*

Note:

1. SMPJHOME is an optional ddname used to specify the Java runtime directory. SMP/E requires the Java runtime when a UNIX shell script is invoked during the restore of a file system element, and this UNIX shell script issues a Java command.
2. The SMPLTS data set is required only when a load module with CALLLIBS is being processed.
3. The SMPDATA1 and SMPDATA2 data sets are required only when the CHANGEFILE subentry of the active OPTIONS entry is set to "YES" for the target zone that RESTORE is operating on.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
5. SMPPARM is only required if exit routines have been defined in SMPPARM member GIMEXITS.

Usage notes

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

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

  – An element being restored has a MODID in the element entry on the distribution zone that does not have a corresponding SYSMOD entry on the target zone. This condition 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 service level of an element being restored is the same in the target library as it is in the distribution library. This condition 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, without being accepted, and the later modifications are the ones that are selected for RESTORE processing.

Consider the following example. The distribution zone shows that an element was last replaced on the distribution libraries by PTF UZ00001, but the related target zone indicates that the last replacement to the element on the system was by PTF UZ00004. The element was also modified on the system by PTFs UZ00002 and UZ00003. The SYSMODs on the related target zone and distribution zone are listed in service order:

TARGET ZONE SYSMODs
DLIB ZONE SYSMODs

UZ00001
UZ00001

UZ00002

UZ00003

UZ00004

If you specified the following, PTFs UZ00002 and UZ00003 would not be considered part of the RESTORE processing group because they are not dependent on PTF UZ00004.

```
SET      BDY(TGT1)        /* Set to target zone.*/.
RESTORE  S(UZ00004)       /*                    */
         GROUP            /*                    */.
```

To correct the error, specify:

```
SET      BDY(TGT1)        /* Set to target zone.*/.
RESTORE  S(UZ00002)       /*                    */
         GROUP            /*                    */.
```

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

– The ineligibility of a member of a RESTORE group terminates processing for the entire group. This can occur both in GROUP and SELECT mode.
– A function SYSMOD containing a ++VER DELETE MCS cannot be restored if any of the specified SYSMODs were actually deleted when the function was applied. (Such a function **is** eligible for RESTORE processing if none of the specified SYSMODs had ever been applied, and were, therefore, not deleted when the function was installed.)

  Function SYSMODs containing a ++DELETE statement for a load module are not eligible for RESTORE processing.

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

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

  The first two conditions described earlier in the first special consideration (SYSMOD ineligibility) can be bypassed by using this option. However, in the first case, the distribution library contains a version of the element that is probably functionally superior to the version being removed. This can cause the executable code in the target system library to be inoperable. In addition, SMP/E updates the element entry on the target zone to reflect the UMID and RMID subentry contents from the element entry on the distribution zone. In this case, the SYSMOD entry might not exist on the target zone, because the BYPASS(APPLYCHECK) operand was probably used on the ACCEPT command; thus, the SYSMOD was never applied to the target system. You should avoid using the BYPASS(ID) option unless it is absolutely necessary.
- Utility failures can cause the RESTORE command to fail. For details on handling *x*37 abends, see the description of the RETRY operand under “Operands” on page 340.
- SYSMOD entries on the target zone have the ERROR and RESTORE status indicators set on before the target system libraries are updated. If processing fails during the updating, these indicators 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 command.

- RESTORE processing relinks the nucleus, using the last version of modules accepted on the DLIBs.
- When a selected SYSMOD contains an element that was deleted from the system by that SYSMOD, RESTORE processing reintroduces that element into the target system using information saved in the SMPSCDS BACKUP entries.
- If you do not use SMP/E to recover after a failure and choose the option of restoring your system and the distribution libraries by means of system and DLIB RESTORE tapes, you must ensure that the SMPPTS, SMPCSI, SMPSCDS, SMPMTS, and SMPSTS data sets are also restored to their previous levels.
- The exception SYSMOD data stored in the global zone SYSMOD entry is not purged when the SYSMOD is restored. If NOREJECT is not set in the OPTIONS entry that is in effect, the global zone entry is purged of all information except the exception SYSMOD data. (Having NOREJECT set off is the default.)

Output

The following reports may be produced during RESTORE processing:

- Causer SYSMOD Summary report
- Cross-Zone Summary report
- File Allocation report
- Element Summary report
- MOVE/RENAME/DELETE report
- SYSMOD Status report

See Chapter 34, “SMP/E reports,” on page 457 for descriptions of these reports.

RESTORE processing may also create library change file records that reflect any successful utility work performed by RESTORE processing to update target libraries. For more information on library change file recording, see *z/OS SMP/E Reference*.

Examples

In each of these examples, the following set of SYSMODs has been received:

```
++PTF(UZ00001)              /* 1st PTF in chain 1.      */.
++VER(Z038) FMID(FXY1040)   /* No prerequisites.        */.
++MOD(XYMOD01)              /*                          */.

++PTF(UZ00002)              /* 2nd PTF in chain 1.      */.
++VER(Z038) FMID(FXY1040)   /*                          */
            PRE(UZ00001)    /* Previous PTF.            */.
++MOD(XYMOD01)              /*                          */.

++PTF(UZ00003)              /* 3rd PTF in chain 1.      */.
++VER(Z038) FMID(FXY1040)   /*                          */
            PRE(UZ00002)    /* Previous PTF.            */.
++MOD(XYMOD01)              /*                          */.

++PTF(UZ00010)              /* 1st PTF in chain 2.      */.
++VER(Z038) FMID(FXY1040)   /*                          */
            PRE(UZ00001)    /* Logical requisite.       */.
++MOD(XYMOD02)              /*                          */.
```

Note: For these examples, assume (1) all modules are present in the target zone and distribution zone, with the result that the DISTLIB operand is not required; and (2) the actual module replacement follows the ++MOD statement.

The following examples are provided to help you use the RESTORE

Example 1: Restoring a single SYSMOD

Assume that you have applied only PTF UZ00001, that an error was detected during testing, and that you want to remove the PTF from your system. The following RESTORE command can be used:

```
SET      BDY(TGT1)          /* Set to target zone.      */.
RESTORE  S(UZ00001)         /* Restore 1 PTF.           */.
```

If you want to clean up all of SMP/E's records for this PTF (the global zone and the SMPPTS data set), you can use the REJECT command after RESTORE processing is complete:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
REJECT   S(UZ00001)         /* Reject 1 PTF.            */.
```

Example 2: Restoring multiple PTFs to remove one PTF

Assume that you have applied all PTFs UZ00001, UZ00002, and UZ00003 to your system, and that during testing an error is found in module XYMOD01. Because the current service level of that module is UZ00003, you want to restore that PTF from the system.

You now have two choices:

1. Restore PTF UZ00001, UZ00002, UZ00003, and then reapply UZ00001 and UZ00002 as follows:

   ```
   SET      BDY(TGT1)          /* Set to target zone.   */.
   RESTORE  S(UZ00001,         /* Restore all 3 PTFs.   */
              UZ00002,         /*                       */
              UZ00003)         /*                       */.
   APPLY    S(UZ00001          /* Then re-apply the two */
              UZ00002)         /* that may be okay.     */.
   ```

2. Accept PTFs UZ00001 and UZ00002, if you are sure that they have no errors, then restore UZ00003 as follows:

   ```
   SET      BDY(DLIB1)         /* Set to DLIB zone.     */.
   ACCEPT   S(UZ00001,         /* Accept two good PTFs. */
              UZ00002)         /*                       */.
   SET      BDY(TGT1)          /* Set to target zone.   */.
   RESTORE  S(UZ00003)         /* Restore the 1 bad PTF.*/.
   ```

The end result in both cases is that module XYMOD01 from PTF UZ00002 is in the target libraries.

Example 3: Restoring PTFs using the GROUP operand

In Example 2, when you wanted to restore the three PTFs, you specified all three in the select list. In a simple case like this, that was very easy; in practice, however, many PTFs are related to one another, and it may not be easy to determine which PTFs must be restored in order to remove the bad one. The GROUP operand can be used to assist in determining this. The following commands can be run to determine which PTFs must be restored to restore UZ00003:

```
SET      BDY(TGT1)          /* Set to target zone.      */.
RESTORE  S(UZ00003)         /* Restore this one PTF     */
         GROUP              /* plus any related PTFs,   */
         CHECK              /* in check mode this time. */.
```

After running these commands, the various SMP/E reports can be used to determine that PTFs UZ00001, UZ00002, and UZ00003 should be restored. You can then determine the correct action: restore all, or accept some and then restore.

Processing

This section describes the following topics:

- Selecting SYSMODs
- Installing elements

- Recording After completion
- Cross-zone processing
- Global zone SYSMOD entries

SYSMOD selection

This section describes how SMP/E selects SYSMODs.

Operands related to SYSMOD selection

You can use the following operands to tell SMP/E which SYSMODs are to be restored:

- SELECT
- GROUP

SELECT tells SMP/E which particular SYSMODs are to be restored. GROUP, while having an effect on SYSMOD selection, does not directly indicate which SYSMODs are affected.

Candidate selection

When the RESTORE command is encountered, SMP/E looks at each SYSMOD specified in the SELECT list to make sure the following conditions hold:

- The SYSMOD has been applied to the target zone specified in the previous SET command.
- The SYSMOD has not been accepted to the distribution zone specified in the RELATED field of the TARGETZONE entry for the target zone specified in the previous SET command.

If any SYSMODs are found that do not meet both of these conditions, error messages are written to the SMPLOG and SMPOUT, and those SYSMODs are not considered eligible to be restored.

Once the SYSMODs specified in the SELECT list have been checked for initial eligibility, SMP/E checks to see whether any other SYSMODs are affected. There are two ways other SYSMODs can be affected:

- SYSMODs can be related to one another through the PRE, REQ, FMID, and SUP operands of their ++VER statement or the REQ operand of the ++IF statement.

  For each candidate SYSMOD, SMP/E checks to see whether dependencies exist between it and any other SYSMOD not yet accepted. (That is, does any PRE, REQ, IFREQ, or FMID relationship exist between the candidate SYSMOD and these other SYSMODs?) If so, that SYSMOD must also be restored. This is true because of the stated dependency on either the functional or service dependency of the SYSMODs.
- SYSMODs can be related to one another, because they have elements in common.

  For each candidate SYSMOD, SMP/E checks to see if any other SYSMOD, not yet accepted, has modules in common with the candidate SYSMOD. If so, that SYSMOD must also be restored. This is true because of the method used to replace the elements on the system libraries. The version of the element in the distribution library is used as the backup. Thus, all SYSMODs that have replaced or modified the elements since the distribution library version was accepted must also be removed.

Processing of these related SYSMODs depends on whether the GROUP operand was specified:

- If the GROUP operand was specified, each of the related SYSMODs, as previously identified, are included as candidates for RESTORE. SMP/E then performs the same checking on these new candidates as on the original set. This process continues until no additional SYSMODs are added.
- If the GROUP operand was not specified, SMP/E issues an error message to SMPOUT and SMPLOG indicating which of the SYSMODs specified in the SELECT list cannot be restored. This information can also be found in the RESTORE SYSMOD Status report.

Element installation

Once the proper SYSMODs have been selected, SMP/E moves the version of the element in the distribution libraries to the proper places in the system libraries. Processing for each type of elements is described in subsequent sections.

Inline JCLIN

If a SYSMOD that had inline JCLIN is restored, SMP/E attempts to restore the target zone entries affected by the JCLIN to the state they were in before the SYSMOD was applied. This is done by accessing the BACKUP entry for such SYSMODs. For each BACKUP entry, SMP/E checks the corresponding target zone entry to ensure that the last modification (LASTUPD subentry) to the target zone entry was for the SYSMOD being restored. If it was, the entry is replaced from the BACKUP entry. If it was not, SMP/E issues messages to indicate that the SYSMOD was not restored, and RESTORE processing stops for that SYSMOD. 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 should occur only for LMOD entries.

Note: RESTORE processing is limited for a SYSMOD using the CHANGE statement in inline JCLIN. When that SYSMOD is restored, the backup copy of the LMOD entry (which does not have the updates from the CHANGE statement) replaces the target zone LMOD entry, and the information from the CHANGE statement is lost. Module names that were changed by the inline JCLIN remain in the load module under their changed names.

As each entry is completed, SMP/E deletes the BACKUP entry. When all BACKUP entries have been processed, SMP/E deletes the related SYSMOD entry. This processing is done before the target system libraries are updated.

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

Deleted elements

If a SYSMOD being restored had element modification control statements with the DELETE operand, SMP/E attempts to bring back the target zone element entries that were deleted when the SYSMOD was applied. This is done by using the BACKUP entries for the SYSMOD. For each BACKUP element entry, SMP/E checks whether there is a corresponding entry in the target zone.

- If there is no target zone entry for the element, SMP/E copies the BACKUP element entry into the target zone.
- If there is a target zone entry for the element, SMP/E issues a message to indicate that the entry has not been replaced with the BACKUP entry, and RESTORE processing continues.

  This can happen if you used UCLIN or JCLIN to re-create an entry after you applied the SYSMOD being restored, or if a subsequent SYSMOD was applied that re-created the entry but did not define a relationship with the SYSMOD being restored.

As SMP/E completes processing for each element, it deletes the corresponding BACKUP entry. When all BACKUP entries for the SYSMOD have been processed, SMP/E deletes the related SYSMOD entry. It then updates the target libraries using the procedure described in the following sections.

Compressing the target libraries

You can use the COMPRESS operand of the RESTORE command to have SMP/E compress the target libraries before restoring SYSMODs. There are two methods of specifying which libraries are to be compressed:

- You can specify selected libraries in the COMPRESS operand (including libraries that may not be affected by the RESTORE command).

- You can specify ALL in the COMPRESS operand; only libraries in which elements will be restored by this RESTORE command are then eligible for compression.

Note: Target libraries residing in a UNIX file system are not compressed.

Once the libraries have been determined, actual processing is dependent on the library type.

- For **source** libraries, any source to be replaced (not updated) by one of the SYSMODs being restored is deleted from the library.
- For **macro** libraries, no macros are deleted, because the SYSMOD replacing the macro might fail, and the failure could lead in turn to the failure of other SYSMODs causing assemblies that use the macro.
- For **data element** libraries, any data element that is to be replaced by one of the SYSMODs being restored is deleted from the library.
- For **load** libraries, SMP/E determines which load modules within the library are composed only of modules that were copied during system generation and are being replaced by the SYSMODs being restored. Such load modules are deleted from the library.

SMP/E then calls the copy utility to perform the actual compress operation.

Note: To reclaim as much space as possible before restoring the SYSMODs, members are deleted before the library is compressed. SMP/E processes one library at a time, first deleting members, then compressing the library.

Data elements

Note: For a list of data element types, refer to the "Data Element MCS" chapter in *z/OS SMP/E Reference*.

Data elements are copied from the distribution libraries to the target libraries by either SMP/E or the copy utility. SMP/E performs the copy in these cases:

- The target library or distribution library is a sequential data set.
- The data element must be reformatted to be compatible with the target library. For more information on reformatting data elements, see “Reformatting data elements” on page 98.

Hierarchical file system elements

Hierarchical file system elements are copied from the distribution libraries to the target libraries using the HFS copy utility.

Java archive file elements

Java Archive (JAR) file elements are copied from the distribution libraries to the target libraries using the HFS copy utility, with the BINARY option specified.

Program elements

Program elements are copied from the distribution libraries to the target libraries using the copy utility. A COPYMOD control statement is passed to the copy utility.

Macros

Macros existing in a target system library (that is, their SYSLIB subentry is nonblank) are simply copied from the distribution library into the appropriate target library. If no version of the macro is found in the distribution library, the macro is deleted from the target library.

For a macro that does not reside in any target library, SMP/E simply removes the current version of that macro from the SMPMTS.

Source

The processing of source code is exactly the same as the processing of macros, except that the SMPSTS is used rather than the SMPMTS.

Assemblies

RESTORE processing for assemblies is done in exactly the same way as during APPLY processing. For the detailed description, see “Assemblies” on page 93.

Modules

RESTORE processing for modules is essentially the same as APPLY processing, except that the source for the module replacement is the module distribution library rather than the selected version from a SYSMOD. For the detailed description, see “Module replacements” on page 95.

Note:

1. A superzap (that is, ++ZAP) is also considered a module, because the whole module is replaced from the distribution libraries.
2. If MODDEL subentries were added to LMOD entries to indicate that a module was deleted during APPLY processing, the MODDEL subentries are deleted from the LMOD entries during RESTORE processing.
3. SMP/E checks whether load modules to be updated have XZMOD subentries with the same name as a module selected to update the load module. If so, SYSMOD processing stops.
4. When multiple output libraries must be updated for link-edit processing, SMP/E may initiate a separate subtask for each such library, as described in “Multitasking of link-edit utility invocations” on page 97.

Load modules created by the SYSMOD being restored

Typically, if a load module was originally created by the SYSMOD being restored, SMP/E deletes the load module and the associated LMOD entry. If such a load module contains cross-zone modules, however, SMP/E does not delete the load module or the LMOD entry. Instead, it invokes the linkage-editor to remove the modules that are defined in the same zone as the load module (leaving a *stub* load module in the target library).

Load modules with a SYSLIB allocation

If RESTORE processing replaces a load module having a SYSLIB allocation with a version of the load module that does not have one, the base version of the load module (if it exists) is deleted from the SMPLTS data set. (In this case, the load module's LMOD entry in the target zone contains a CALLLIBS subentry list; that entry is replaced by an LMOD entry from the SMPSCDS that does not contain a CALLLIBS subentry list.) As a result, the executable version of the load module in its target libraries may contain modules that were included by the automatic library call mechanism when the load module was link-edited during APPLY processing of the SYSMOD now being restored. If there are still external references to these modules, they may continue to function in the load module; otherwise, they become inactive code in the load module.

Deleted load modules

SMP/E cannot restore a load module that was deleted by the ++DELETE statement.

Moved elements and load modules

If a macro, a module, a source, or a load module was moved by a ++MOVE statement, SMP/E returns it to its original library and deletes it from the one it was moved to. If a ++MOVE statement moved an element from one distribution library to another, the DISTLIB ddname in the target zone element entry is restored to its value before the move.

Renamed load modules

If a load module was renamed by the ++RENAME statement, SMP/E restores its original name.

Note: If a SYSMOD being restored contained a ++RENAME statement for a load module containing cross-zone modules, SMP/E checks whether those zones indicate that cross-zone updates should be done automatically. (This is done if you specify the AUTOMATIC option.) If so, the cross-zone MOD entries are updated during cross-zone processing. For more information, see *z/OS SMP/E Reference*.

Building load modules

After selecting the elements to be restored, SMP/E builds new load modules and rebuilds existing load modules, as required. See Appendix C, "Building load modules," on page 547 for a description of the process used by SMP/E to build load modules.

Recording after completion

Results of processing are recorded in the following entries.

Target zone element entries

The various function and service level fields (FMID, RMID, and UMID) in the target zone entries are modified to be the same as they are in the corresponding distribution zone entry.

SMPSCDS BACKUP entries

For each SYSMOD successfully restored that had inline JCLIN, the corresponding SMPSCDS BACKUP entry is deleted.

Target zone SYSMOD entries

Superseded SYSMODs

All SYSMOD entries that are superseded by SYSMODs being restored have the SUPBY subentries for those SYSMODs deleted. If all the SUPBY subentries for a superseded SYSMOD are deleted, the SYSMOD entry itself is deleted. As a result of restoring a SYSMOD that superseded a previously applied SYSMOD, target zone 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.

SYSMOD entry

When a SYSMOD is successfully restored, the SYSMOD entry is deleted from the target zone.

Cross-zone processing

If entries for modules or load modules that have been successfully restored contain cross-zone subentries, and if the associated cross-zones can be automatically updated, SMP/E does cross-zone processing.

First, SMP/E obtains access to the CSIs containing the cross-zones. It then checks each cross-zone to make sure it contains DDDEF entries for the target libraries needed for link-edit processing.

For each restored module that is part of a cross-zone load module, SMP/E checks the cross-zone LMOD entries to make sure the set-to zone originally supplied the modules to be processed. If so, SMP/E takes these actions:

- If the module was replaced by a distribution zone copy of the module, SMP/E schedules link-edit processing to include the replacement module.
- If the module was deleted, SMP/E schedules link-edit processing to delete that module.

Note: If a cross-zone LMOD entry to be processed consists only of cross-zone subentries, no processing is done for that load module. The load module no longer really exists.

For each cross-zone module contained in a renamed LMOD that was restored in the set-to zone, SMP/E changes the XZLMOD subentry to reflect the old LMOD name.

The Cross-Zone Summary report provides a summary of all the cross-zone work done, except for cross-zone work caused by renamed LMODs. This is summarized in the MOVE/RENAME/DELETE report.

Global zone SYSMOD entries

When a SYSMOD is successfully restored and the NOREJECT indicator in the OPTIONS entry in use is off, the global zone SYSMOD entry and the SMPPTS modification control statement entries are deleted along with any SMPTLIB data sets associated with that SYSMOD.

If the SYSMOD being restored has any exception SYSMOD data (that is, ++HOLD data) associated with it, that information is not deleted when the SYSMOD entry is deleted. You can restore a SYSMOD and then modify it and receive it again without having to rereceive the exception data associated with it.

When a SYSMOD is successfully restored and the NOREJECT indicator in the OPTIONS entry in use is on, the APPID subentry matching the target zone name is deleted, indicating that the SYSMOD is no longer applied to that zone.

Zone and data set sharing considerations

The following identifies the phases of RESTORE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

2. RESTORE processing

   Target zone
   Update with exclusive enqueue.

   DLIB zone
   Update with shared enqueue.

   Cross-zones
   Read with shared enqueue.

3. Global zone update

   Global zone
   Update with exclusive enqueue.

   SMPPTS
   Update with exclusive enqueue.

   Target zone
   Update with exclusive enqueue.

4. Cross-zone processing

   Distribution zone
   Read with shared enqueue.

   Cross-zones
   Read with shared enqueue.

Distribution zone
Update with exclusive enqueue.

Cross-zones
Update with exclusive enqueue.

Global zone
Read with no enqueue.

5. Termination

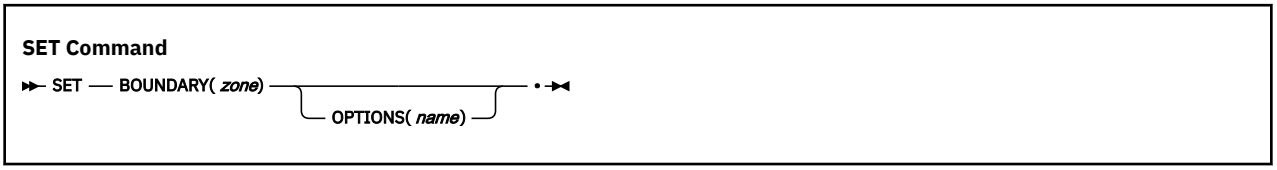
   All resources are freed.

Chapter 23. The SET command

Most SMP/E commands update certain zones. For example, the ACCEPT command updates a distribution zone, and the APPLY command updates a target zone. To specify which zone should be updated by a given command, you must use the SET command. The zone is identified on the BOUNDARY operand, which indicates that all subsequent commands, up to the next SET command, should be processed for the specified zone.

The SET command can also be used to request a particular set of predefined operating options. This is done with the OPTIONS operand. The OPTIONS operand specifies the global zone OPTIONS entry containing the processing options to be used for all subsequent commands, until the next SET command.

Syntax



Operands

BOUNDARY
specifies which zone should be updated by the commands following the SET command.

Note:

1. BOUNDARY is a required operand.
2. BOUNDARY can also be specified as BDY.

OPTIONS
specifies an OPTIONS entry that should be used for the commands following the SET command. If an OPTIONS entry is specified, it overrides the one specified in the zone definition.

Note:

1. The specified OPTIONS is used only to process the zone specified on the BOUNDARY operand.
2. For cross-zone processing, SMP/E uses the OPTIONS entry specified in the TARGETZONE entry for the cross-zone. If no OPTIONS entry is defined for the cross-zone, SMP/E uses default values when doing work to the cross-zone.

Data sets used

The following data sets might be needed to run the SET command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA	SMPPARM	*zone*

Note:

1. If SMP/E does not find the SMPLOG DD statement when parsing the SET command, SMP/E buffers all messages until after the specified zone has been determined. At that time, SMP/E accesses that zone to try to dynamically allocate the SMPLOG DD statement.

2. If SMP/E does not find the SMPOUT DD statement when parsing the SET command, SMP/E buffers all messages until after the specified zone has been determined. At that time SMP/E accesses that zone to try to dynamically allocate the SMPOUT DD statement.
3. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

SMP/E uses the SET command to control dynamic allocation. When SMP/E needs to dynamically allocate a data set, it allocates that data set once per zone. That data set remains allocated until the next SET command is processed. If SMP/E fails to dynamically allocate a data set, it keeps a record of that unsuccessful attempt and does not try to reallocate the data set. When SMP/E processes the next SET command, it frees all dynamically allocated data sets and erases the records of allocation attempts that failed. This has certain benefits:

- Each zone can use different definitions for the same data set.
- Performance is improved, because SMP/E does not need to dynamically allocate and free each data set every time it is needed.

For more information about dynamic allocation, see the "How to dynamically allocate data sets to be used during SMP/E processing" section in *z/OS SMP/E User's Guide*.

Examples

The following examples are provided to help you use the SET command.

Example 1: Receiving SYSMODs into the SMPPTS data set

To receive SYSMODs into the SMPPTS data set, SMP/E must be directed to process the global zone. Suppose you want to receive PTFs from an ESO tape containing service level 0701 into the SMPPTS. To do this, specify the following set of commands:

```
SET      BDY(GLOBAL)         /* Process global zone.     */.
RECEIVE  SYSMODS             /* Receive SYSMODs.         */.
LIST     SYSMODS             /* List SYSMOD entry        */
         MCS                 /* and MCS                  */
         SOURCEID(PUT0701)   /* for SYSMODS received.    */.
```

This causes all applicable SYSMODs to be received and to be assigned the source ID, specified in the ESO (in this case, 0701). The LIST command causes SMP/E to list the for all the SYSMODs just received.

Example 2: Applying SYSMODs to the target libraries

After receiving a set of SYSMODs, the next step is to apply them to the target libraries. To do this, the SET command must specify the target zone associated with those libraries. In this example, the SYSMODs are being installed into a target zone named MVSTST1 that represents a set of test libraries. The following commands are required to apply a SYSMOD:

```
SET      BDY(MVSTST1)        /* Set to process MVSTST1.  */.
APPLY    SOURCEID(PUT0701)   /* Apply service level      */
         GROUP               /* with group to pick up
                                previous resolved
                                exception SYSMODs.       */.
```

The result is that all PTFs that were received and assigned a source ID of PUT0701, and that are applicable to the functions in the MVSTST1 target system, are applied.

Note: The GROUP operand automatically includes requisites for the PTFs with the indicated source ID.

Example 3: Accepting SYSMODs to the distribution libraries

After applying a set of SYSMODs, the final step is to accept them into the distribution libraries. To do this, the SET command must specify the distribution zone associated with those libraries. In this example, the SYSMODs will be installed into a distribution zone named MVSDLB1. The following commands are required to accept a SYSMOD:

```
SET      BDY(MVSDLB1)       /* Set to process MVSDLB1.  */.
ACCEPT   SOURCEID(PUT0701)  /* Accept service level.    */.
```

The result is that SMP/E accepts all PTFs that were received and assigned a source ID of PUT0701, that have been applied, and that are applicable to the functions in the MVSDLB1 distribution zone.

Example 4: Processing multiple commands in one invocation of SMP/E

The preceding set of examples showed how the SET command is used to define the scope of processing to SMP/E when only one operation is to be performed at a time. SMP/E makes it possible to perform all these operations during one invocation, if desired. The commands are as follows:

```
SET      BDY(GLOBAL)        /* Process global zone.     */.
RECEIVE  SYSMODS            /* Receive SYSMODs.         */
         SOURCEID(PUT0701)  /* Assign SOURCEID.         */.
LIST     SYSMODS            /* List SYSMOD entry        */
         MCS(PUT0701)       /* and MCS                  */
         SOURCEID(PUT0701)  /* for SYSMODs received.    */.
SET      BDY(MVSTST1)       /* Set to process MVSTST1.  */.
APPLY    SOURCEID(PUT0701)  /* Apply service level.     */.
SET      BDY(MVSDLB1)       /* Set to process MVSDLB1.  */.
ACCEPT   SOURCEID(PUT0701)  /* Accept service level.    */.
```

Note: In a job with multiple SET commands, if you use DDDEF entries that specify SYSOUT for SMP/E output (such as SMPOUT or SMPRPT), SMP/E produces multiple SYSOUT data sets. This can cause undesirable results; for example, the output may appear to be out of sequence from one SET command to the next. Therefore, when you run such a job, you may prefer to use DD statements, rather than DDDEF entries, for SMP/E output data sets.

Example 5: Changing which OPTIONS entry is used

You can define multiple OPTIONS entries in the global zone so that various processing options can be used as required. The global zone and each target zone and distribution zone identify the default OPTIONS entry to be used in processing that zone. At times, you may require that a different OPTIONS entry be used for the installation of a given product or PTF. Rather than change the name of the default OPTIONS entry in the zone definition, SMP/E allows you to override the default OPTIONS name on the SET command.

For example, suppose you want to use the default OPTIONS entry to install all the service PTFs in service level 0701, but PTF UR12345 must be installed using another OPTIONS entry (previously defined with the correct unique processing options for this PTF). You can use the following set of commands:

```
SET      BDY(MVSTST1)       /* Set to process MVSTST1.  */.
APPLY    SOURCEID(PUT0701)  /* Apply all PTFs           */
         EXCLUDE(UR12345)   /* except UR12345.          */.
SET      BDY(MVSTST1)       /* Reset to change          */
         OPTIONS(URPTFS)    /* OPTIONS entry used.      */.
APPLY    S(UR12345)         /* Now apply UR12345.       */.
```

Example 6: Resolving errors in dynamic allocation

During processing, SMP/E attempts to dynamically allocate a data set one time per zone. If the allocation fails, SMP/E remembers and uses the information if the data set is requested again. For this example, let us assume that you are calling SMP/E from a terminal and that you have entered the following commands:

```
SET      BDY(MVSDLB1)       /* Set to process MVSDLB1.  */.
ACCEPT   S(UR12345)         /* Accept UR12345.          */.
```

Also, assume PTF UZ12345 requires distribution library AMACLIB, but that no DD statement has been allocated and no DDDEF entry is present. SMP/E issues an error message indicating the AMACLIB could not be allocated because no DDDEF entry was found, and the accept of the PTF fails. You can correct the problem by entering the following commands:

```
RESETRC                     /* Allows UCLIN to run after
                               accept failed.           */.
UCLIN                       /* Add AMACLIB DDDEF.       */.
  ADD    DDDEF(AMACLIB)     /* Add DDDEF                */
         DA(SYS1.AMACLIB)   /* with data set            */
         OLD                /* and disposition.         */.
ENDUCL                      /* End UCL changes.         */.
SET      BDY(MVSDLB1)       /* Set to process MVSDLB1.
                               Will also cause
                               allocation history for
                               AMACLIB to be deleted.   */.
ACCEPT   S(UR12345)         /* Accept UR12345.          */.
```

If the SET command had not been specified after the UCLIN operation, SMP/E would have issued a message indicating that an earlier attempt to allocate AMACLIB had failed and that no allocation attempt was made. As a result, the ACCEPT would have failed again.

Processing

When a SET command is encountered, SMP/E attempts to open the data set containing that zone. The data set to be opened is identified by looking in the global zone ZONEINDEX list.

- If no ZONEINDEX subentry exists, SMP/E reports an error condition.
- If a ZONEINDEX subentry exists, SMP/E checks to see if the data set specified for that zone is already open; if so, it does no further processing.
- If the data set containing the zone is not already open, SMP/E checks to see whether a DD statement has been provided (the ddname is equal to the zone name).
  - If a DD statement has been provided, the data set pointed to by the DD statement is opened.
  - If no DD statement has been provided, SMP/E attempts to dynamically allocate a DD statement using the zone name as the ddname and the data set specified in the ZONEINDEX as the data set name.

The SET command also determines whether the zone requires a higher level of SMP/E than the level of SMP/E that is currently running. This can happen if the zone had previously been updated with incompatible changes by a higher level SMP/E. If it does, SMP/E issues a message with a severe return code and the SET command fails.

Processing then continues with the next command.

Some common errors that can occur during a SET command are:

- The specified zone cannot be found in the global zone ZONEINDEX. In this case, SMP/E checks the syntax of all subsequent commands, but does not process them because it cannot determine where to direct the processing. To fix this problem, define the zones and rerun the job.
- The specified zone cannot be found on the data set specified in the global zone ZONEINDEX or the data set pointed to by the DD statement for that zone. In this case, the only SMP/E command that can be executed is the UCLIN command to define the zone definition entry. Other SMP/E commands fails because of insufficient information to process them.

Zone and data set sharing considerations

The following identifies the phases of SET processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

Target zone
Read without enqueue.

DLIB zone
Read without enqueue.

Note: The type of zone that is accessed depends on the zone specified in the SET command.

2. Global zone update

Global zone
Update with exclusive enqueue.

SMPPTS
Update with exclusive enqueue.

Target zone
Update with exclusive enqueue.

DLIB zone
Update with exclusive enqueue.

Note:

a. This phase is executed only if the zone type in the SET command was either a target zone or a distribution zone, and only if that target zone or distribution zone contained pending global zone updates from a previous APPLY, ACCEPT, or RESTORE command.

b. Either the target zone or distribution zone is accessed, according to the zone type specified in the SET command.

3. Termination

All resources are freed.

Chapter 24. The UCLIN command

With the UCLIN command you can add, delete, or replace entries in the following SMP/E data sets:

- SMPCSI
- SMPMTS
- SMPSCDS
- SMPSTS

Note: With the UCLIN command, you can make changes similar to those that can be made to other data sets with the IMASPZAP utilities. However, you cannot use a utility or an editor to change the information in these data sets; you must use the UCLIN command.

UCLIN updates only entries in SMP/E data sets. It does nothing to any elements or load modules in any product libraries. You must ensure that the appropriate changes are made to the libraries.

Be sure you understand the relationships between the various entries before making any UCLIN changes. This helps ensure that any UCLIN changes you make are complete and consistent with one another. When SMP/E processes UCLIN, it checks only the specified entry. It does not check how the changes might affect other entries.

The following terms are used in this discussion of UCLIN processing:

subentry
A field within an entry. Each subentry has an associated type and value. An example of a single-value subentry is the PEMAX subentry in the OPTIONS entry.

subentry list
Multiple occurrences of the same subentry type in an entry, each with a different value. For example, the modules supplied by a PTF are saved as names in the MOD subentry list within the PTF's SYSMOD entry.

subentry indicator
A field in an entry that does not have a data value associated with it. An example of a subentry indicator is the APP indicator in a SYSMOD entry.

Zones for SET BOUNDARY

For the UCLIN command, the SET BOUNDARY command must specify either the zone whose entries are to be changed, or the zone containing the DDDEF entry for the data set that is to be changed.

UCLIN and ENDUCL syntax

Three types of statements are needed for UCLIN processing:

1. The `UCLIN` command indicates the start of UCL processing.
2. `UCL statements` follow the UCLIN command and describe the changes for a specific entry. There are three types of UCL statements: ADD, DEL, and REP. These statements can add, delete, or replace entries or subentries in the entries.

   `ADD` is used to add the following entries:

   - A new entry
   - A new subentry to an existing entry
   - A new subentry list to an existing entry
   - A new subentry list value to an existing subentry list in an existing entry
   - A new subentry indicator to an existing entry

DEL is used to delete the following entries:

- An entire entry
- A subentry
- A complete subentry list
- A value from a subentry list
- A subentry indicator

REP is used to replace the following entries:

- A subentry in an existing entry
- A subentry list in an existing entry
- A subentry indicator in an existing entry

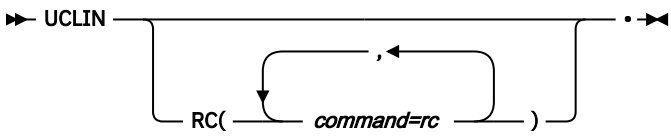
Note: Do **not** use the REP statement to replace an individual value in a subentry list. If the entry already contains the specified subentry list—for example, `FMID(ABC1234,DEF5678)`—SMP/E replaces **all** the current values with the new value specified on the REP statement.

Many UCL statements can follow a single UCLIN command. "UCL statement syntax" on page 363 describes the syntax of specific UCL statements for each entry type.

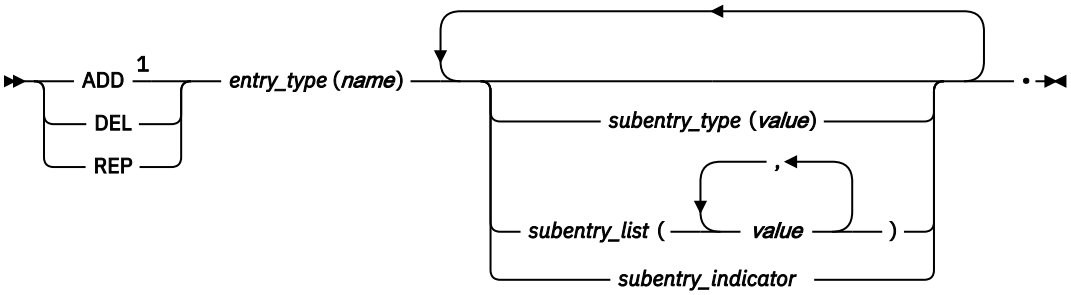
3. The ENDUCL command indicates the end of the UCL statements and the end of UCLIN processing.

This is the general syntax for these statements:

UCLIN Command



UCL Statements



Notes:

[1] You can specify more than one ADD, DEL, and REP statement.

ENDUCL Command

Operands

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the UCLIN command.

Before SMP/E processes the UCLIN command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the UCLIN command. Otherwise, the UCLIN command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the UCLIN command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

entry-type
specifies the entry type to be updated. "UCL statement syntax" on page 363 shows the entry types that can be specified. These entries are described in more detail in *z/OS SMP/E Reference*.

name
specifies the name of the entry to be updated.

subentry-type
specifies the subentry type to be updated. "UCL statement syntax" on page 363 shows the subentry types that can be specified. These subentries are described in more detail in *z/OS SMP/E Reference*.

subentry-list
specifies the type of subentry list to be updated. "UCL statement syntax" on page 363 shows the subentry types that can be specified. These subentries are described in more detail in *z/OS SMP/E Reference*.

subentry-indicator
specifies the subentry indicator to be updated. "UCL statement syntax" on page 363 shows the subentry types that can be specified. These subentries are described in more detail in *z/OS SMP/E Reference*.

UCL statement syntax

The UCL (Update Control Language) syntax descriptions in this chapter are arranged in alphabetical order. Table 23 on page 363 shows which entries can be processed in which zones and data sets.

Table 23. SMP/E entries that can be processed by UCLIN

Entry type	DLIB zone	Target zone	Global zone	Other data set
ASSEM	Yes	Yes		
BACKUP				SMPSCDS
Data element entries	Yes	Yes		
DDDEF	Yes	Yes	Yes	
DLIB	Yes	Yes		
DLIBZONE	Yes			
FEATURE			Yes	
FMIDSET			Yes	
GLOBALZONE			Yes	
Hierarchical file system element entry	Yes	Yes		
Java Archive file element entry	Yes	Yes		
LMOD	Yes	Yes		

Table 23. SMP/E entries that can be processed by UCLIN (continued)				
Entry type	**DLIB zone**	**Target zone**	**Global zone**	**Other data set**
MAC	Yes	Yes		
MOD	Yes	Yes		
MTSMAC				SMPMTS
OPTIONS			Yes	
ORDER			Yes	
PRODUCT			Yes	
PROGRAM	Yes	Yes		
SRC	Yes	Yes		
STSSRC				SMPSTS
SYSMOD	Yes	Yes	Yes	
TARGETZONE		Yes		
UTILITY			Yes	
ZONESET			Yes	

Not all the UCL statements can be used for each entry type. Table 24 on page 364 shows which UCL statements can be used for entries in which SMP/E data sets.

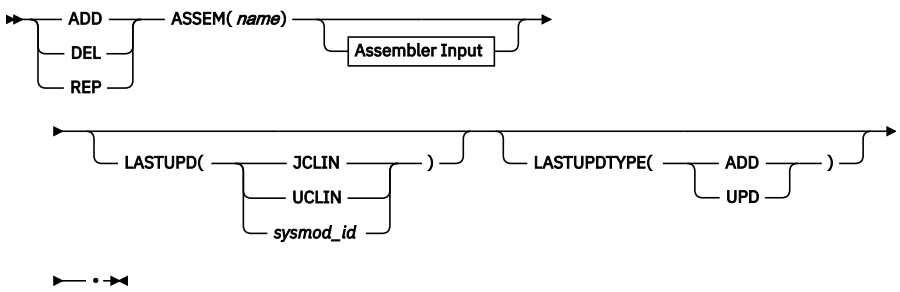
Table 24. UCL statements for SMP/E data sets			
Data set	**ADD**	**DEL**	**REP**
SMPCSI	Yes	Yes	Yes
SMPMTS		Yes	
SMPSCDS		Yes	
SMPSTS		Yes	

This chapter shows only the syntax of UCL statements used to process entries. See *z/OS SMP/E Reference* for additional information about each entry, such as:

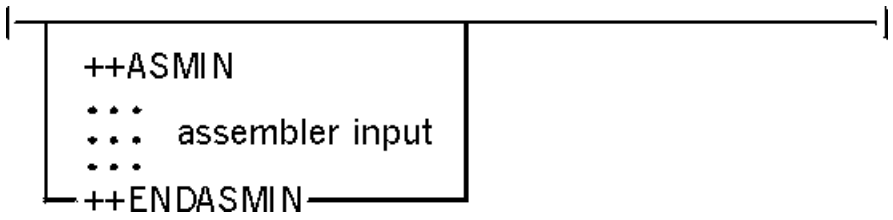
- A description of the entry and its subentries
- LIST examples
- UNLOAD examples
- UCLIN examples

ASSEM entry syntax (distribution and target zone)

ASSEM Entry



Assembler Input:



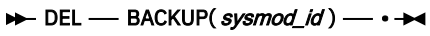
Note:

1. After UCLIN changes are done, the ASSEM entry must contain at least these subentries, unless the entire entry has been deleted:
   - ++ASMIN and ++ENDASMIN statements
   - The associated assembler input
2. The ++ASMIN and ++ENDASMIN statements must start in column 1.

For a description of the subentries in the distribution or target zone ASSEM entry, see *z/OS SMP/E Reference*.

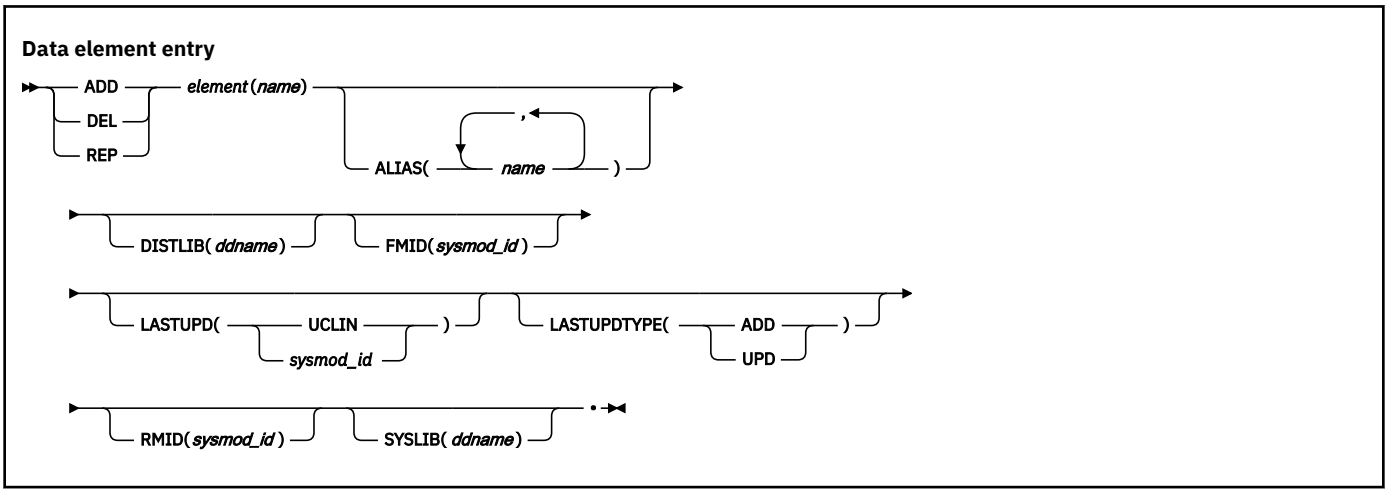
BACKUP entry syntax (SMPSCDS data set)

BACKUP Entry



For a description of the BACKUP entry, see *z/OS SMP/E Reference*.

Data element entry syntax (distribution and target zone)



Note:

1. After UCLIN changes are done, the data element entry must contain at least these subentries, unless the entire entry has been deleted:
   - DISTLIB
   - FMID
   - RMID
2. The "Data Element MCS" section in *z/OS SMP/E Reference* shows the types of data elements that can be specified for the *element* operand.
3. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand might not contain any *xxx* value.) *z/OS SMP/E Reference* contains a table, "National Language Identifiers Used for Language-Unique Elements", that shows the *xxx* values and the languages they represent.

For a description of the subentries in data element entries, see *z/OS SMP/E Reference*.

DDDEF entry syntax (distribution, target, and global zone)

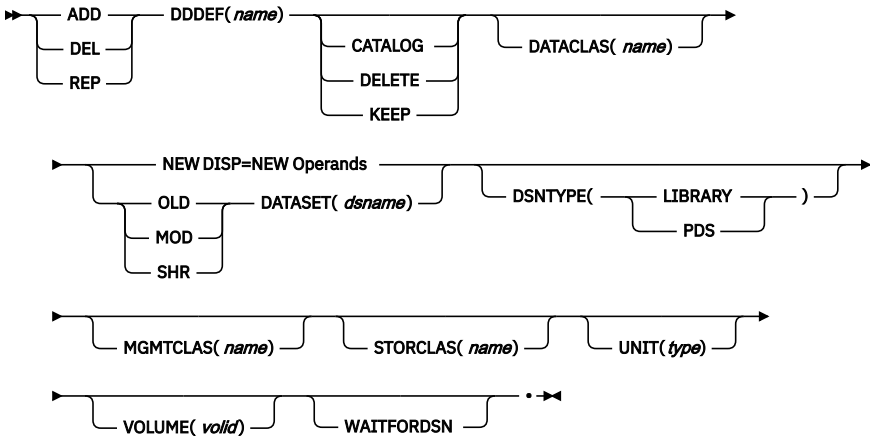
A separate syntax diagram is provided for each of the following types of data sets:

- Individual data set other than SMPTLIB or SYSOUT
- SMPTLIB data set in the global zone
- SYSOUT data set
- Concatenated data sets
- Path in a UNIX file system

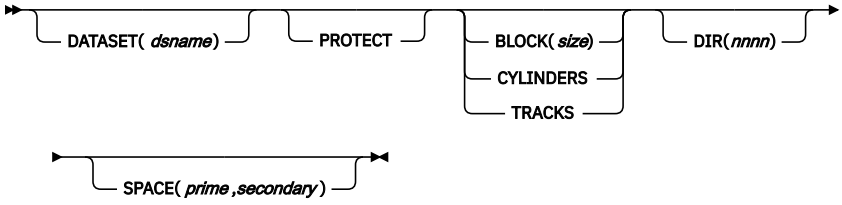
For a description of the subentries in DDDEF entries, see *z/OS SMP/E Reference*.

Individual data set other than SMPTLIB or SYSOUT

DDDEF entry



DISP=NEW Operands



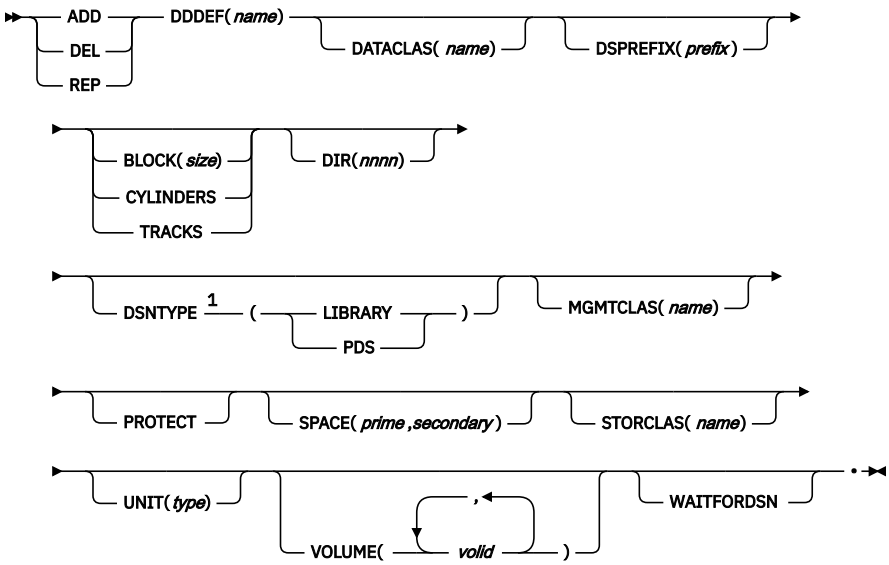
Notes except for concatenated data sets and paths:

1. BLOCK can also be specified as BLK.
2. CYLINDERS can also be specified as CYL.
3. DATASET can also be specified as DA.
4. When SMP/E RECEIVE processing allocates a new SMPTLIB data set, it uses the original DSNTYPE of the corresponding RELFILE data set. If SMP/E cannot determine the original DSNTYPE of the corresponding RELFILE data set, SMP/E uses the DSNTYPE value specified in the SMPTLIB DDDEF entry.
5. DSPREFIX may only be specified in a global zone DDDEF entry.
6. TRACKS can also be specified as TRK.
7. WAITFORDSN can also be specified as WAIT.

SMPTLIB data set (global zone)

For information on SMPTLIB DDDEF entries in the target or distribution zones, refer to the SMPTLIB section in *z/OS SMP/E Reference*.

DDDEF entry

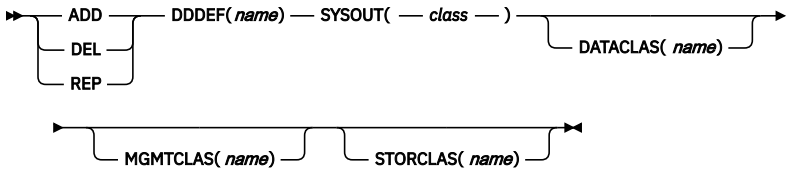


Notes:

[1] SMP/E ignores the DSNTYPE value in the SMPTLIB DDDEF entry, unless it cannot determine the DSNTYPE for the associated RELFILE data set.

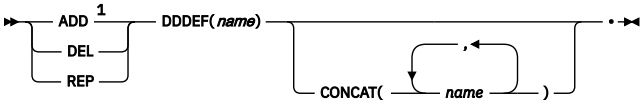
SYSOUT data set

DDDEF entry



Concatenated data sets

DDDEF entry



Notes:

[1] You cannot use ADD to add a subentry value to the CONCAT subentry. You must use REP to replace all the old values with the desired new values.

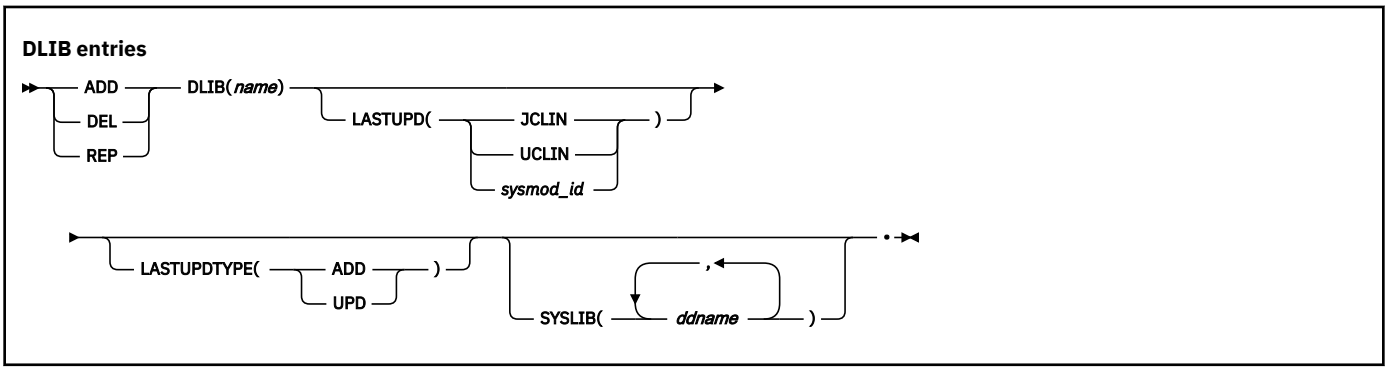
Path in a UNIX file system

Although a PATH subentry can be defined for a DDDEF entry in any type of zone, it is meaningful only in a target zone, because the information is used only when processing a target zone.

Notes except for concatenated data sets and paths:

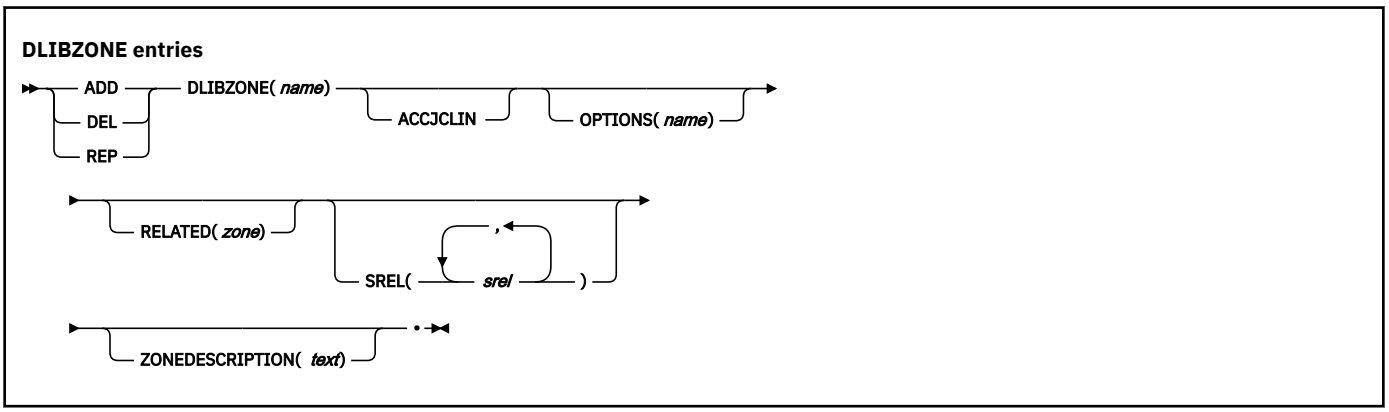
1. BLOCK can also be specified as BLK.
2. CYLINDERS can also be specified as CYL.
3. DATASET can also be specified as DA.
4. When SMP/E RECEIVE processing allocates a new SMPTLIB data set, it uses the original DSNTYPE of the corresponding RELFILE data set. If SMP/E cannot determine the original DSNTYPE of the corresponding RELFILE data set, SMP/E uses the DSNTYPE value specified in the SMPTLIB DDDEF entry.
5. DSPREFIX may only be specified in a global zone DDDEF entry.
6. TRACKS can also be specified as TRK.
7. WAITFORDSN can also be specified as WAIT.

DLIB entry syntax (distribution and target zone)



For a description of the subentries in the distribution or target zone DLIB entry, see *z/OS SMP/E Reference*.

DLIBZONE entry syntax (distribution zone)

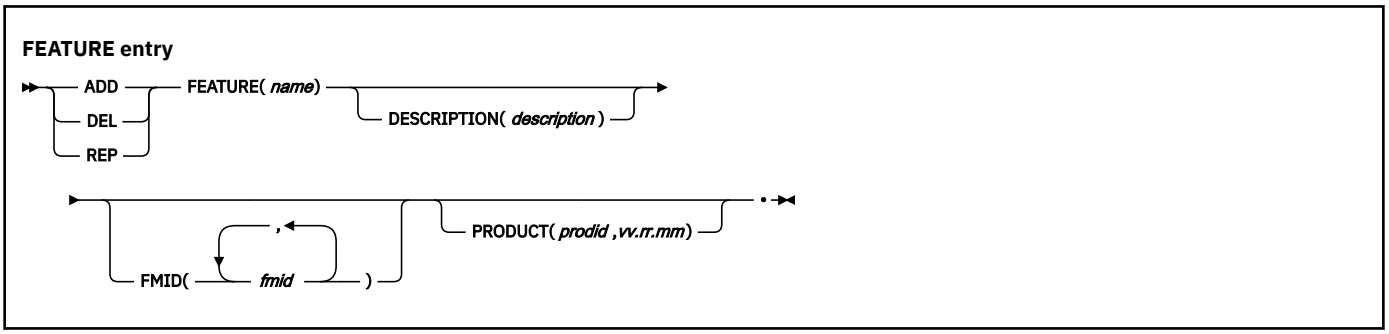


Note:

1. DLIBZONE can also be specified as DZONE.
2. ZONEDESCRIPTION can also be specified as ZDESC.

For a description of the subentries in the distribution zone DLIBZONE entry, see *z/OS SMP/E Reference*.

FEATURE entry syntax (global zone)

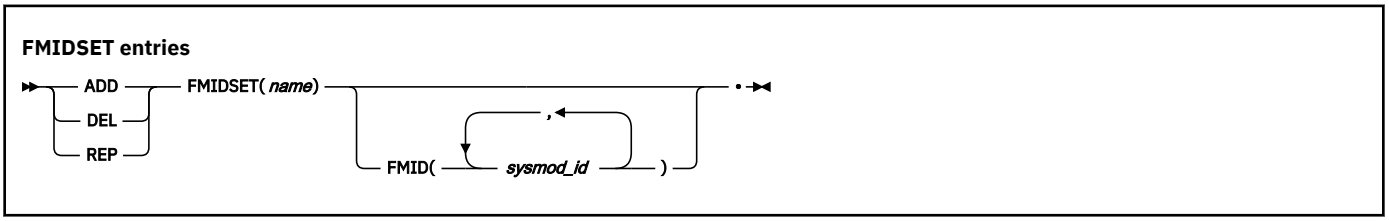


Note:

1. After UCLIN changes are made, the FEATURE entry must contain at least the DESCRIPTION and PRODUCT subentries, unless the entire entry has been deleted.
2. DESCRIPTION can also be specified as DESC.

For a description of the subentries in the FEATURE entry, see *z/OS SMP/E Reference*.

FMIDSET entry syntax (global zone)



Note:

1. After UCLIN changes are made, the FMIDSET entry must contain at least the FMID subentries, unless the entire entry has been deleted.
2. FMIDSET can also be specified as FMSET.

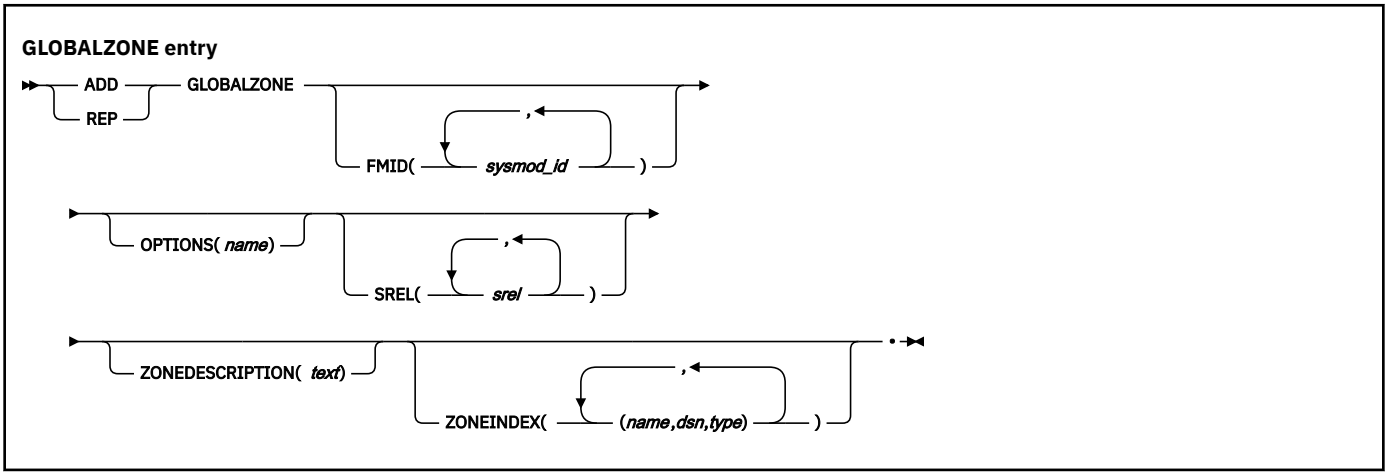
For a description of the subentries in the FMIDSET entry, see *z/OS SMP/E Reference*.

GLOBALZONE entry syntax (global zone)

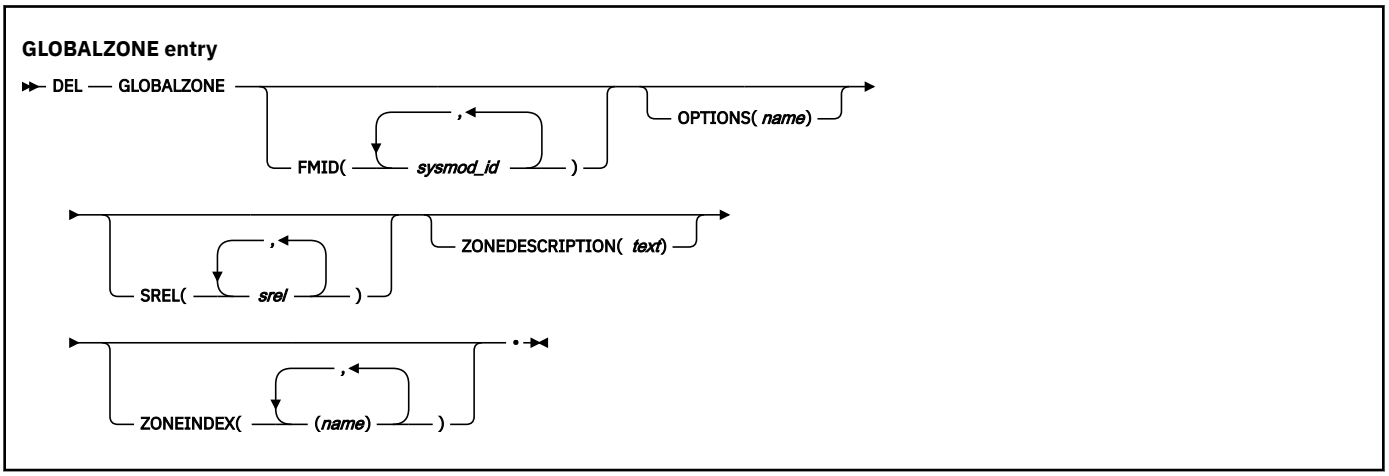
A separate syntax diagram is provided for:

- ADD and REP commands
- DEL commands

ADD and REP syntax



DEL syntax

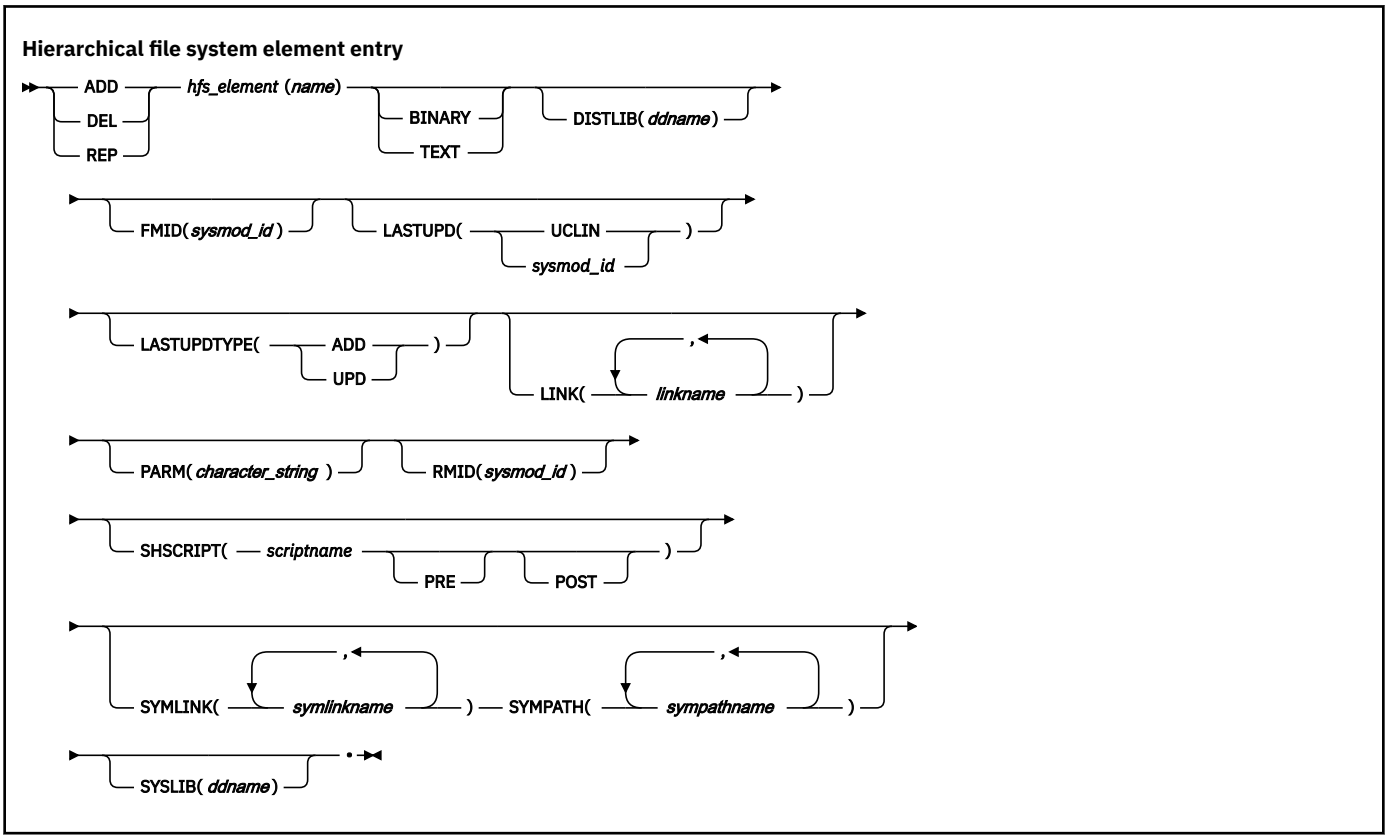


Notes for ADD, DEL, and REP syntax:

1. After UCLIN changes are made, the GLOBALZONE entry must contain at least one of these subentries, unless the entire entry has been deleted:
   - FMID
   - OPTIONS
   - SREL
   - ZONEINDEX
2. GLOBALZONE can also be specified as GZONE.
3. ZONEDESCRIPTION can also be specified as ZDESC.
4. ZONEINDEX can also be specified as ZINDEX.

For a description of the subentries in the GLOBALZONE entry, see *z/OS SMP/E Reference*.

Hierarchical file system element entry syntax (distribution and target zone)

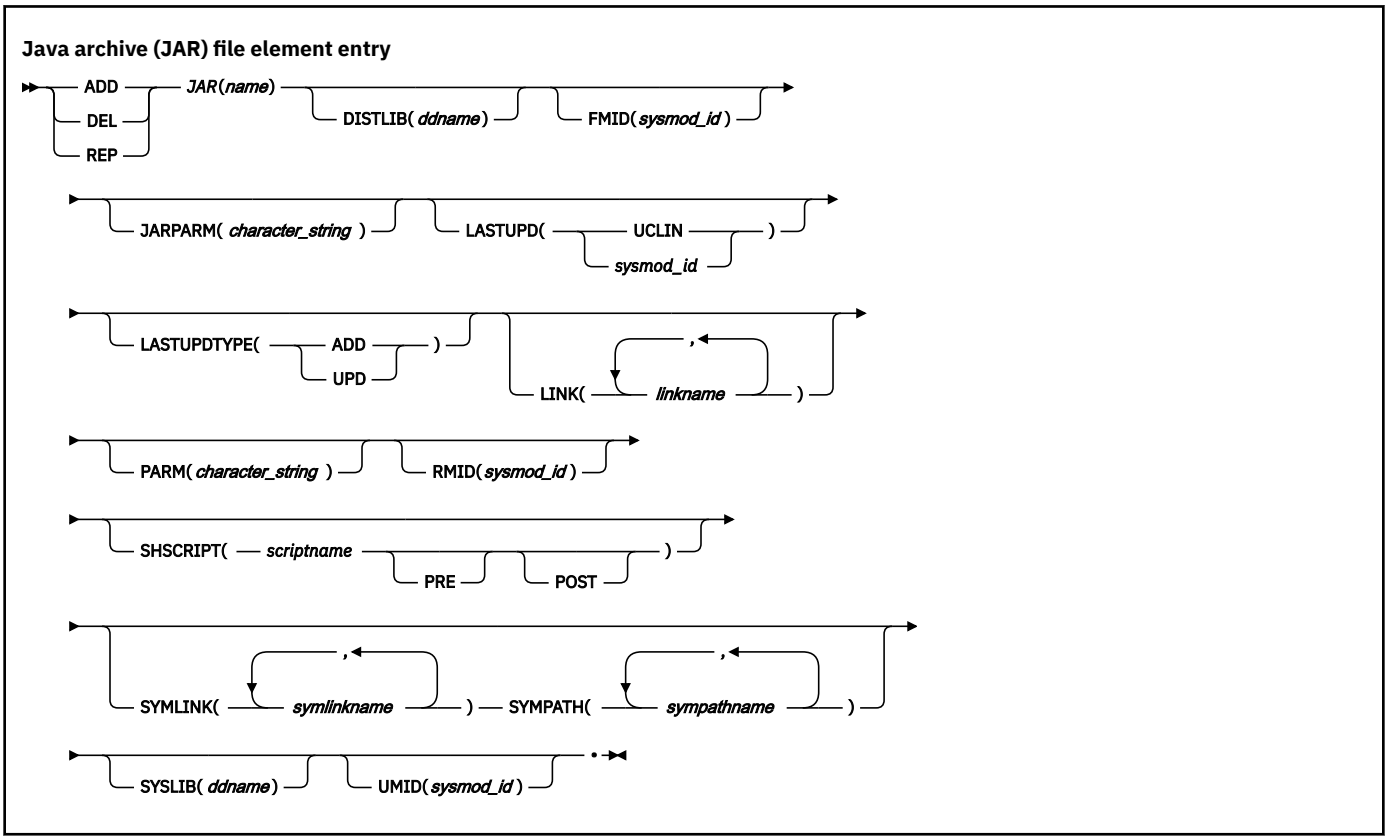


Note:

1. After UCLIN changes are made, the hierarchical file system element entry must contain at least these subentries, unless the entire entry has been deleted:
   - DISTLIB
   - FMID
   - RMID
   - SYSLIB
2. When the hierarchical file system element entry is SHELLSCR, observe the following restrictions:
   - *PRE* is not valid
   - *scriptname* must match the element's name.

For a description of the subentries in the hierarchical file system element entry, see *z/OS SMP/E Reference*.

Java archive (JAR) file element entry syntax (distribution and target zone)



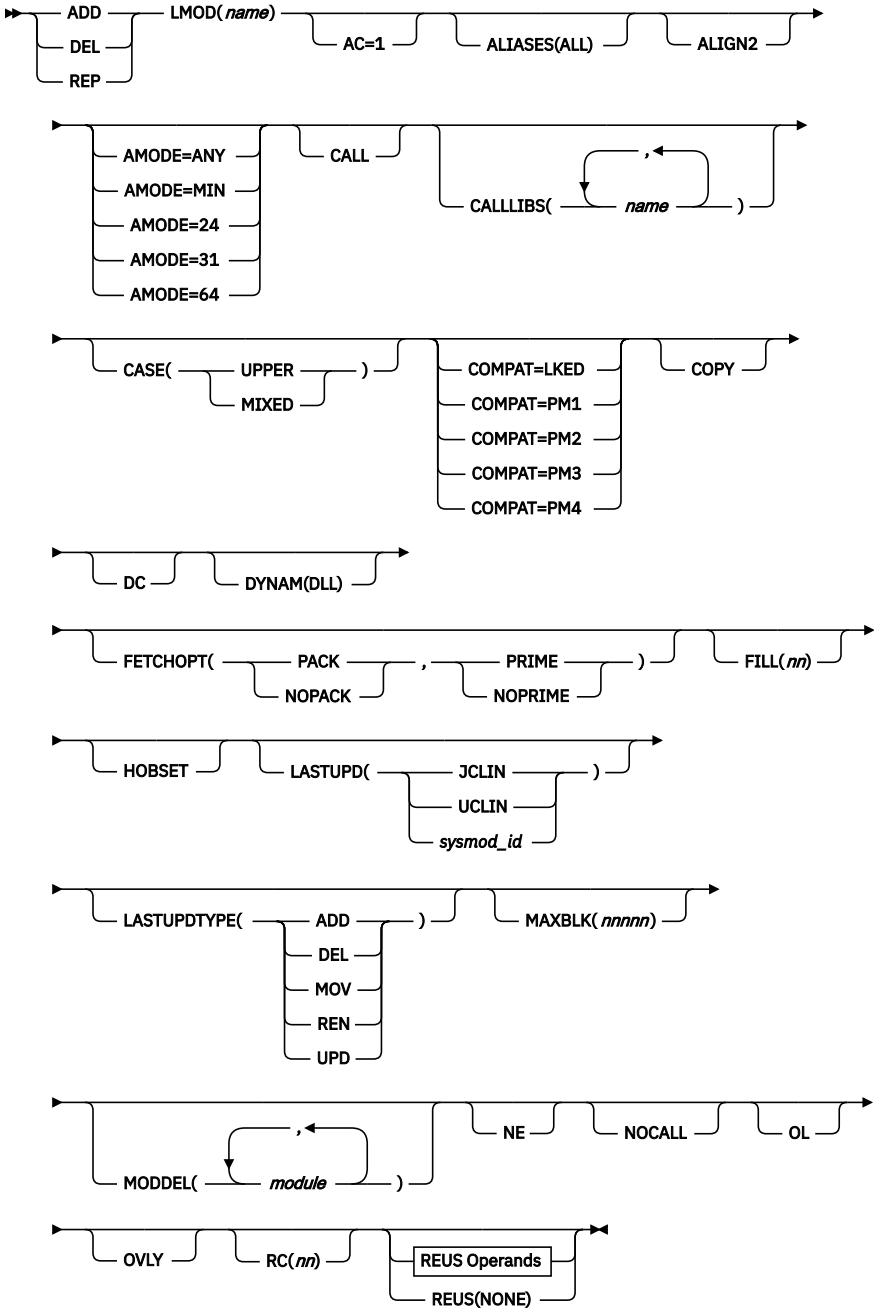
Note: After UCLIN changes are made, the JAR element entry must contain at least these subentries, unless the entire entry has been deleted:

- DISTLIB
- FMID
- RMID
- SYSLIB

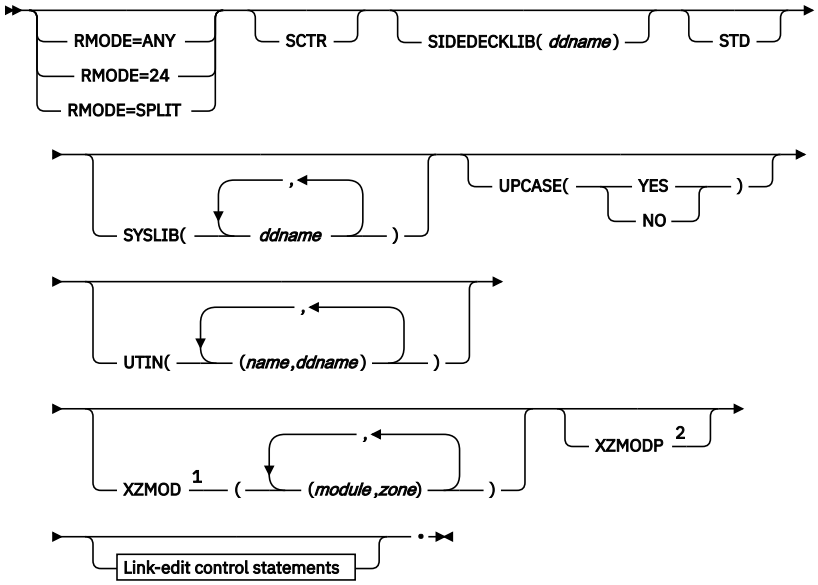
For a description of the subentries in the JAR file element entry, see *z/OS SMP/E Reference*.

LMOD entry syntax (distribution and target zone)

LMOD entry



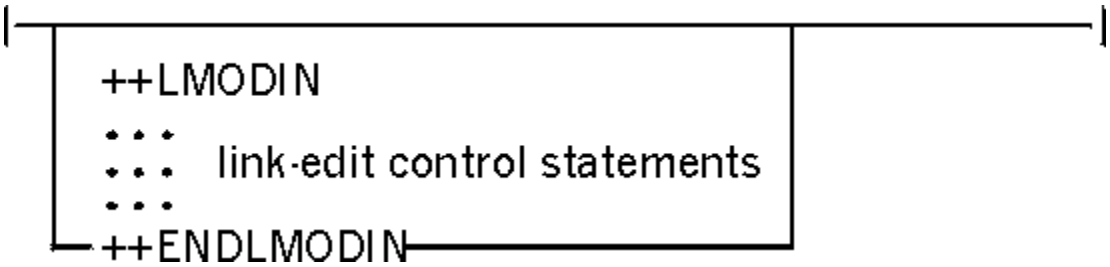
REUS Operands



Notes:

[1] XZMOD is valid only for a target zone.
[2] XZMODP is valid only for a target zone.

Link-edit control statements:

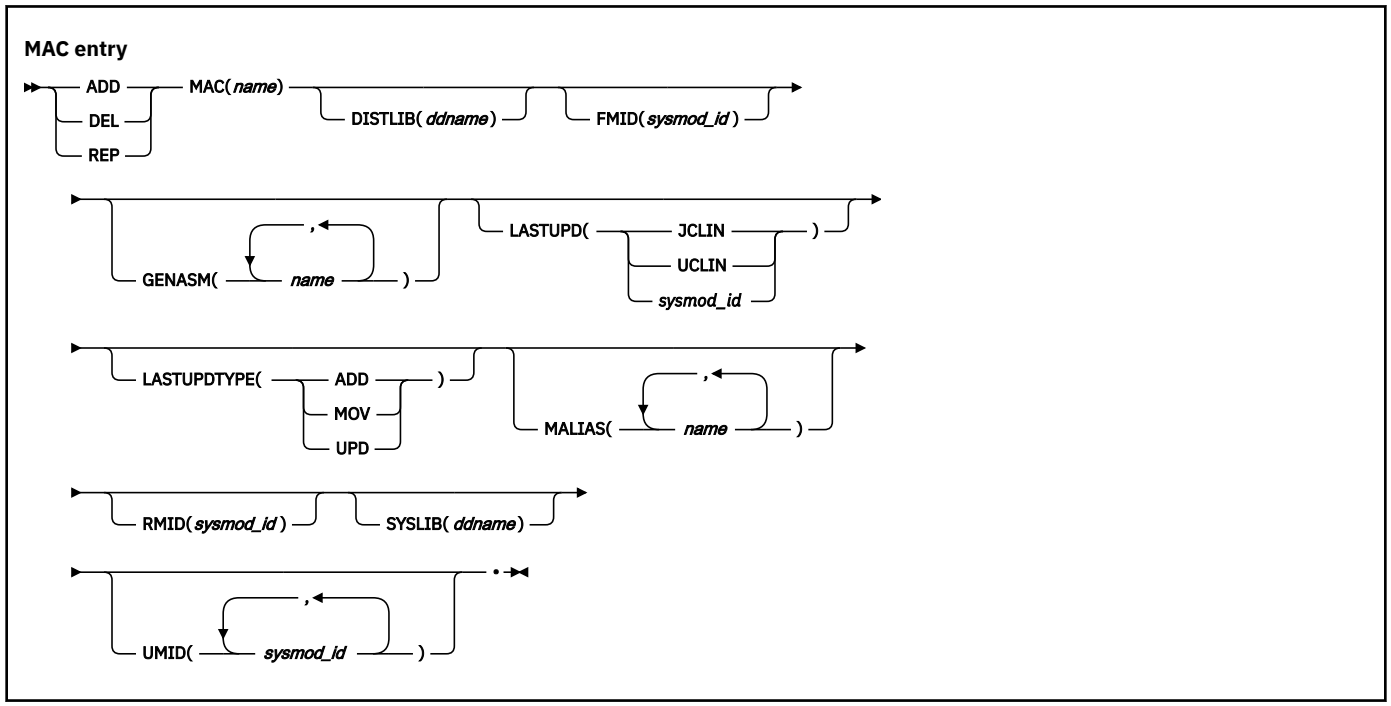


Note:

1. After UCLIN changes are made, the LMOD entry must contain at least the SYSLIB subentries, unless the entire entry has been deleted.
2. XZMOD and XZMODP subentries are valid only for target zone entries.
3. AMODE=*value* can also be specified as AMOD=*value*.
4. NOCALL can also be specified as NCAL.
5. REUSE(NONE) is mutually exclusive with REFR and RENT, as well as REUS.
6. RMODE=*value* can also be specified as RMOD=*value*.
7. The ++LMODIN and ++ENDLMODIN statements must start in column 1.
8. The link-edit control statements must start in or after column 2.

For a description of the subentries in the distribution or target zone LMOD entry, see *z/OS SMP/E Reference*.

MAC entry syntax (distribution and target zone)



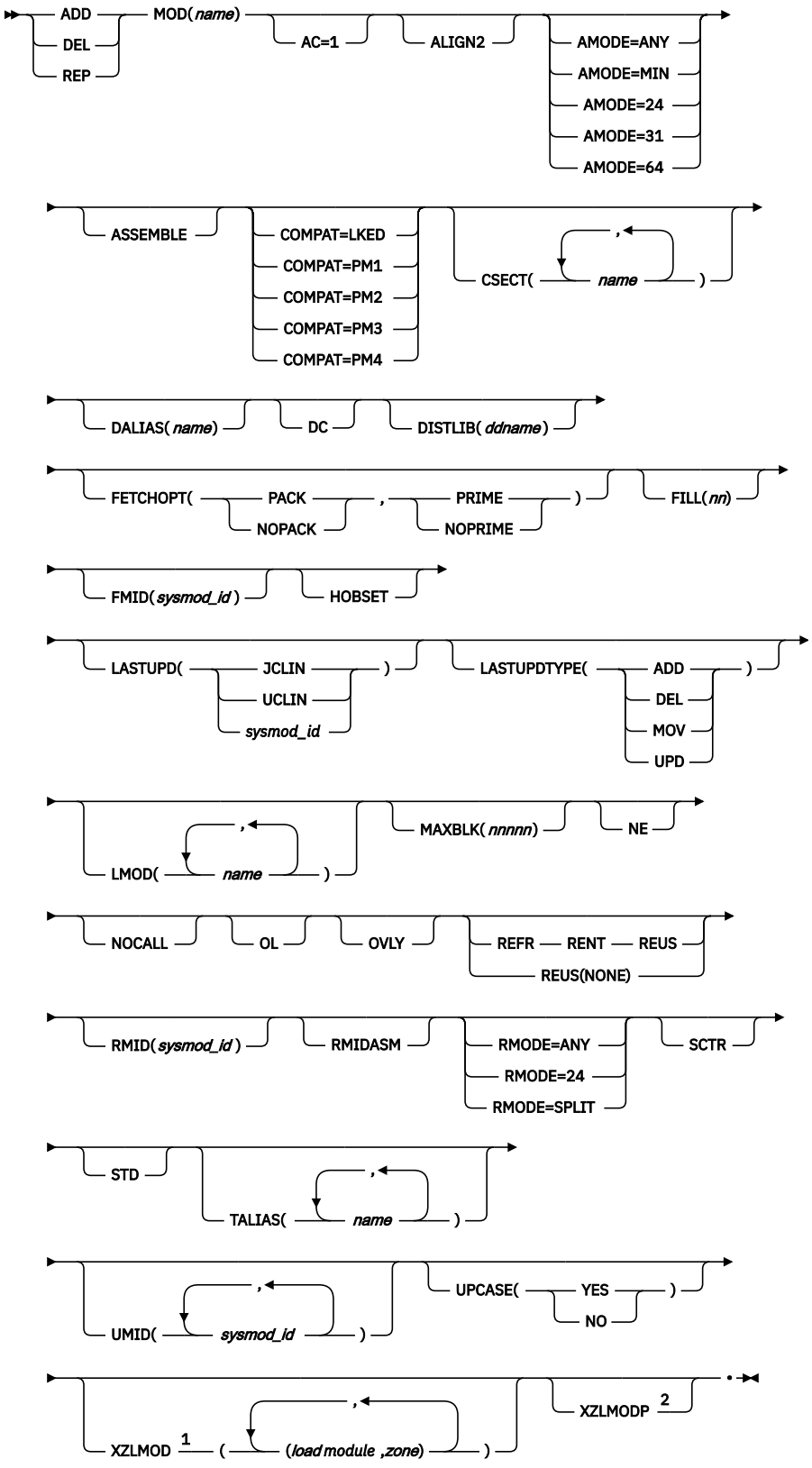
Note:

1. After UCLIN changes are made, the MAC entry must contain at least these subentries, unless the entire entry has been deleted:
   - FMID
   - RMID
2. GENASM can also be specified as ASSEM.

For a description of the subentries in the MAC entry, see *z/OS SMP/E Reference*.

MOD entry syntax (distribution and target zone)

MOD entry



Notes:

[1] XZLMOD is valid only for a target zone.
[2] XZLMODP is valid only for a target zone.

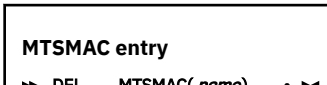
Note:

1. After UCLIN changes are made, the MOD entry must contain at least these subentries, unless the entire entry has been deleted:
   - DISTLIB
   - FMID
   - RMID
2. XZLMOD and XZLMODP subentries are valid only for target zone entries.
3. ALIGN2 can also be specified as ALN2.
4. AMODE=*value* can also be specified as AMOD=*value*.
5. NOCALL can also be specified as NCAL.
6. REUSE(NONE) is mutually exclusive with REFR and RENT, as well as REUS.
7. RMODE=*value* can also be specified as RMOD=*value*.

For a description of the subentries in the MOD entry, see *z/OS SMP/E Reference*.

MTSMAC entry syntax (SMPMTS data set)

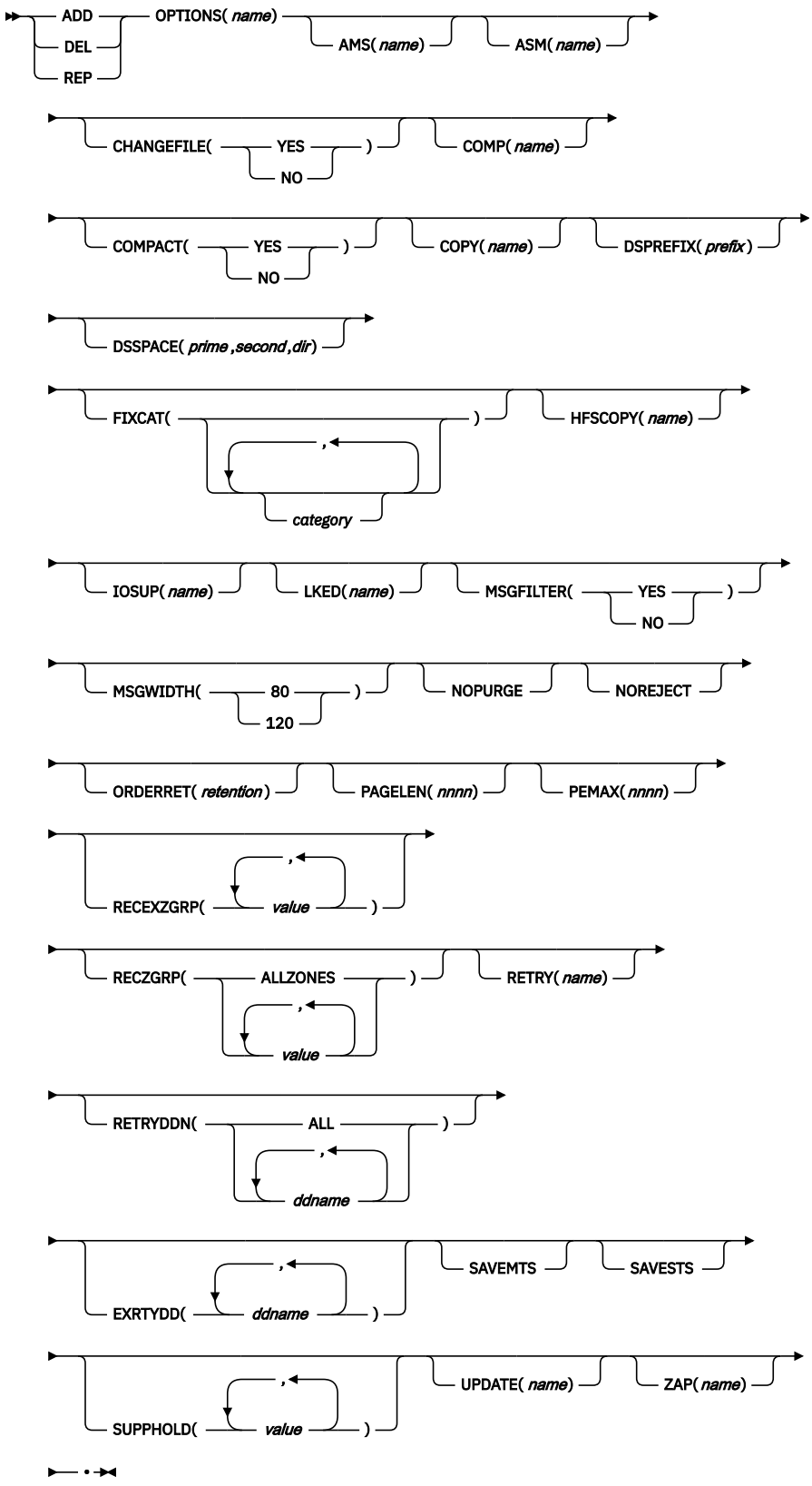
MTSMAC entry



For a description of the MTSMAC entry, see *z/OS SMP/E Reference*.

OPTIONS entry syntax (global zone)

OPTIONS entry



For a description of the subentries in the OPTIONS entry, see *z/OS SMP/E Reference*.

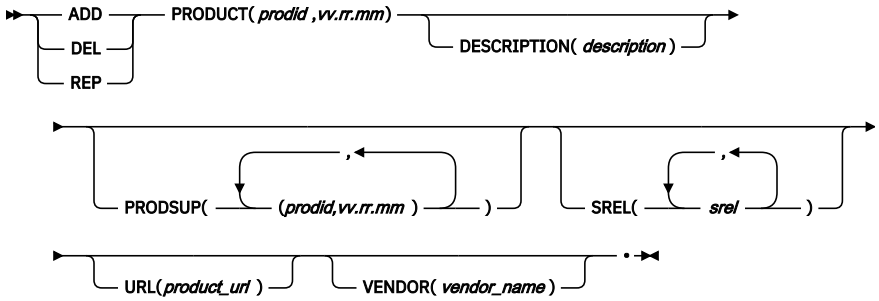
ORDER entry syntax (global zone)

ORDER entry

For a description of the ORDER entry, see *z/OS SMP/E Reference*.

PRODUCT entry syntax (global zone)

PRODUCT entry



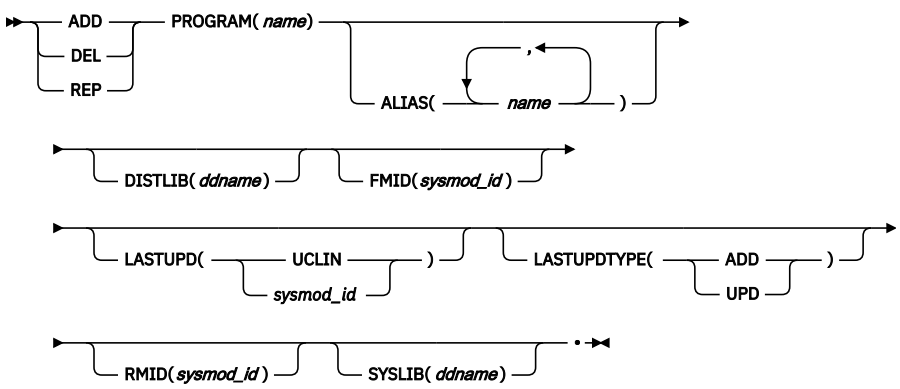
Note:

1. After UCLIN changes are made, the PRODUCT entry must contain at least the DESCRIPTION and SREL subentries, unless the entire entry has been deleted.
2. DESCRIPTION can also be specified as DESC.

For a description of the subentries in the PRODUCT entry, see *z/OS SMP/E Reference*.

Program element entry syntax (distribution and target zone)

Program element entry

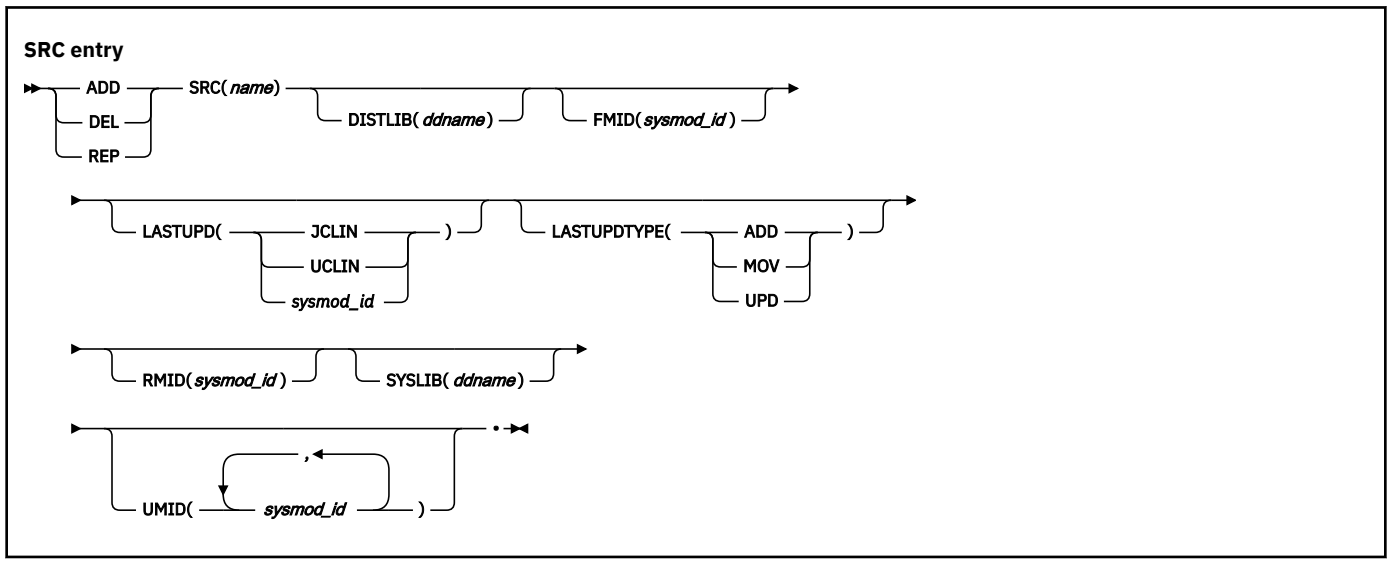


Note:

1. After UCLIN changes are done, the program element entry must contain at least these subentries, unless the entire entry has been deleted:
   - DISTLIB
   - FMID
   - RMID

For a description of the subentries in the program element entry, see *z/OS SMP/E Reference*.

SRC entry syntax (distribution and target zone)



For a description of the subentries in the SRC entry, see *z/OS SMP/E Reference*.

STSSRC entry syntax (SMPSTS data set)

For a description of the STSSRC entry, see *z/OS SMP/E Reference*.

SYSMOD entry syntax (distribution and target zone)

A separate syntax diagram is provided for each of the following types of SYSMOD entries:

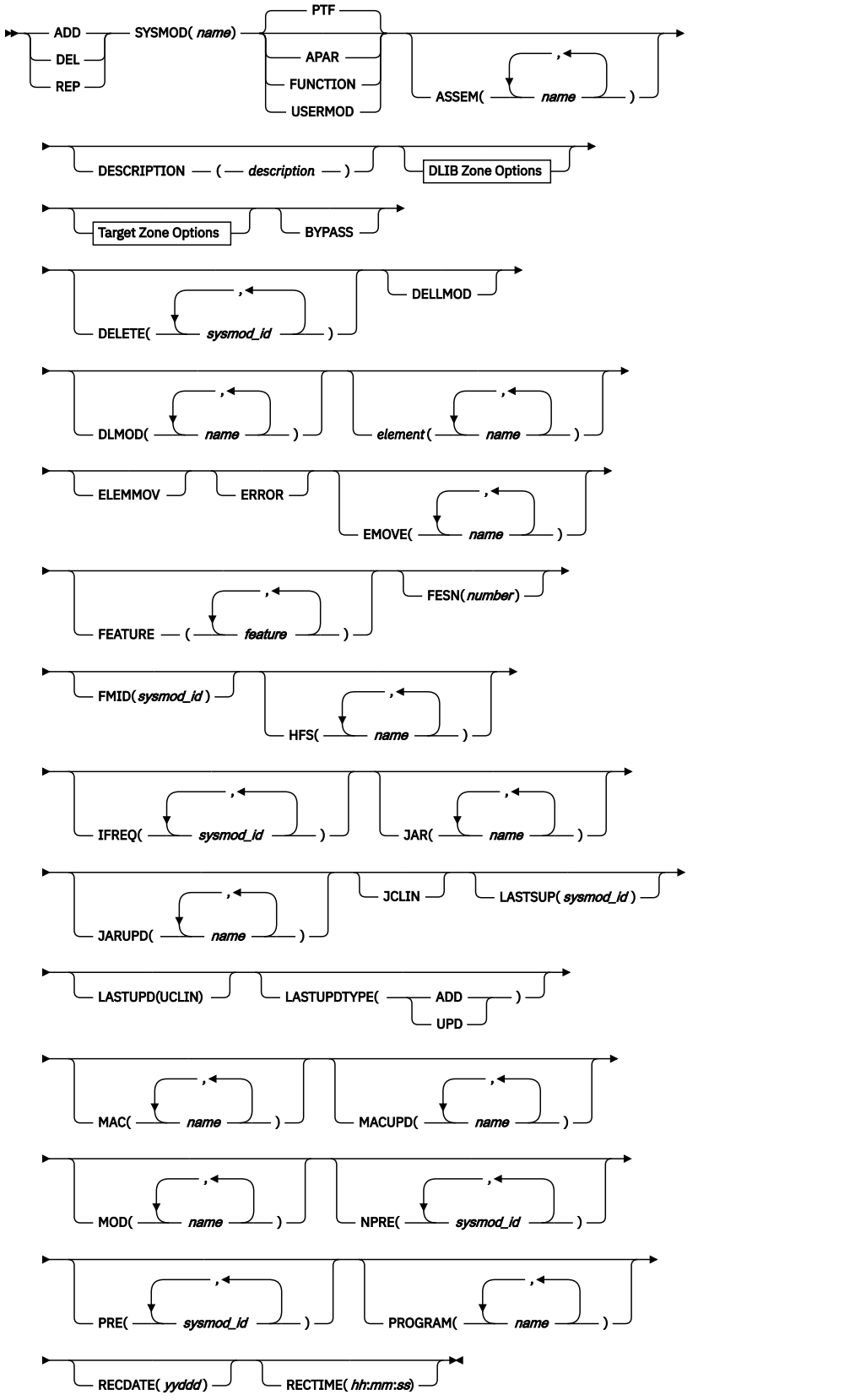
- SYSMOD entry other than for a deleted SYSMOD
- SYSMOD entry for deleted SYSMOD
- SYSMOD entry containing only a CIFREQ subentry

Note:

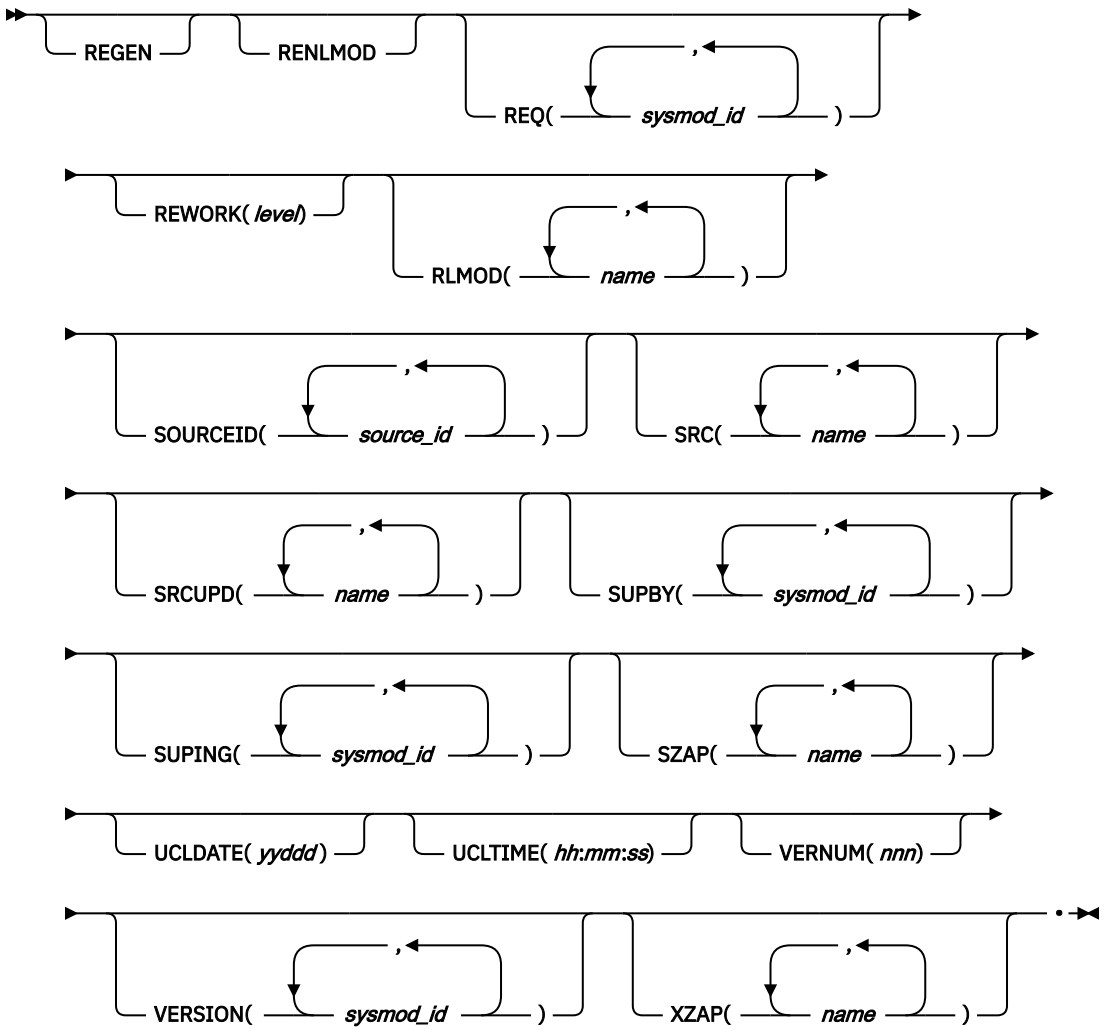
1. A source ID value cannot span lines in the UCLIN command.
2. You must explicitly specify a source ID in the UCLIN command. You cannot use asterisks and percent signs as placeholders to implicitly identify a set of source IDs in the UCLIN command.

SYSMOD entry other than deleted SYSMOD

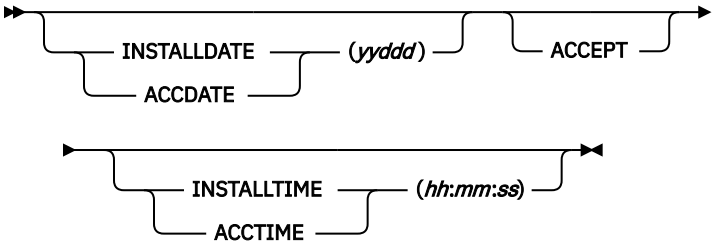
SYSMOD entry (Part 1 of 2)



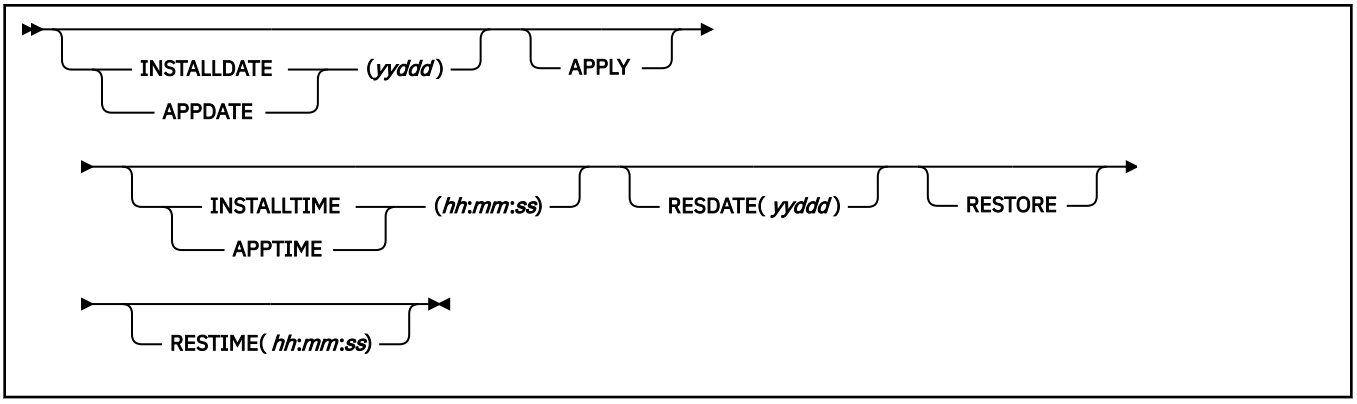
SYSMOD entry (Part 2 of 2)



DLIB zone options



Target zone options



Note:

1. Generally, after UCLIN changes are made, the SYSMOD entry must contain at least these subentries, unless the entire entry has been deleted:
   - ACCEPT or APPLY
   - APAR, FUNCTION, PTF, or USERMOD
   - INSTALLDATE
   - RECDATE
   - FMID

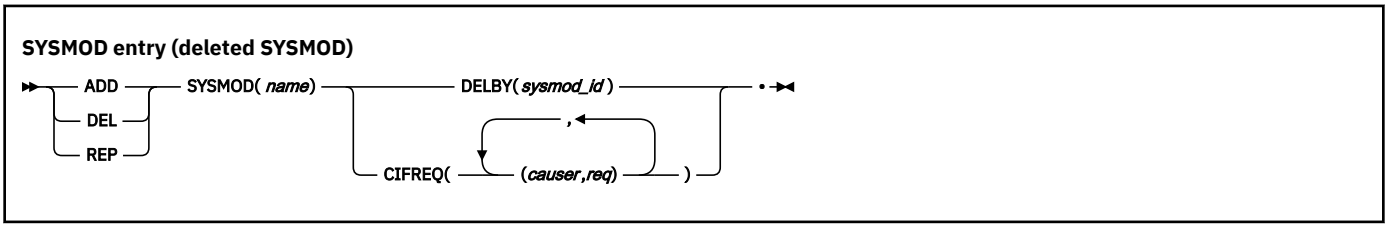
   However, the entry for a deleted (but not superseded) SYSMOD can contain only the DELBY and CIFREQ subentries. The entry for a SYSMOD that is superseded, or both deleted and superseded, must contain the SUPBY subentry, instead of the DELBY subentry. Also, a SYSMOD entry can be created with only a CIFREQ subentry to identify requisites for a SYSMOD to be installed later.
2. ACCDATE, ACCEPT, and ACCTIME can be used only in a distribution zone SYSMOD entry.
   - ACCEPT can also be specified as ACPT or ACC.
   - INSTALLDATE can be specified instead of ACCDATE. INSTALLDATE can also be specified as INSDATE.
   - INSTALLTIME can be specified instead of ACCTIME. INSTALLTIME can also be specified as INSTIME.
3. APPDATE, APPLY, and APPTIME can be used only in a target zone SYSMOD entry.
   - APPLY can also be specified as APPL or APP.
   - INSTALLDATE can be specified instead of APPDATE. INSTALLDATE can also be specified as INSDATE.
   - INSTALLTIME can be specified instead of APPTIME. INSTALLTIME can also be specified as INSTIME.
4. The *element* operand represents data elements. The "Data Element MCS" section *z/OS SMP/E Reference* shows the types of data elements that can be specified for the *element* operand.

   Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx* value.) *z/OS SMP/E Reference* contains a table, "National Language Identifiers Used for Language-Unique Elements", that shows the *xxx* values and the languages they represent.
5. ERROR can also be specified as ERR.
6. REGEN can also be specified as RGN.
7. RESDATE, RESTORE, and RESTIME can be used only in a target zone SYSMOD entry.
8. RESTORE can also be specified as REST or RES.

9. SUPBY can also be specified as SUP.
10. The CIFREQ operand is mutually exclusive with all other UCL operands.

For a description of the subentries in the distribution or target zone SYSMOD entry, see *z/OS SMP/E Reference*.

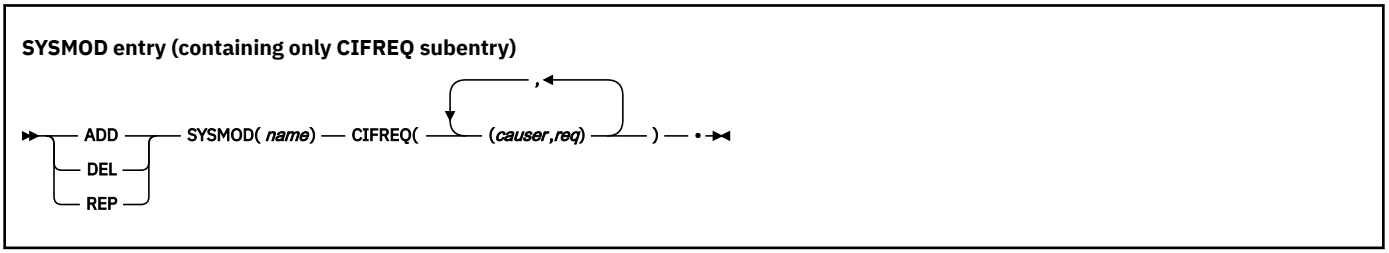
SYSMOD entry for deleted SYSMOD



Note: The DELBY and CIFREQ operands are mutually exclusive. Two separate UCLIN commands must be used to change both CIFREQ and DELBY.

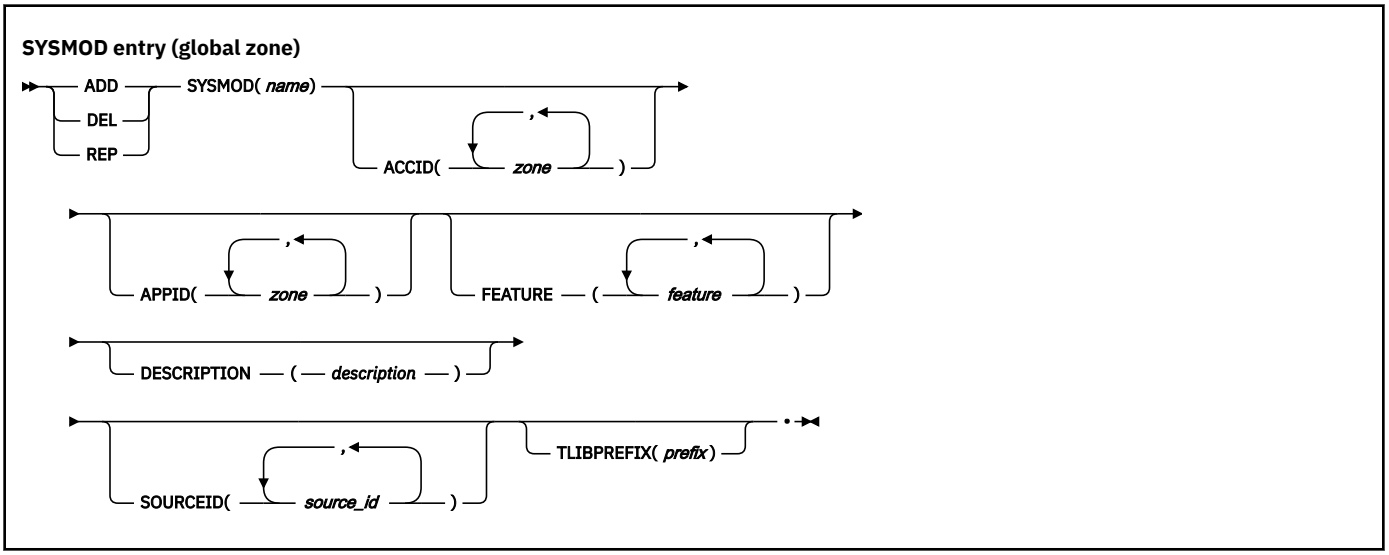
For a description of the subentries in the distribution or target zone SYSMOD entry, see *z/OS SMP/E Reference*.

SYSMOD entry containing only a CIFREQ subentry



For a description of the subentries in the distribution or target zone SYSMOD entry, see *z/OS SMP/E Reference*.

SYSMOD entry syntax (global zone)

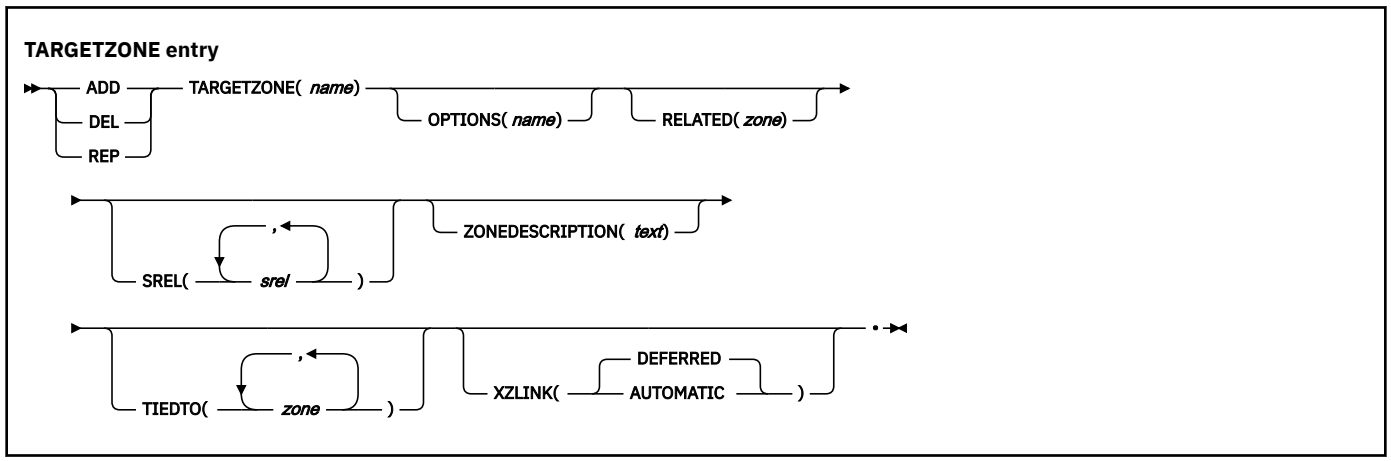


Note:

1. A source ID value cannot span lines in the UCLIN command.
2. You must explicitly specify a source ID in the UCLIN command. You cannot use asterisks and percent signs as placeholders to implicitly identify a set of source IDs in the UCLIN command.

For a description of the subentries in the global zone SYSMOD entry, see *z/OS SMP/E Reference*. Note that only a limited subset of these subentries can be modified with UCLIN.

TARGETZONE entry syntax (target zone)

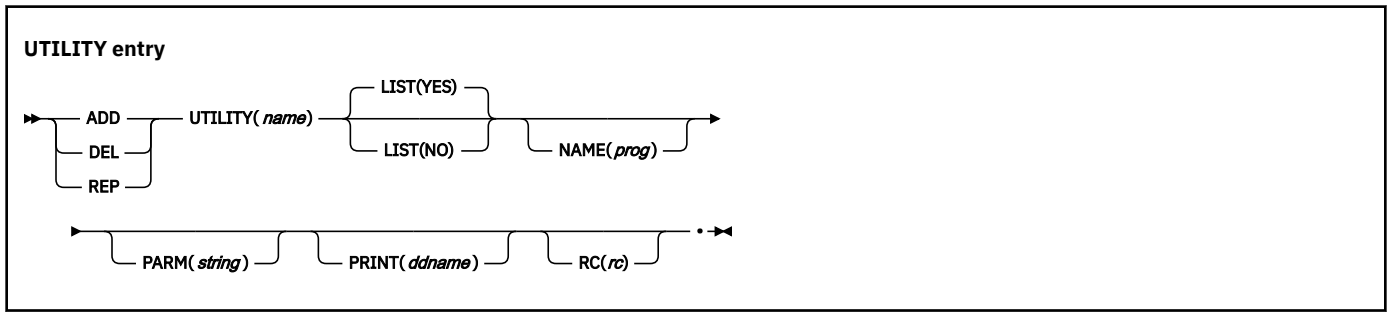


Note:

1. TARGETZONE can also be specified as TZONE.
2. ZONEDESCRIPTION can also be specified as ZDESC.

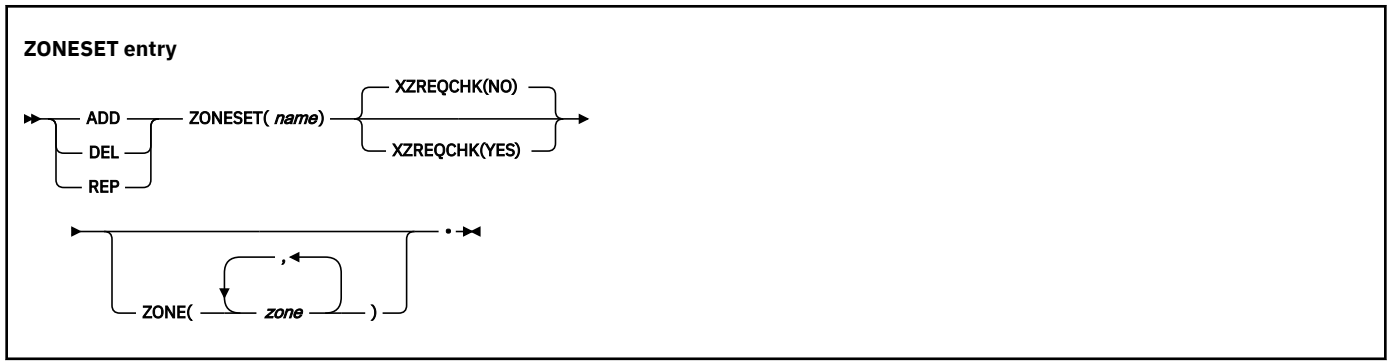
For a description of the subentries in the TARGETZONE entry, see *z/OS SMP/E Reference*.

UTILITY entry syntax (global zone)



For a description of the subentries in the UTILITY entry, see *z/OS SMP/E Reference*.

ZONESET entry syntax (global zone)



For a description of the subentries in the ZONESET entry, see *z/OS SMP/E Reference*.

Data sets used

The following data sets might be needed to run the UCLIN command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPTS	SMPSTS
SMPCSI	SMPMTS	SMPSCDS	*zone*
SMPLOG	SMPOUT	SMPSNAP	

Note:

1. SMPMTS is required if the MTSMAC entry is specified on a UCL statement.
2. SMPSTS is required if the STSSRC entry is specified on a UCL statement.
3. SMPSCDS is required if BACKUP entries are specified on a UCL statement.
4. *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated through the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

The File Allocation report is produced during UCLIN processing. It is described in Chapter 34, "SMP/E reports," on page 457.

Usage notes

- If you want to add a subentry or a subentry value, you should use the ADD statement. Although, in some cases, you may get the desired results using the REP statement, in other cases, the results may not be what you expected. For more information, see "Processing" on page 389.
- If you want to delete a subentry or a subentry value, do not try to do it by adding a null value. For example, do not use ADD *subentry* `()`. Use the DEL statement instead.
- For subentries and subentry lists, if you want to delete all the values but do not know what they all are, you can specify the subentry or subentry list operand followed by a left and right parenthesis. For example, to delete all MOD values in the SYSMOD entry for UR11111, you could use these commands:

```
SET      BDY(TGT1)       /* Set to target zone.      */.
UCLIN                    /*                          */.
DEL      SYSMOD(UR11111)/* Delete data from SYSMOD: */
         MOD()           /* all MOD subentries.      */
                         /*                          */.
ENDUCL                   /*                          */.
```

The parentheses mean that SMP/E should delete the subentry or subentry list they enclose, even though no value is specified.

Note: This procedure can also be used with REP. As with DEL, there must be a current existing value.

Examples

The following general examples are provided to help you use the UCLIN command.

For specific examples of UCLIN for each entry type, see *z/OS SMP/E Reference*.

Example 1: UCLIN to change a global zone entry

You can use the following commands to change global zone entries:

```
SET      BDY(GLOBAL)        /* Set to global zone.      */.
UCLIN                       /*                          */.
...                         /*                          */
... UCL statements for global zone entries
...                         /*                          */
ENDUCL                      /*                          */.
```

Example 2: UCLIN to change a target zone entry

You can use the following commands to change target zone entries:

```
SET      BDY(TGT1)          /* Set to target zone.      */.
UCLIN                       /*                          */.
...                         /*                          */
... UCL statements for target zone entries
...                         /*                          */
ENDUCL                      /*                          */.
```

Example 3: UCLIN to change a distribution zone entry

You can use the following commands to change distribution zone entries:

```
SET      BDY(DLIB1)         /* Set to DLIB zone.        */.
UCLIN                       /*                          */.
...                         /*                          */
... UCL statements for DLIB zone entries
...                         /*                          */
ENDUCL                      /*                          */.
```

Example 4: UCLIN to delete an ORDER entry in the global zone

You can use the following commands to delete an ORDER entry in the global zone:

```
SET      BDY(GLOBAL)         /* Set to GLOBAL zone        */.
UCLIN                              /* Start UCLIN processing    */.
  DEL ORDER(ORD00035)            /*  Delete ORDER              */.
ENDUCL                           /*  End UCLIN processing     */.
```

Example 5: UCLIN to change an ORDER retention subentry in an OPTIONS entry in the global zone

You can use the following commands to change the Order Retention (ORDERRET) to 0 days:

```
SET      BDY(GLOBAL)         /* Set to GLOBAL zone        */.
UCLIN                              /* Start UCLIN processing    */.
  REP OPTIONS(SOMEOPTS)          /*                           */
      ORDERRET(0)                          /* Change ORDERRET to 0 days  */.
ENDUCL                           /*  End UCLIN processing     */.
```

Example 6: UCLIN statements using the FIXCAT subentry in various OPTIONS entries in the global zone

You can use the following commands to create a new OPTIONS entry with FIXCAT subentries in the global zone:

```
SET      BDY(GLOBAL)               /* Set to GLOBAL zone            */.
UCLIN                                    /* Start UCLIN processing        */.
  ADD OPTIONS(OPTENT)                 /* Identify the OPTIONS        */
      FIXCAT(                             /* Fix categories                */
                IBM.Device.zIPP            /*                                  */
                IBM.Device.2094.z/OS)    /*                                   */.
ENDUCL                           /*  End UCLIN processing       */.
```

You can use the following commands to replace an existing FIXCAT subentry with a new list of fixes:

```
SET      BDY(GLOBAL)                /* Set to GLOBAL zone            */.
UCLIN                                     /* Start UCLIN processing        */.
  REP OPTIONS(OPTENT)                  /* Identify the OPTIONS        */
      FIXCAT(                            /* Fix categories                */
                IBM.Function.DataSharing.MVS/ESA    /*
*/.
ENDUCL                             /*  End UCLIN processing        */.
```

You can use the following commands to delete a FIXCAT subentry from the OPTIONS entry:

```
SET      BDY(GLOBAL)                /* Set to GLOBAL zone            */.
UCLIN                                     /* Start UCLIN processing        */.
  DEL OPTIONS(OPTENT)                  /* Identify the OPTIONS        */
      FIXCAT()                           /* ..delete its FIXCATs        */.
ENDUCL                             /*  End UCLIN processing        */.
```

You can use the following commands to delete the entire OPTIONS entry:

```
SET      BDY(GLOBAL)                /* Set to GLOBAL zone            */.
UCLIN                                     /* Start UCLIN processing        */.
  DEL OPTIONS(OPTENT)                  /* Identify the OPTIONS        */
                                           /* ...delete the entry           */.
ENDUCL                             /*  End UCLIN processing        */.
```

Processing

SMP/E processes each UCL statement and each operand on each UCL statement as they are encountered. Table 25 on page 389 shows how these statements are processed.

Table 25. How SMP/E processes UCL statements

Type specified	ADD	DEL	REP
Entry	If the entry does not exist, SMP/E creates a new one. If the entry exists, SMP/E assumes a new subentry or subentry value is being added.	If the entry exists, SMP/E deletes it. Otherwise, an error occurs.	If the entry does not exist, SMP/E creates a new one. If the entry exists, SMP/E assumes that a new subentry or subentry value is being replaced.
Subentry	If the subentry does not exist in the entry, SMP/E adds the new subentry. Otherwise, an error occurs.	If the subentry exists in the entry, SMP/E deletes the subentry. Otherwise, an error occurs.	If the subentry exists in the entry, SMP/E replaces the current value with the new value. If the subentry does not exist in the entry, SMP/E adds the new subentry.
Subentry list or subentry list value	If the subentry list does not exist in the entry, SMP/E adds it. Likewise, if the subentry list value does not exist in the entry, SMP/E adds it. Otherwise, an error occurs.	If the subentry list value exists in the entry, SMP/E deletes that value. If all the existing subentry list values were specified, or if a null value was specified, SMP/E deletes the entire subentry. Otherwise, an error occurs.	SMP/E does **not** replace individual values in a subentry list. If the subentry list exists in the entry, SMP/E replaces **all** the current values with the new value. If the subentry list does not exist in the entry, SMP/E adds it.

Table 25. How SMP/E processes UCL statements (continued)

Type specified	ADD	DEL	REP
Notes for subentry list or subentry list value: • ADD cannot be used to add a subentry value to a CONCAT subentry list for a DDDEF entry, because the order of values in the subentry list is important for CONCAT. Adding a single value to an existing list does not provide enough information as to how the whole list should be ordered. • REP must be used to logically add a new value to a CONCAT subentry list for a DDDEF entry, because the order of the subentry list values is important for CONCAT, and REP is the only provided method for specifying the desired order. (REP replaces the entire subentry list; therefore, you must specify all the existing subentry values, as well as the new subentry value.)			
Subentry indicator	If the subentry indicator is not set, SMP/E sets it to "on." Otherwise, an error occurs.	If the subentry indicator is set to "on," SMP/E resets it to "off." Otherwise, an error occurs.	SMP/E sets the indicator to "on," regardless of its current setting.

For element and SYSMOD entries, SMP/E first copies the entry into storage. Then, as SMP/E encounters each operand, it updates the copy of the entry. When SMP/E reaches the end of the UCL statement, it makes sure the updated entry contains at least the minimum data required and that the data in that entry is consistent. If processing was successful, or if only informational or warning messages were issued, SMP/E replaces the original entry with the updated copy.

If an error occurs while one of the UCL statements is being processed, SMP/E does not make the change for that one statement. However, it continues to process subsequent UCL statements and, if they are valid, makes the requested changes. For each error, SMP/E issues an error message indicating the cause of the problem and deletes the copy of the entry. The original entry remains unchanged.

Note: For entries other than element entries or FEATURE, PRODUCT, or SYSMOD entries, if UCLIN deletes all the subentries for that entry, SMP/E deletes the entire entry.

For each element entry successfully changed by a UCL statement, SMP/E sets the LASTUPD subentry to `UCLIN` and the LASTUPDTYPE subentry to either `ADD` or `UPD`.

The return code for all the UCLIN processing is the highest return code from the processing of any of the UCL statement in the UCLIN input.

UCLIN processing stops when the ENDUCL command is encountered.

Zone and data set sharing considerations

The following identifies the phases of UCLIN processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, "Sharing SMP/E data sets," on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: Either the global zone, target zone, or the distribution zone is accessed, based on the zone specified in the previous SET command.

2. UCLIN processing

Global zone
Update with exclusive enqueue.

SMPPTS
Update with exclusive enqueue.

Target zone
Update with exclusive enqueue.

DLIB zone
Update with exclusive enqueue.

Note: Either the global zone and SMPPTS, the target zone, or the distribution is accessed, based on the zone specified in the previous SET command.

3. Termination

   All resources are freed.

Chapter 25. The UNLOAD command

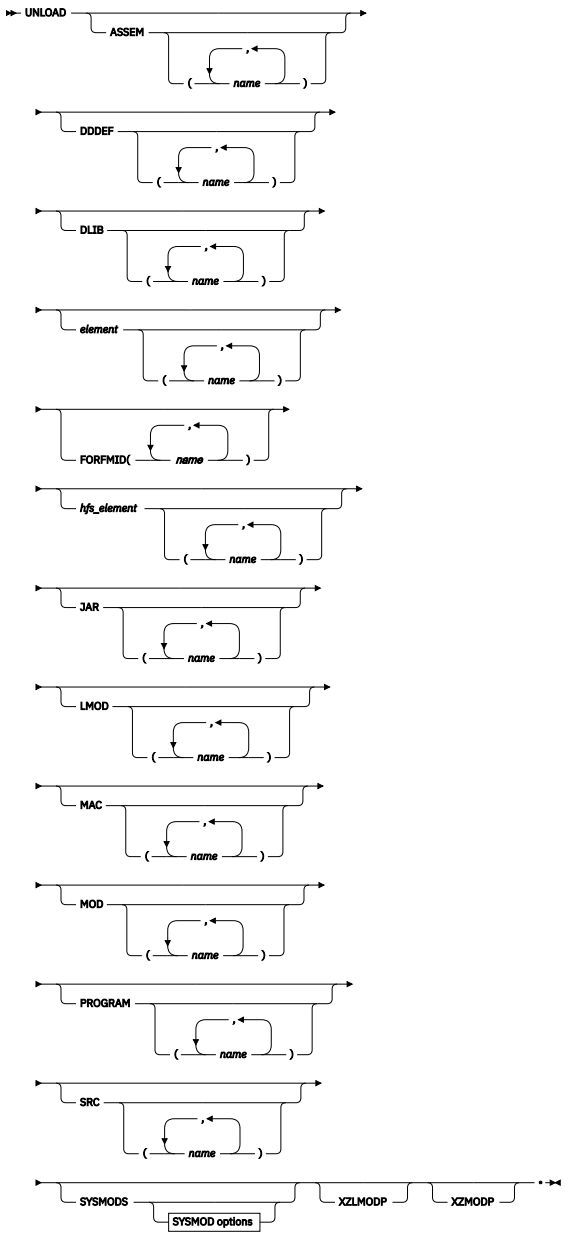
The UNLOAD command causes SMP/E to unload entries from the target zone or distribution zone to the SMPPUNCH data set. The output produced is in the form of UCL statements, which, when processed by SMP/E, re-create the unloaded entries in the distribution zone or the target zone. This function allows the user to unload selected parts of a target zone or distribution zone for initialization of entries on other zones. Do **not** use UNLOAD to back up a zone. Use ZONECOPY or ZONEEXPORT/ZONEIMPORT instead.

Zones for SET BOUNDARY

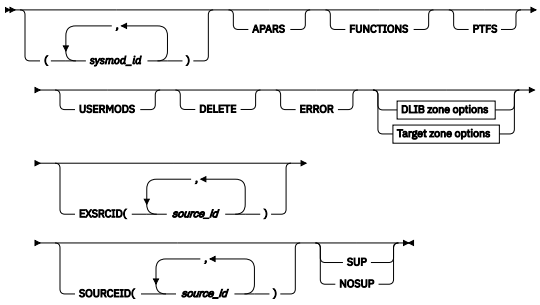
For the UNLOAD command, the SET BOUNDARY command must specify the name of the target or distribution zone containing the entries to be unloaded. You must ensure that the data you request to be unloaded is valid for the zone type specified.

Syntax

UNLOAD Command



SYSMOD options



DLIB zone options

Target zone options

Note:

1. The SYSMODS operand is optional if you specify any of the following operands:

APARS	NOACCEPT	RESTORE
DELETE	NOAPPLY	SOURCEID
ERROR	NOSUP	SUP
EXSRCID	PTFS	USERMODS
FUNCTIONS		

2. The XZLMODP and XZMODP operands are valid only for target zone entries.

See the operand descriptions for details.

Operands

APARS
indicates that SMP/E should unload APAR SYSMODs.

Note:

1. APARS can also be specified as APAR.
2. When APARS is used with FUNCTIONS, PTFS, or USERMODS, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.

ASSEM
indicates that SMP/E should unload all ASSEM entries or the specified ASSEM entries.

DDDEF
indicates that SMP/E should unload all DDDEF entries or the specified DDDEF entries.

DELETE
indicates that SMP/E should unload entries for function SYSMODs that have been explicitly deleted from the target zone or distribution zone by other function SYSMODs.

Note:

1. DELETE can also be specified as DEL.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD was also specified, even if it was not.

DLIB
indicates that SMP/E should unload all DLIB entries or the specified DLIB entries.

element
is used to unload a particular type of data element entry. It indicates that SMP/E should unload all data element entries of that type or the specified data element entries.

Note:

1. Data element entries exist only in the target and distribution zones.
2. The "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* shows the types of data elements that can be specified for the *element* operand.
3. Some types of elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *element* operand does not contain any *xxx*

value.) The "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* contains a table that shows the *xxx* values and the languages they represent.

ERROR
indicates that SMP/E should unload SYSMOD entries in which the ERROR indicator is set.

Note:

1. ERROR can also be specified as ERR.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD was also specified, even if it was not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

EXSRCID
indicates that SYSMODs associated with the specified source IDs should **not** be unloaded.

Note:

1. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
2. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are excluded.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU* are excluded.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are excluded.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are excluded.
     - A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for example, SYSMODs that contain any of these source IDs are excluded: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not excluded.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source ID specifications:

   ```
   RSU0709
   RSU*
   IBM.Device.20%4
   IBM.Device.*.zAAP
   ```

3. A given source ID may be explicitly specified **only once** on the EXSRCID operand.
4. The same source ID may **not** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified implicitly on the EXSRCID operand and also, either implicitly or explicitly, on the SOURCEID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs, if at least one of those source IDs is specified either implicitly or explicitly on the SOURCEID operand, and if another one of its source IDs is specified either implicitly or explicitly on the EXSRCID operand, the SYSMOD is excluded from processing.

   For example, assume PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.

7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.
8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

FORFMID
indicates that SMP/E should unload only entries currently owned by one of the specified FMIDs or by an FMID defined in one of the specified FMIDSET entries.

Note:

1. You can specify FMIDs, FMIDSET entries, or both.
2. Only element and SYSMOD entries are unloaded by the FORFMID operand.
3. Because this operand describes the type of SYSMOD to be listed, SMP/E processes it as though SYSMOD had also been specified, even if it has not, **unless** an element type operand was also specified. In that case, FORFMID limits the element entries that are unloaded.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

FUNCTIONS
indicates that SMP/E should unload function SYSMODs.

Note:

1. FUNCTIONS can also be specified as FUNCTION.
2. When FUNCTIONS is used with APARS, PTFS, or USERMODS, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.

hfs_element
is used to unload a particular type of hierarchical file system element entry. It indicates that SMP/E should unload all hierarchical file system element entries of that type or the specified hierarchical file system element entries.

Note:

1. Hierarchical file system element entries exist only in the target and distribution zones.
2. The "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* shows the types of hierarchical file system elements that can be specified for the *hfs_element* operand.
3. Some types of hierarchical file system elements, such as panels, messages, or text, may have been translated into several languages. In these cases, the *hfs_element* operand contains *xxx*, which represents the language used for the element. (If an element was not translated, the *hfs_element* operand does not contain any *xxx* value.) The "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference* contains a table that shows the *xxx* values and the languages they represent.

JAR
indicates that SMP/E should unload all JAR entries or the specified JAR entries.

LMOD
indicates that SMP/E should unload all LMOD entries or the specified LMOD entries.

MAC
indicates that SMP/E should unload all MAC entries or the specified MAC entries.

MOD
indicates that SMP/E should unload all MOD entries or the specified MOD entries.

NOACCEPT
indicates that SMP/E should unload SYSMOD entries from the current zone that are not accepted into the specified distribution zone.

Note:

1. NOACCEPT can also be specified as NOACC.
2. If a target zone is specified on the SET command and no distribution zone is specified on NOACCEPT, SMP/E uses the distribution zone from the RELATED subentry in the TARGETZONE entry.
3. If a distribution zone is specified on the SET command and no distribution zone is specified on NOACCEPT, SMP/E issues an error message.
4. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
5. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

NOAPPLY
indicates that SMP/E should unload SYSMOD entries from the current zone that are not applied to the specified target zone.

Note:

1. NOAPPLY can also be specified as NOAPP.
2. If a distribution zone is specified on the SET command and no target zone is specified on NOAPPLY, SMP/E uses the target zone from the RELATED subentry in the DLIBZONE entry.
3. If a target zone is specified on the SET command and no target zone is specified on NOAPPLY, SMP/E issues an error message.
4. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
5. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

NOSUP
indicates that SMP/E should unload entries for SYSMODs that have not been superseded.

Note:

1. NOSUP is mutually exclusive with SUP.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

PROGRAM
indicates that SMP/E should unload all program element entries or the specified program element entries.

PTFS
indicates that SMP/E should unload PTF SYSMODs.

Note:

1. PTFS can also be specified as PTF.
2. When PTFS is used with APARS, FUNCTIONS, or USERMODS, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.
3. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.

RESTORE
indicates that SMP/E should unload SYSMOD entries in which the RESTORE indicator is set. These SYSMODs have been incompletely restored and are “in error.”

Note:

1. RESTORE is allowed when the SET command specifies a target zone.
2. RESTORE can also be specified as RES.
3. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
4. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

SOURCEID
indicates that SMP/E should unload only those SYSMOD entries associated with one of the specified SOURCEID values.

Note:

1. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
2. There are two ways to specify source IDs:
   - Explicitly, by fully specifying a particular source ID (for example, RSU0711). In this case, all SYSMODs that contain the identified source ID are selected.
   - Implicitly, by partially specifying a source ID value using asterisks (*) as global characters and percent signs (%) as placeholders.
     - A single asterisk indicates that zero or more characters can occupy that position. Here are some examples:
       - For RSU*, all SYSMODs that contain a source ID that begins with the character string RSU are selected.
       - For *0711, all SYSMODs that contain a source ID that ends with the character string 0711 are selected.
       - For RSU*1, all SYSMODs that contain a source ID that begins with the character string RSU and ends with the character string 1 are selected.
     - A single percent sign indicates that any one single character can occupy that position. For RSU0%11, for instance, SYSMODs that contain any of these source IDs are selected: RSU0711, RSU0211, and RSU0311. SYSMODs that contain source ID RSU00711 are not selected.

   Any number of asterisks and percent signs can be used within a single partially specified source ID.

   The following examples are valid source IDs:

   ```
   RSU0709
   RSU*
   IBM.Device.20%4
   IBM.Device.*.zAAP
   ```
3. A given source ID may be explicitly specified **only once** on the SOURCEID operand.
4. The same source ID may **not** be explicitly specified on both the EXSRCID and SOURCEID operands.
5. If a source ID is specified implicitly on the SOURCEID operand and also, either implicitly or explicitly, on the EXSRCID operand, all SYSMODs with that source ID are excluded from processing.
6. If a given SYSMOD has multiple source IDs, if at least one of those source IDs is specified either implicitly or explicitly on the SOURCEID operand, and if another one of its source IDs is specified either implicitly or explicitly on the EXSRCID operand, the SYSMOD is excluded from processing.

   For example, suppose PTF UZ12345 has been assigned source IDs SMCREC and PUT0703. If you specify SOURCEID(SMC*) and EXSRCID(PUT0703), the SYSMOD is excluded from processing.
7. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

8. A source ID value might contain mixed case alphabetic characters. However, SMP/E ignores the case when identifying matches for a specified source ID value. For example, a specified source ID value of ABCDEF matches a value of abcdef.

SRC
indicates that SMP/E should unload all SRC entries or the specified SRC entries.

SUP
indicates that SMP/E should unload entries for SYSMODs that have been superseded.

Note:

1. SUP is mutually exclusive with NOSUP.
2. Because this operand describes the type of SYSMOD to be unloaded, SMP/E processes it as though SYSMOD had also been specified, even if it has not.
3. If no SYSMOD types are specified, all eligible SYSMODs are included. To process specific types of SYSMODs, you must specify the desired SYSMOD types.

SYSMODS
indicates that SMP/E should unload all SYSMOD entries or the specified SYSMOD entries.

You can limit which SYSMOD entries are unloaded by coding one or more of the following SYSMOD qualifier operands:

APARS, FUNCTIONS, PTFS, or USERMODS
DELETE
ERROR
EXSRCID
FORFMID
NOACCEPT
NOAPPLY
NOSUP or SUP
RESTORE
SOURCEID

Note:

1. SYSMODS can also be specified as SYSMOD.
2. If you specify any of the SYSMOD qualifier operands, SMP/E assumes that you want the SYSMOD entries unloaded and thus processes as if you had also entered SYSMOD.

USERMODS
indicates that SMP/E should unload USERMODs.

Note:

1. USERMODS can also be specified as USERMOD.
2. When USERMODS is used with APARS, FUNCTIONS, or PTFS, SMP/E unloads any SYSMOD whose type matches **any one** of those specified.

XZLMODP
indicates that SMP/E should unload MOD entries for all modules that have been linked into load modules controlled by a different target zone. (The MOD entries for these modules contain XZLMODP subentries.)

Note:

1. XZLMODP is allowed only when the SET command specifies a target zone.
2. The appropriate MOD entries are unloaded, regardless of whether the MOD operand was specified on the UNLOAD command.

3. If both MOD and XZLMODP are specified, only MODs with cross-zone subentries are unloaded. If a list of MODs and XZLMODP are specified, all the specified MODs, as well as all the MODs with cross-zone subentries, are unloaded.

XZMODP
indicates that SMP/E should unload LMOD entries for all load modules containing modules from a different target zone. (The LMOD entries for these load modules contain XZMODP subentries.)

Note:

1. XZMODP is allowed only when the SET command specifies a target zone.
2. The appropriate LMOD entries are unloaded, regardless of whether the LMOD operand was specified on the UNLOAD command.
3. If you specify both LMOD and XZMODP, only LMODs with cross-zone subentries are unloaded. If you specify a list of LMODs and XZMODP, all the specified LMODs, as well as all the LMODs with cross-zone subentries, are unloaded.

For examples of unloading each specific entry type, see the section for that entry in the "SMP/E Data Set Entries" chapter in *z/OS SMP/E Reference*.

Syntax notes

- Except where noted, you can specify multiple operands on a single UNLOAD command. This improves the overall performance of SMP/E.
- The order in which the entries are unloaded depends on their sequence in the appropriate SMP/E data set, not on the order specified on the UNLOAD command.
- You can mix mass requests and selective requests on the same UNLOAD command. For example:

```
SET      BDY(TGT1)      /* Set to target zone.      */.
UNLOAD   MOD            /* Unload all modules,      */
         MAC(MAC01      /* only two macros,         */
             MAC02)     /*                          */
         SRC(SRC01      /* only two source,         */
             SRC02)     /*                          */
         DLIB           /* all DLIBs,               */
         DDDEF          /* all DDDEFs,              */
         SYSMOD(UZ00001 /* only these five SYSMODs. */
                UZ00002 /*                          */
                UZ00003 /*                          */
                UZ00004 /*                          */
                UZ00005)/*                          */.
```

- If you specify a given SYSMOD on the SYSMODS operand, the SYSMOD **is** unloaded, regardless of whether it may be included or excluded by other operands.

Data sets used

The following data sets might be needed to run the UNLOAD command. You can define them by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOGA	SMPPUNCH	SMPSNAP
SMPCSI	SMPOUT	SMPRPT	*zone*
SMPLOG			

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

UNLOAD output is written to the SMPPUNCH data set. For an example of the UNLOAD output for a particular entry type, see the section for that entry type in the "SMP/E Data Set Entries" chapter in *z/OS SMP/E Reference*.

The following reports are produced during UNLOAD processing:

- File Allocation report
- UNLOAD Summary report

These reports are described in Chapter 34, “SMP/E reports,” on page 457.

Examples

For examples of the UNLOAD command and UNLOAD output for each entry type, see the "SMP/E Data Set Entries" chapter in *z/OS SMP/E Reference*.

Processing

Before SMP/E unloads any entries, it first determines what type of UNLOAD processing was requested:

- If no specific entries are specified (for example, if `UNLOAD.` is specified), it does mass-mode processing.

  Likewise, if specific entry types are specified, (for example, if `UNLOAD MAC MOD.` is specified to unload all the MAC and MOD entries in a target zone), SMP/E also does mass-mode processing.
- If specific entries are specified (for example, if `UNLOAD MAC(MACA,MACB) MOD(MODA,MODB).` is specified to unload specific MAC and MOD entries in a target zone), SMP/E does select-mode processing.

Note: A single UNLOAD command may combine mass-mode and select-mode processing.

Mass-mode processing

In mass-mode processing, SMP/E checks whether any entry types were specified. If so, SMP/E unloads all entries of the indicated type found in the set-to zone. SMP/E reads sequentially through the set-to zone. For each entry it finds, it checks whether the entry is of the specified type. If the entry meets all the requirements, SMP/E formats and prints the data.

If no entry types were specified, SMP/E unloads all the entries in the set-to zone and reads sequentially through that zone. For each entry it finds, it formats and prints the data.

Select-mode processing

In select-mode processing, SMP/E unloads all the specified entries found in the set-to zone. SMP/E goes directly to each of the entries specified and locates the data for the entry. For each entry it finds, it formats and prints the data.

Zone and data set sharing considerations

The following identifies the phases of UNLOAD processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

DLIB zone
Read without enqueue.

Note: Either the target zone or the distribution zone is used during setup according to the zone type specified in the previous SET command.

2. UNLOAD processing

Global zone
Read with shared enqueue.

Target zone
Read with shared enqueue.

DLIB zone
Read with shared enqueue.

Note: The zones used depend on the UNLOAD command operands, and the zone type specified in the previous SET command.

3. Termination

All resources are freed.

Chapter 26. The UPGRADE command

With the UPGRADE command, you can specify when SMP/E is permitted to make incompatible changes to SMP/E data sets.

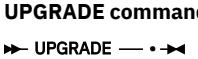
Note: Once the UPGRADE command has been run for a zone, there is no way to reverse the changes it makes to that zone.

Zones for SET BOUNDARY

For the UPGRADE command, the SET BOUNDARY command must specify the zone for which you want to permit SMP/E to make incompatible changes to the SMP/E data sets associated with the zone. The zone specified can be a target, distribution or global zone.

Syntax

UPGRADE command



Operands

This command has no operands.

Data sets used

The following data sets might be needed to run the UPGRADE command. They may be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA	SMPRPT	*zone*

Note: *zone* represents the DD statement required for the set-to zone to be processed by this command. If the DD statement is not specified, the data set is dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry.

Output

The UPGRADE command produces a File Allocation Report. See Chapter 34, “SMP/E reports,” on page 457 for a description of this report.

Example

Suppose an APPLY command fails because SMP/E detects that an incompatible change would be made to SMP/E data sets if the command were allowed to complete. You must decide if you are ready to make such changes or if you want to continue to use a prior release level of SMP/E that will not make those changes. Once you have decided you no longer need to use prior release levels of SMP/E to process the SMP/E data sets and are ready to make incompatible changes to your SMP/E data sets, you can use the UPGRADE command to allow SMP/E to make such changes.

```
SET BDY(ZOSTGT).
UPGRADE.
APPLY SOURCEID(RSU0203)
```

```
BYPASS(HOLDSYS)
CHECK.
```

In this example, the UPGRADE command indicates incompatible changes may be made to SMP/E data sets. If UPGRADE is **not** specified, then SMP/E stops any processing that would make incompatible changes to SMP/E data sets. In this example, the APPLY command would very likely fail if the UPGRADE command were not first run. However, once the UPGRADE command is run for a particular zone, then all SMP/E commands are authorized to make incompatible changes to all SMP/E data sets in that zone.

Processing

The UPGRADE command will record the release and PTF level of the currently running SMP/E in the UPGLEVEL subentry of the zone entry for the set-to zone only if it has not already been set to the current SMP/E release level or a higher value. If the UPGLEVEL subentry for the set-to zone is already set to the current SMP/E release level or a higher value, no further processing is necessary. So, for example, if the UPGRADE command is run using SMP/E Version 3 Release 2 (with no service applied) and the set-to zone has no UPGLEVEL subentry, or the UPGLEVEL subentry for the set-to zone is less than *SMP/E 32.00*, SMP/E updates the zone entry to set the UPGLEVEL to *SMP/E 32.00*. If the same zone is later processed by another UPGRADE command using a different SMP/E Version 3 Release 2 that has service applied, the higher service level SMP/E will update the UPGLEVEL subentry to reflect its PTF level (*SMP/E 32.11*, for example).

Zone and data set sharing considerations

The following identifies the phases of UPGRADE processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: The global zone is always opened for read during the Initialization phase. A target or distribution zone may also be opened for read, but only if it was specified in the previous SET command.

2. UPGRADE processing

   set-to zone
   Update with exclusive enqueue.

3. Termination

   All resources are freed.

Chapter 27. The ZONECOPY command

The ZONECOPY command can be used to copy an entire target or distribution zone from an existing CSI data set to another CSI data set. With ZONECOPY, you can copy a zone from a CSI containing multiple zones, or from a CSI containing just one zone. SMP/E copies the data from the input zone to the other CSI and renames the receiving zone.

ZONECOPY processing is similar to access method services (AMS) REPRO processing of the same data. To copy a single zone, or to copy a CSI containing just one zone, though, you should not use AMS REPRO —use ZONECOPY instead. In addition to copying the zone, ZONECOPY renames the copied zone for you. However, to copy a complete CSI containing multiple zones, you need to use AMS REPRO. SMP/E does not provide any commands that copy more than one zone at a time.

You can use ZONECOPY to copy the following input zones into the following receiving zones:

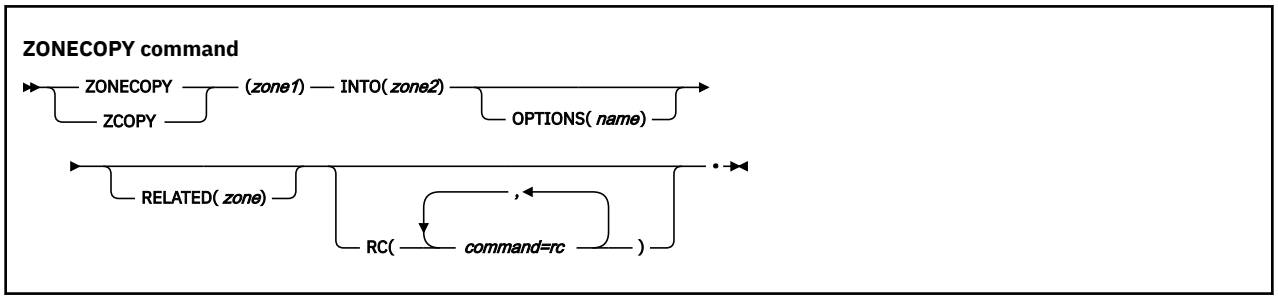
- A distribution zone into a distribution zone
- A distribution zone into a target zone
- A target zone into a target zone

Note: You cannot copy a target zone into a distribution zone. A global zone cannot be an input or receiving zone.

Zones for SET BOUNDARY

For the ZONECOPY command, the SET BOUNDARY command must specify which target or distribution zone should receive the copied zone. This name must match the name of the receiving zone specified on the INTO operand of the ZONECOPY command.

Syntax



Operands

zone1
specifies the name of the zone to be copied. This cannot be the global zone.

INTO
specifies the name of the zone to receive the copied data. This cannot be the global zone. You can specify the following combinations of input and receiving zones:

- Distribution zone to distribution zone
- Target zone to target zone
- Distribution zone to target zone

Note: A distribution zone cannot receive a target zone.

A ZONEINDEX subentry for the receiving zone must be specified in the global zone. Also, before you run the ZONECOPY command, make sure the receiving CSI has been allocated and has at least a GIMZPOOL record in it.

OPTIONS
specifies the name of the OPTIONS entry to use for processing the receiving zone. The OPTIONS subentry in the zone definition of the receiving zone is set to the specified name.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONECOPY command.

Before SMP/E processes the ZONECOPY command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONECOPY command. Otherwise, the ZONECOPY command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONECOPY command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RELATED
specifies the name of the target or distribution zone to which the receiving zone is related. The RELATED subentry in the zone definition of the receiving zone is set to the specified name.

Syntax notes

The input and receiving zones must meet **all** these conditions:

- Be defined in the same global zone
- Have different names
- Be located in different CSI data sets

If you specify the OPTIONS or RELATED operand, SMP/E either adds new subentries if none exist or replaces the existing subentries with the new ones before writing them to the receiving zone.

Data sets used

The following data sets might be needed to run the ZONECOPY command. They **must** be defined by DD statements. They must not be defined by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA	SMPRPT	*zone*

Note: *zone* represents the DD statements required for each distribution zone or target zone referred to in this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Updating cross-zone subentries

- If the copied zone contained any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the new zone and any identified cross-zones are properly related through cross-zone subentries. The type of corrective action required depends on:

- The types of cross-zone subentries found
- How you plan to use the new zone
- What you plan to do with the input zone
- Any actions you plan to perform (or may have already performed) with the zones identified in the warning messages

Here are some examples of possible situations and the associated corrective action:

- The new zone replaces the old zone, which will be deleted.

  You need to update the cross-zone subentries in each identified zone to indicate the new zone name. To do this, use the ZONEEDIT command.

- All the identified zones are being copied to create a clone of the original zone structure.

  You need to connect all the new zones correctly. To do this, use the ZONEEDIT command.

- The new zone contains only TIEDTO and XZMOD subentries, and you want to obtain updates for the identified load modules from the cross-zones.

  You need to update the identified zones with the information required to ensure that the identified modules are automatically linked into the affected load modules. Specifically, you need to add XZLMOD subentries to the MOD entries in the other zones, as well as a TIEDTO subentry to the TARGETZONE entry for each of these other zones. To add the XZLMOD subentries, use the UCLIN command. To add the TIEDTO subentries, use either the UCLIN command or the SMP/E Administration dialog.

- The new zone is a temporary copy and will be deleted after brief use.

  You can either do nothing or disconnect the new zone from all the identified zones. To disconnect the zone, use the ZONEEDIT command to delete all the cross-zone subentries from the new zone.

Output

The File Allocation report is produced during ZONECOPY processing. See Chapter 34, “SMP/E reports,” on page 457 for a description of this report.

Examples

The following examples are provided to help you use the ZONECOPY command.

Example 1: Copying a target zone to a target zone

Assume that you have a test system zone named CPYTEST in data set SMP.TEST.CSI and you need to copy this zone to another target zone, CPYPROD, for a production system. The following SMP/E commands create the new zone:

```
SET        BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                       /* Define zone index.       */.
ADD        GZONE            /*                          */
           ZINDEX(          /*                          */
                  (CPYPROD,SMP.PROD.CSI,TARGET)     /*  */
                 )          /*                          */
                            /*                          */.
ENDUCL                      /*                          */.
SET        BDY(CPYPROD)     /* Set to new zone.         */.
ZONECOPY   (CPYTEST)        /* Copy target CPYTEST to   */
           INTO(CPYPROD)    /* target zone CPYPROD.     */.
```

After the ZONECOPY operation, the contents of zone CPYTEST have been copied into zone CPYPROD.

Note: The two zones must be in different CSI data sets.

Example 2: Copying a distribution zone to a distribution zone

Assume that you have a distribution zone named TSTDLB1 in data set SMP.DLB1.CSI, with a related target zone TSTTGT1. You want to create a copy of TSTDLB1 to do some testing. The new distribution zone will be named TSTDLB2, will have a related target zone of TSTTGT2, and will be processed according to options entry TSTOPT. The following SMP/E commands create the new zone:

```
SET        BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                       /* Define zone index.       */.
ADD        GZONE            /*                          */
           ZINDEX(          /*                          */
                  (TSTDLB2,SMP.DLB2.CSI,DLIB)       /*  */
                 )          /*                          */
                            /*                          */.
ENDUCL                      /*                          */.
SET        BDY(TSTDLB2)     /* Set to new zone.         */.
ZONECOPY   (TSTDLB1)        /* Copy DLIB zone TSTDLB1   */
           INTO(TSTDLB2)    /* to DLIB zone TSTDLB2.    */
           OPTIONS(TSTOPT)  /* Add OPTIONS entry        */
           RELATED(TSTTGT2) /* and related zone.        */.
```

After the preceding SMP/E commands have completed, the global zone ZONEINDEX indicates that zone TSTDLB1 is still in SMP.DLB1.CSI and has not been changed. A new global zone ZONEINDEX subentry has been created to indicate that zone TSTDLB2 is in SMP.DLB2.CSI, with a new related zone TSTTGT2 and options entry TSTOPT.

Note: The two zones must be in different CSI data sets.

Example 3: Copying a distribution zone to a target zone

Assume that you have a distribution zone named CPYDLIB in data set SMP.DLB.CSI. You have used the distribution libraries described by this zone to do a system generation, and you now want to create a target zone describing the content of your new operating system. The new target zone is to be named CPYTGT and is to be in data set SMP.TGT.CSI. You also want to change the related zone from a target zone to a distribution zone. The following SMP/E commands create the new zone:

```
SET        BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                       /* Define zone index.       */.
ADD        GZONE            /*                          */
           ZINDEX(          /*                          */
                  (CPYTGT,SMP.TGT.CSI,TARGET)       /*  */
                 )          /*                          */
                            /*                          */.
ENDUCL                      /*                          */.
SET        BDY(CPYTGT)      /* Set to new zone.         */.
ZONECOPY   (CPYDLIB)        /* Copy DLIB zone CPYDLIB   */
           INTO(CPYTGT)     /* to target zone CPYTGT.   */
           RELATED(CPYDLB1)./* Change related zone.     */.
```

After the preceding SMP/E commands have completed, the global zone ZONEINDEX specifies that zone CPYDLIB is still in SMP.DLB.CSI and has not been changed. A new global zone ZONEINDEX subentry has been created and specifies that zone CPYTGT is in SMP.TGT.CSI, with a new related zone CPYDLB1.

Note: The two zones must be in different CSI data sets.

Processing

The ZONECOPY command copies a specified distribution or target zone from one CSI data set to another. SMP/E adds the new zone to the end of the receiving CSI data set. The input distribution or target zone remains in the input CSI data set.

Before copying the zone, SMP/E checks that the parameters entered are correct and that the copy request is valid. If any of the checks fail, SMP/E issues an error message, and the ZONECOPY command fails. SMP/E checks that:

- The input and receiving zones are defined in the same global zone.
- The input and receiving zones have different names.

- The input and receiving zones are in different CSI data sets.
- The combination of input and receiving zone types is valid.
- No cross-zone subentry from the input zone refers to a zone with the same name as the receiving zone.

If all the validity checking is successful, the zone can be copied. SMP/E opens the input zone for read access and opens the receiving zone for update processing. SMP/E then reads the data from the input zone and writes the data into the receiving zone.

If the input zone contained cross-zone subentries, SMP/E issues warning messages.

If the input and receiving zone types are different, SMP/E changes the zone type in the zone definition record before writing it to the receiving zone. When it reaches the end of the input zone file, SMP/E closes the input and receiving zones.

Zone and data set sharing considerations

The following identifies the phases of ZONECOPY processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: The target zones or distribution zones accessed during this phase are the input zone and the receiving zone.

2. ZONECOPY processing

   Target zone
   Read with shared enqueue (input zone).

   Target zone
   Update with exclusive enqueue (receiving zone).

   DLIB zone
   Read with shared enqueue (input zone).

   DLIB zone
   Update with exclusive enqueue (receiving zone).

   Note: The target or distribution zone specified as the input zone is opened for read access with a shared enqueue, and the target or distribution zone specified in the INTO operand is opened for update access with exclusive enqueue.

3. Termination

   All resources are freed.

Chapter 28. The ZONEDELETE command

There are times when it is necessary to delete the SMP/E data for one of the systems you are supporting. Examples of when this may become necessary are:

- After performing a full system generation or running the output from the SMP/E GENERATE command, you have to delete the information describing the previous target system libraries and rebuild that information to describe the new set of target system libraries built from the distribution libraries.
- After installing a new level of a product that existed in its own target zone and distribution zone, you want to delete the information about the old level of the product and continue processing only the new level.

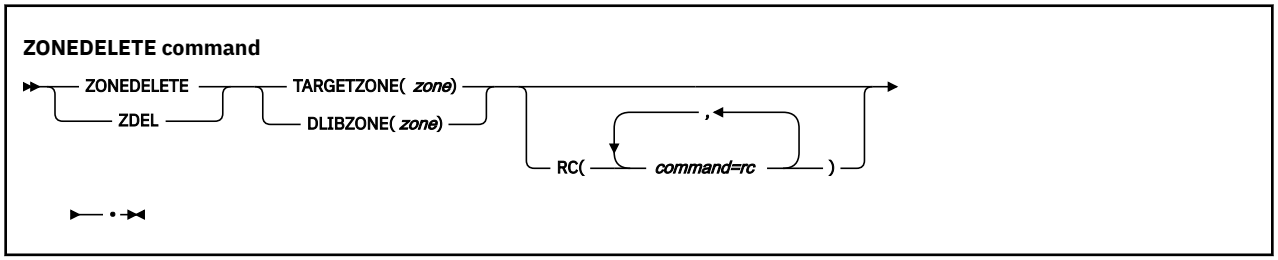
With the ZONEDELETE command, you can delete a specified target zone or distribution zone from the CSI in which it was contained.

Note: The CSI data set is not deleted; only the specified zone is deleted.

Zones for SET BOUNDARY

For the ZONEDELETE command, the SET BOUNDARY command must specify the name of the zone to be deleted. This same zone must also be specified on the ZONEDELETE command.

Syntax



Operands

DLIBZONE
specifies the distribution zone to be deleted.

Note:

1. DLIBZONE can also be specified as DZONE.
2. DLIBZONE is mutually exclusive with TARGETZONE.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEDELETE command.

Before SMP/E processes the ZONEDELETE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEDELETE command. Otherwise, the ZONEDELETE command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEDELETE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

TARGETZONE
specifies the target zone to be deleted.

Note:

1. TARGETZONE can also be specified as TZONE.
2. TARGETZONE is mutually exclusive with DLIBZONE.

Syntax notes

- For SMP/E to determine which CSI data set contains the zone to be deleted, there must already be a zone index for that zone.
- The zone type specified in the zone index must match the type specified on the ZONEDELETE command.

Data sets used

The following data sets might be needed to run the ZONEDELETE command. They can be defined by DD statements or, normally, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA	SMPRPT	*zone*

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

- Once a ZONEDELETE command has been successfully processed, the data cannot be recovered unless you have a backup copy of the CSI data set that contained the zone. Therefore, you should be very careful when entering the ZONEDELETE command. Make sure the zone specified is actually the one you want to delete; that is, check the spelling, and so on.
- Once the ZONEDELETE command has been successfully processed, all references to the zone are gone, and nothing more can be done with that zone. Therefore, the next command after ZONEDELETE **must** be a SET command.
- If the deleted zone contained any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the other zones have valid cross-zone entries so that SMP/E can function properly. The type of corrective action required depends on:
  - The types of cross-zone subentries found
  - Whether a copy was made of the deleted zone; if so, whether it is being used
  - Any actions you plan to perform (or may have already performed) with the zones identified in the warning messages

  Here are some examples of possible situations and the associated corrective action:

  - The deleted zone is being removed from the system and will not be replaced.

    You need to delete any cross-zone subentries in the identified zones that refer to the deleted zone. To do this, use the ZONEEDIT command.
  - The deleted zone was copied and will be replaced by the copy.

    You need to update the cross-zone subentries in the identified zones with the name of the replacement zone. To do this, use the ZONEEDIT command.

Output

The File Allocation report is produced during ZONEDELETE processing. This report is described in Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the ZONEDELETE command.

Example 1: Deleting a target zone

Assume that you have a test system zone named MVSTST1, and you have just performed a full MVS™ system generation to create a new set of libraries, and then you have done the processing needed to set up a new zone, MVSTST2, representing the new set of target libraries (for examples of setting up a target zone after system generation, see “Examples” on page 440). You are now ready to delete the old target zone:

```
SET      BDY(MVSTST1)        /* Process MVSTST1 tgt zone.*/.
ZDEL     TZONE(MVSTST1)      /* Delete it.               */.
```

Example 2: Deleting a distribution zone

Assume that you are running IMS in your installation and that the IMS product exists in its own target zone and distribution zone. You are in the process of migrating from IMS-x to IMS-x+1. IMS-x+1 has been installed into its own distribution zone (that is, not into the distribution zone that IMS-x was in), testing has been completed, and you now want to delete IMS-x. Assume the name of the distribution zone containing IMS-x is IMSX, and the name of the distribution zone containing IMS-x+1 is IMSXP1. The following commands delete the IMS-x distribution zone:

```
SET      BDY(IMSX)           /* Process old IMS zone.    */.
ZDEL     DZONE(IMSX)         /* Delete it.               */.
```

Processing

When a ZONEDELETE command is encountered, SMP/E ensures that you made no mistake in entering the command, as follows:

- SMP/E checks the operand on the ZONEDELETE command (that is, TARGETZONE, TZONE, DLIBZONE, or DZONE) and the zone type specified in the global zone ZONEINDEX to ensure that the zone types match. If they do not, an error condition is reported, and the ZONEDELETE request is not processed.
- SMP/E checks the name specified on the ZONEDELETE command operand to ensure that it matches the names specified in the previous SET command. If not, an error condition is reported, and the ZONEDELETE request is not processed.

If the deleted zone contained cross-zone subentries, SMP/E issues warning messages.

If no verification errors occur, SMP/E deletes the specified zone from the CSI data set it was contained in. After deleting the target zone or distribution zone from the CSI data set, SMP/E deletes the global zone ZONEINDEX entry for that zone. SMP/E now has no references to the deleted zone.

Zone and data set sharing considerations

The following identifies the phases of ZONEDELETE processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

Target zone
Read without enqueue.

DLIB zone
Read without enqueue.

Note: Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

2. Delete zone records

Target zone
Update with exclusive enqueue.

DLIB zone
Update with exclusive enqueue.

Note: Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

3. Delete zone index entry

Global zone
Update with exclusive enqueue.

4. Termination

All resources are freed.

Chapter 29. The ZONEEDIT command

The ZONEEDIT command can be used instead of multiple UCL statements to make mass changes in selected SMP/E entries in the same zone. You can also use the command add certain subentries to selected SMP/E entries in the same zone. Here are some examples of when you might want to use the ZONEEDIT command:

- To change all the volume serial numbers in all the DDDEF entries in a specified zone
- To change the ddname for the print output data set in all the UTILITY entries in the global zone
- To change a zone name in all the cross-zone subentries in a specified target zone
- To modify path names of DDDEF entries during the service process for OS/390 or z/OS UNIX System Services
- To add a UNIT value to all DDDEF entries in the specified zone that do not currently have a UNIT value

Zones for SET BOUNDARY

For the ZONEEDIT command, the SET BOUNDARY command must specify the appropriate type of zone for the entry type that is changed.

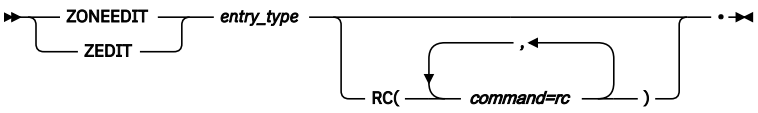
- For DDDEF entries, it must specify the appropriate global, target, or distribution zone.
- For UTILITY entries, it must specify the global zone.
- For cross-zone subentries, it must specify the appropriate target zone.

Syntax

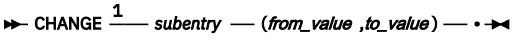
Three types of commands are necessary for ZONEEDIT processing:

1. The ZONEEDIT command indicates the start of ZONEEDIT processing.
2. ZONEEDIT CHANGE statements (for unconditional changes) and IF…THEN CHANGE statements (for conditional changes) show the changes to be made.
3. The ENDZONEEDIT command indicates the end of the ZONEEDIT commands and the end of ZONEEDIT processing.

ZONEEDIT command



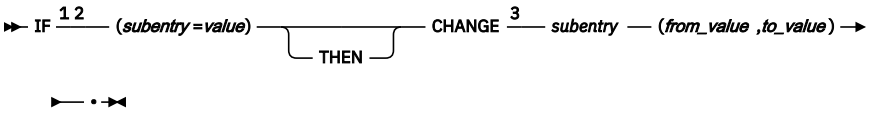
CHANGE statement



Notes:

[1] You can specify more than one CHANGE statement.

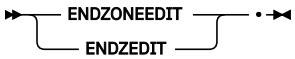
Conditional CHANGE statement



Notes:

[1] You can specify more than one IF statement.
[2] Wildcard characters are permitted within an IF statement. If either the '*' or '%' character is part of the explicitly specified value, the value must be enclosed in single quotation marks.
[3] The PATH and XZENTRIES subentries may not be specified on the conditional CHANGE statement.

ENDZONEEDIT command



Operands

entry-type
identifies the type of entry to be changed. The valid types are:

- DDDEF (for a global, distribution, or target zone)
- UTILITY (for the global zone only)
- XZENTRIES (for a target zone only)

Note: XZENTRIES is not an actual entry type. It is used to change a zone name in all the cross-zone subentries in the specified zone:

- XZLMOD subentries in MOD entries
- XZMOD subentries in LMOD entries
- TIEDTO subentries in the TARGETZONE entry

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEEDIT command.

Before SMP/E processes the ZONEEDIT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEEDIT command. Otherwise, the ZONEEDIT command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC Operand".

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEEDIT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

CHANGE
specifies the subentry to be changed, its old value, and its new value.

Note:

1. CHANGE can also be specified as C.
2. If you specify several subentries on a CHANGE statement, use blanks, not commas as separators.

subentry
specifies the type of subentry to be changed. These are the values that can be specified:

- For DDDEF entries:

  DATASET or DA
  PATH
  SYSOUT
  UNIT

VOLUME
WAITFORDSN or WAIT

- For UTILITY entries:

  NAME(*utility*)
  PRINT

- For XZENTRIES:

  ZONEVALUE

  Note: ZONEVALUE is not an actual subentry type. It is used to change a zone name in all the cross-zone subentries in the specified zone. You cannot change ZONEVALUE to the name of the set-to zone.

from-value
specifies the current value of the subentry. There are four ways to specify this value:

- Explicitly, by fully specifying the subentry value.

  Note:

  1. If an explicitly specified SYSOUT *from_value* consists of an asterisk (*), the *from_value* must be enclosed in single quotation marks, otherwise SMP/E treats the symbol as a wildcard character.
  2. A UNIT *from_value* might contain any character except the asterisk (*), percent (%), or single quotation mark (').
  3. Specifying a *from_value* of 'NO' for the WAITFORDSN subentry is equivalent to specifying the null value.

- *, which is a wildcard character that indicates that the specified subentry is to be changed, regardless of its value.

  If there is no current value for a UNIT, VOLUME, WAITFORDSN or PRINT subentry, an explicitly specified *to_value* will be added to the entry. For all other subentries, or if the *to_value* is not explicitly specified, the subentry will not be added.

  Note:

  1. Wildcard characters are not allowed for XZENTRIES.
  2. See “Specifying a pathname on the CHANGE PATH statement” on page 421 for additional considerations on using wildcard characters for PATH subentries.

- '', represents a null value. It consists of two single quotation marks with no blanks between them and indicates the specified subentry is to be added to all entries of the specified type that do not currently have a subentry value. This is used for UNIT, VOLUME, WAITFORDSN, and PRINT subentries.

  Note: The UNIT, VOLUME, and WAITFORDSN values will only be added to DDDEF entries which contain the DATASET subentry.

- *char*.*, where *char* is a character string you specify. This can be used for data set names and pathnames and means SMP/E must change all names beginning with the specified string.

to-value
specifies the new value of the subentry. There are three ways to specify this value:

- Explicitly, by fully specifying the subentry value.

  Note:

  1. If an explicitly specified SYSOUT *to_value* consists of an asterisk (*), the *to_value* must be enclosed in single quotation marks, otherwise, SMP/E treats the symbol as a wildcard character.
  2. A UNIT *to_value* can contain any character except the asterisk (*), percent (%), or single quotation mark (').

3. A *to_value* of 'NO' for the WAITFORDSN subentry is equivalent to specifying a *to_value* of '*'.

- *, which erases the current subentry value.

  Note:

  1. This is not allowed for PATH subentries.
  2. This is not allowed when you add a subentry to those entries that do not currently have a value for that particular subentry (when the *from_value* represents a null value).

- *char*.*, where *char* is a character string you specify. This can be used for data set names and pathnames and means SMP/E must change all values meeting the CHANGE condition so they begin with the specified string.

  Note: This is not allowed when you add a subentry to those entries that do not currently have value for that particular subentry (when the *from_value* represents a null value).

Note: Except for PATH subentries, coding CHANGE *subentry*(*,*) . deletes all values for that subentry. You should be sure, therefore, that is what you want when you use this form of the CHANGE statement. (CHANGE PATH(*,*) . is not allowed.)

IF ... THEN

specifies another subentry of the specified entry type that SMP/E is to check before making the change that follows. THEN is optional.

value

specifies the current value for another subentry of the specified entry type. Only entries containing this particular subentry value will be modified. The value can be specified in two ways:

- Explicitly, by fully specifying the particular value. If an explicitly specified SYSOUT *value* consists of an asterisk (*), the *value* must be enclosed in single quotation marks, otherwise, SMP/E treats the symbol as a wildcard character.
- Implicitly, by partially specifying a value using asterisks (*) as global characters and percent signs (%) as placeholders.
  - A single asterisk indicates that zero or more characters can occupy that position. For example, (UNIT=*R1) or (DATASET=SYS1.*DATA). In the first case, any DDDEF entries containing a UNIT value that ends with the character string 'R1' will be selected for processing. That would include values like SYSR1, DLIBR1 and ABCDEFR1. In the second case, any DDDEF entries containing a DATASET value that starts with the character string 'SYS1.' and ends with the character string 'DATA' will be selected for processing. That would include values like SYS1.LIBRARY.DATA, SYS1.LIBRARY.INFODATA and SYS.LIBRARY.STUFF.DATA.
  - A single percent sign indicates that any one single character can occupy that position. For example. (UNIT=SYS%1) OR (DATASET=SYS%.MIGLIB). In the first case, any DDDEF entries containing any of the following UNIT values will be selected for processing, SYSR1, SYSQ1 and SYSX1. DDDEF entries would not be selected for containing UNIT value SYSRES. In the second case, any DDDEF entries containing any of the following DATASET values will be selected for processing, SYS1.MIGLIB, SYS2.MIGLIB and SYSX.MIGLIB.

Any number of asterisks and percent signs may be used within a single partially specified subentry value. The following examples are valid conditional specifications:

- (DATASET=SYS%.*.DATA)
- (UNIT=SYSR1)
- (DATASET=SYS1.*)
- (UNIT=SYS*D%)

SMP/E will process all entries of the specified type that contain a subentry matching the specified explicit value or implicit pattern.

Note: Conditional changes are not allowed for XZENTRIES or PATH.

Syntax notes

- The subentry names must match the existing UCLIN subentry names.
- Make sure to specify an existing subentry value, or represent a null value. Otherwise, you might get unexpected results.

Specifying a pathname on the CHANGE PATH statement

The specific pathname syntax rules for the ZONEEDIT CHANGE PATH statement are:

- On both the from-value and to-value parameters, the pathname can be from 1 to 255 characters. If a wildcard character (*) is used, the pathname that results from processing the wildcard must not be greater than 255 characters in length.
- If a wildcard character (*) is not used, a full pathname must be specified, and that pathname must begin and end with a slash (/).
- If a wildcard character (*) is used, either a full or partial pathname may be specified, and that pathname must begin with a slash (/) and end with the wildcard character (*).
- If a wildcard character (*) is used in the from-value, and the character immediately preceding the wildcard is a slash (for example, /abc/*), then the to-value must also end in a slash.
- No more than one wildcard character (*) may be used in a pathname and it must always be the last character in the pathname.
- A pathname must be enclosed in single apostrophes (') if any of the following is true:
  - The pathname contains lowercase alphabetic characters.
  - The pathname contains a character that is not uppercase alphabetic, numeric, national ($, #, or @), slash (/), plus (+), hyphen, period, or ampersand (&).
  - The pathname spans more than one line in the CHANGE statement.
- The apostrophes must be outside the required slashes, as in `'/pathname/'`, not `'/pathname'/` or `'/pathname'*`, not `'/pathname*'`.
- The single apostrophes used to enclose a pathname (the delimiters) do not count as part of the 255-character limit.
- Any apostrophes specified as part of a pathname (not the delimiters) must be doubled. Such doubled apostrophes count as two characters toward the 255-character limit.
- The pathname can include characters X'40' through X'FE'.

Table 26 on page 421 describes variations in the way a from-value and a to-value may be specified, and describes how a wildcard character (*) may be used to simplify changing a pathname. Unless otherwise noted, any combination of from-value and to-value are valid.

Table 26. From-value and to-value variations and wild cards

From-value	To-value
The actual subentry value -- a complete path name.	The actual subentry value -- a complete path name.
'/char'* or '/char/'*, where char is a character string you specify. This means SMP/E must change all pathnames beginning with the specified string.	'/char'* or '/char/'*, where char is a character string you specify. This means SMP/E must change all values meeting the CHANGE condition so they begin with the specified string. This wildcard specification may not be used in the to-value unless it is also used in the from-value. If a wildcard character (*) is used in the from-value, and the character immediately preceding the wildcard is a slash (for example, /abc/*), then the to-value must also end in a slash.

Table 26. From-value and to-value variations and wild cards (continued)	
From-value	**To-value**
*, which means change the specified subentry regardless of its value. If there is no current value for the subentry, the change is not made.	* is not allowed in the to-value for PATH subentries.

SMP/E will inform you if no changes were made by the CHANGE statement because the subentry value specified does not exist (for example, change XYZ to ABC, but XYZ does not exist) or because no entries of the specified type exist in the indicated zone (that is, change DDDEF but no DDDEFs are in the zone).

Data sets used

These data sets might be needed to run the ZONEEDIT command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA	SMPRPT	*zone*

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Output

Two reports are produced during ZONEEDIT processing:

- File Allocation report
- ZONEEDIT Summary report

These reports are described in Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the ZONEEDIT command.

Example 1: Editing DDDEF entries

In this example, the UNIT value for all the DDDEF entries is to be changed to 3350, regardless of its current value. Additionally, a UNIT subentry of 3350 will be added to all DDDEF entries which contain a DATASET subentry but do not currently have a UNIT subentry. The following ZONEEDIT command makes this change:

```
SET         BDY (TZONE1)      /* Set to zone to edit.    */.
ZONEEDIT    DDDEF             /* Edit DDDEF entries.     */.
CHANGE      UNIT(*,3350)      /* Change unit to 3350 and  */.
                              /* add UNIT if none exists. */.
ENDZONEEDIT                   /* End of ZONEEDIT.        */.
```

Example 2: Conditionally editing DDDEF entries

In this example, only DDDEF entries for data sets on the SYSRES volume are to be changed. UNIT values of 3330 are to be changed to 3350, and DATASET names that start with SYS1 are to start with SYS2. The following ZONEEDIT command makes this change:

```
SET        BDY (TZONE1)      /* Set to zone to edit.    */.
ZONEEDIT   DDDEF             /* Edit DDDEF entries      */.
IF         (VOLUME=SYSRES)   /* for SYSRES volume only.  */
CHANGE     UNIT(3330,3350)   /* Change unit from 3330,   */
           DATASET(SYS1.*,
                   SYS2.*)   /* data set name from SYS1. */.
ENDZONEEDIT                  /* End of ZONEEDIT.         */.
```

Example 3: Changing the SYSOUT value

In this example, the SYSOUT value for DDDEF entries is to be changed from * to A. The following ZONEEDIT command makes this change:

```
SET        BDY (TZONE1)     /* Set to zone to edit.       */.
ZONEEDIT   DDDEF            /* Edit DDDEF entries         */.
CHANGE     SYSOUT('*',A)    /* Change SYSOUT from * to A. */.
ENDZONEEDIT                 /* End of ZONEEDIT.           */.
```

There must be single quotation marks around the * if SMP/E is to change only DDDEF entries in which SYSOUT=*.

To change **all** the SYSOUT values to A, the following ZONEEDIT command can be used:

```
SET        BDY (TZONE1)     /* Set to zone to edit.     */.
ZONEEDIT   DDDEF            /* Edit DDDEF entries.      */.
CHANGE     SYSOUT(*,A)      /* Change all SYSOUT to A.  */.
ENDZONEEDIT                 /* End of ZONEEDIT.         */.
```

Because there are no quotation marks around the *, **all** the SYSOUT values are changed. SYSOUT subentries will not be added to those DDDEF entries which do not currently have SYSOUT subentries. Only PRINT, UNIT, VOLUME, and WAITFORDSN subentries can be added using the ZONEEDIT command.

Example 4: Changing the zone value in cross-zone subentries

Assume that you have used the LINK command to link modules from target zone CICS1 into load modules in target zone TZONE2. Later, because of new zone naming conventions, you used the ZONERENAME command to rename zone CICS1 to CICSPRD. During ZONERENAME processing, zone TZONE2 was identified as the only zone connected to zone CICS1 by cross-zone subentries.

You now need to change cross-zone subentries in zone TZONE2 to show that TZONE2 is now connected to the renamed zone, CICSPRD. The following ZONEEDIT command makes this change:

```
SET      BDY (TZONE2)       /* Set to zone to edit.       */.
ZONEEDIT XZENTRIES        /* Edit cross-zone subentries.*/.
CHANGE   ZONEVALUE(CICS1  /* Change zone from old name  */
                CICSPRD)  /* to new name.               */.
ENDZONEEDIT               /* End of ZONEEDIT.           */.
```

Example 5: Changing the PATH value of DDDEF entries

In this example, DDDEF entries that describe a PATH in a UNIX file system are to be changed. Specifically, any PATH that starts with "/usr/lpp/" will be changed to "/service/usr/lpp/". The following ZONEEDIT command makes this change:

```
SET BDY(TZONE1)          /* Set to zone to edit.       */.
ZONEEDIT DDDEF           /* Edit DDDEF entries.        */.
CHANGE PATH('/usr/lpp/'*,
            '/service/usr/lpp/'*)
                         /* Add /service to /usr/lpp/  */
                         /* directories.               */.
ENDZONEEDIT              /* End of ZONEEDIT.           */.
```

If the wildcard character (*) is used on a pathname, it must appear outside the enclosing apostrophes, as shown previously.

Example 6: Adding VOLUME, WAITFORDSN, and UNIT values to DDDEF entries

In this example, UNIT, WAITFORDSN and VOLUME values are being added to DDDEF entries that do not currently have a UNIT, VOLUME or WAITFORDSN value. Because UNIT, VOLUME and WAITFORDSN values can only be added to DDDEF entries that have a DATASET subentry:

- A UNIT value of 3350 will be added to all DDDEF entries that have a DATASET subentry but do not currently have a UNIT value.
- A WAITFORDSN value of 'YES' will be added to all DDDEF entries that have a DATASET subentry but do not currently have a WAITFORDSN value.
- A VOLUME value of SYSRSA will be added to all DDDEF entries that have a DATASET subentry value beginning with 'SYS1.', but do not currently have a VOLUME value.
- A VOLUME value of SYSRSB will be added to all DDDEF entries that have a DATASET subentry value beginning with 'SYS2.', but do not currently have a VOLUME value.

The following ZONEEDIT command makes this change:

```
SET      BDY(TZONE1)         /* Set to zone to edit.        */.
ZONEEDIT DDDEF               /* Edit DDDEF entries.         */.
CHANGE   UNIT('',3350)       /* Add UNIT value of 3350      */.
CHANGE   WAITFORDSN('', YES) /* Add WAITFORDSN value of YES */.
IF (DATASET=SYS1.*) THEN CHANGE VOLUME('', SYSRSA) /* ADD VOLUME
                                   value of SYSRSA TO all DDDEFs
                                   with SYS1 hlq and no existing
                                   VOLUME value.            */.
IF (DATASET=SYS2.*) THEN CHANGE VOLUME('', SYSRSB) /* ADD VOLUME
                                   value of SYSRSB to all DDDEFs
                                   with SYS2 hlq and no existing
                                   VOLUME value.            */.
ENDZONEEDIT                  /* End of ZONEEDIT             */.
```

Example 7: Adding a PRINT value to UTILITY entries

In this example, a PRINT value of MYOUTPUT is being added to all UTILITY entries that have a NAME subentry value that begins with the character string 'IEB' and which do not currently have a PRINT value. The following ZONEEDIT command makes this change:

```
SET      BDY(TZONE1)         /* Set to zone to edit.        */.
ZONEEDIT UTILITY             /* Edit UTILITY entries        */.
IF       (NAME=IEB*)         /* For all IEB utilities       */
CHANGE   PRINT('',MYOUTPUT)  /* Add PRINT value of MYOUTPUT */.
ENDZONEEDIT                  /* End of ZONEEDIT             */.
```

To add a PRINT value of MYOUTPUT to all UTILITY entries that do not currently have a PRINT value, the following ZONEEDIT command can be used:

```
SET      BDY(TZONE1)         /* Set to zone to edit        */.
ZONEEDIT UTILITY                /* Edit UTILITY entries       */.
CHANGE   PRINT('',MYOUTPUT)  /* Add PRINT value of MYOUTPUT*/.
ENDZONEEDIT.                 /* End of ZONEEDIT            */.
```

Processing

The ZONEEDIT command changes or adds the values for a subentry in different DDDEF or UTILITY entries in the same zone. It also changes the zone name in all the cross-zone subentries in a specified target zone. Before making any changes, SMP/E checks that the entry type specified is DDDEF, UTILITY, or XZENTRIES (for cross-zone subentries). It then opens the specified zone for update.

Next, SMP/E checks that the subentry name is valid for the entry and that the *to-value* is valid.

If all of these checks are successful, SMP/E performs the change as follows:

- When changing a subentry value:

  - For any conditional changes, SMP/E tests the IF condition and verifies that the *from-value* exists. If these checks succeed, SMP/E replaces the *from-value* with the *to-value*.
  - For unconditional changes, SMP/E verifies that the *from-value* exists and, if so, replaces the *from-value* with the *to-value.*
- When adding a subentry value, SMP/E searches the zone for the entries of the specified type that do not have a value for the specified subentry. If any are found,
  - For any conditional changes, SMP/E tests the IF condition. If this check succeeds, SMP/E adds the *to_value*.
  - For unconditional changes, SMP/E adds the *to_value*.

When adding subentries, with either a conditional or unconditional statement, DDDEF entries have an implied IF condition in that the UNIT, VOLUME, and WAITFORDSN values will only be added to DDDEF entries that have a DATASET subentry.

SMP/E then writes the ZONEEDIT Summary report to the SMPRPT data set and closes the zone.

If any of the checks fail, or if the ENDZONEEDIT command is missing, an error message is issued, and ZONEEDIT processing fails.

Zone and data set sharing considerations

The following identifies the phases of ZONEEDIT processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, "Sharing SMP/E Data Sets".

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: The global zone, the target zone, or the distribution zone is accessed, according to the zone specified in the previous SET command.
2. ZONEEDIT processing

   Global zone
   Update with exclusive enqueue.

   Target zone
   Update with exclusive enqueue.

   DLIB zone
   Update with exclusive enqueue.
3. Termination

   All resources are freed.

Chapter 30. The ZONEEXPORT command

The ZONEEXPORT command copies a specified zone from a VSAM data set to a sequential data set. There are two ways you can use this copy of the zone:

- As a **backup** copy: You can use this copy to re-create a zone that you or a program erased or otherwise destroyed.
- As a **transportable** copy: You can use this copy to re-create the same zone on another system or in another CSI on the same system.

When you have a backup copy of a zone, you also need backup copies of the SMP/E data sets corresponding to that zone. For SMP/E to restore the zone correctly, these data sets must be synchronized. For example, SMP/E needs these data sets to restore a zone either on another system or as a zone in another CSI on the same system.

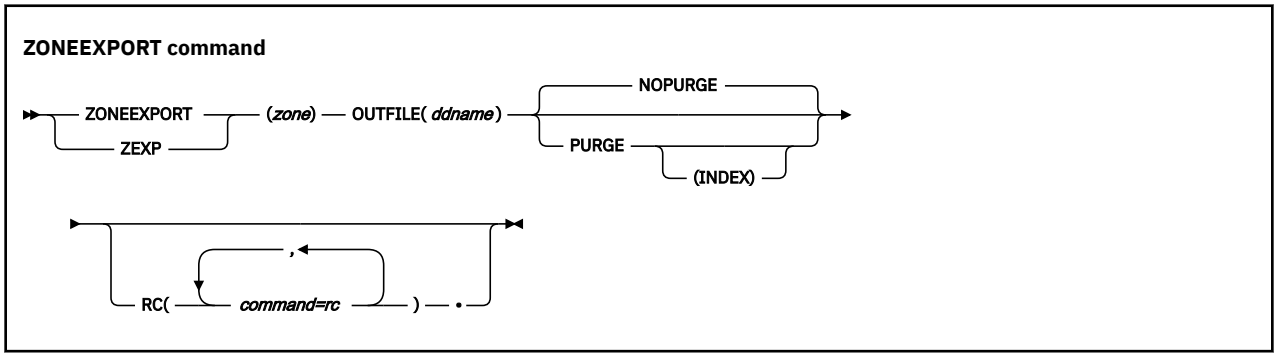
The ZONEIMPORT command processes the ZONEEXPORT output to restore the zone. ZONEIMPORT processing is similar to access method services REPRO processing of the same data.

Note: Although the access method services REPRO command accepts ZONEEXPORT output, it does not process it correctly. Data in another zone may be replaced, or the new zone may not be accessible to SMP/E. You must, therefore, use the ZONEIMPORT command to process ZONEEXPORT output.

Zones for SET BOUNDARY

For the ZONEEXPORT command, the SET BOUNDARY command must specify the name of the zone to be exported. This name must match the name of the input zone on the ZONEEXPORT command.

Syntax



Operands

zone
specifies the name of the input zone to be copied. This name must match the one specified on the SET BOUNDARY command.

OUTFILE
specifies the name of the DD statement that defines the output data set, which is the input for the ZONEIMPORT command.

Note:

1. OUTFILE can also be specified as OFILE.
2. Data set attributes for OUTFILE are described in the "SMP/E Data Sets" chapter in *z/OS SMP/E Reference*.

NOPURGE
specifies that the input zone is **not** to be deleted from the CSI when the ZONEEXPORT is done. This is the default.

Note: NOPURGE is mutually exclusive with PURGE.

PURGE
specifies that the input zone is to be deleted from the CSI data set when the ZONEEXPORT is done.

Note:

1. PURGE is not allowed for the global zone.
2. PURGE is mutually exclusive with NOPURGE.

INDEX
specifies that the index entry for the input zone is to be deleted from the global zone. If INDEX is not specified on the PURGE operand, only the data in the zone is deleted.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEEXPORT command.

Before SMP/E processes the ZONEEXPORT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEEXPORT command. Otherwise, the ZONEEXPORT command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEEXPORT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

Data sets used

The following data sets might be needed to run the ZONEEXPORT command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

outfile	SMPLOG	SMPOUT	SMPSNAP
SMPCNTL	SMPLOGA	SMPRPT	*zone*
SMPCSI			

Note:

1. *zone* represents the DD statements required for each distribution zone or target zone referred to in this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
2. *outfile* represents the DD statement required for the output data set. It is pointed to by the OUTFILE operand.

Usage notes

If the exported zone contained any cross-zone subentries, SMP/E issues warning messages. If you import that zone into a new zone, you need to determine what corrective action is needed to ensure that the new zone and any identified cross-zones are properly related through cross-zone subentries. This corrective action is similar to what you would do after using ZONECOPY. For more information, see "Updating cross-zone subentries" on page 408.

Output

The File Allocation report is produced during ZONEEXPORT processing. It is described in Chapter 34, "SMP/E reports," on page 457.

Example: Backing up target and distribution zones

In this example, two zones are exported. The first, a target zone named MVSTZ, is backed up and deleted. The second, a distribution zone named MVSDZ, is backed up but is not deleted.

```
SET        BDY (MVSTZ)      /* Set to zone to export.   */.
ZONEEXPORT (MVSTZ)          /* Export MVSTZ.            */
           OUTFILE(EXPORT1) /* DD statement for output. */
           PURGE            /* Delete MVSTZ.            */.
SET        BDY (MVSDZ)      /* Set to zone to export.   */.
ZONEEXPORT (MVSDZ)          /* Export MVSDZ.            */
           OFILE(EXPORT2)   /* DD statement for output. */
           NOPURGE          /* Do not delete MVSDZ.     */.
```

Processing

The ZONEEXPORT command makes a backup or transportable copy of a specified distribution, target, or global zone.

Before doing the export, SMP/E checks that the correct parameters were entered and that the export request is valid. If any of the checks fail, SMP/E issues an error message, and the ZONEEXPORT command fails. SMP/E checks that:

- The zone name on the SET BOUNDARY command matches the input zone name.
- If the zone to be exported is the global zone, the PURGE operand is not specified.

If all the validity checking is successful, the zone can be exported. SMP/E opens the input zone and reads the first record. If the input zone is not the global zone, this first record is the zone definition record. (For the global zone, SMP/E must generate a zone definition record.)

SMP/E then opens the output data set and writes the zone definition record to the output data set. Next, SMP/E reads the data from the input zone and writes it to the output data set using sequential reads and writes. After reading all the records from the input zone, SMP/E writes a record containing hex FFFF to the output data set. This record defines the end of data for a successful zone export.

SMP/E then closes the output data set and writes a message to indicate that the data has been written out successfully. If PURGE was specified, the records in the zone are deleted by ZONEDELETE. If INDEX was specified on the PURGE operand, and the global zone is available, the associated zone index is also deleted. Otherwise, the zone index is not changed. Finally, SMP/E closes the input zone.

If the exported zone contained cross-zone subentries, SMP/E issues warning messages. If SMP/E cannot open the input data set or the receiving zone, it issues an error message, and the ZONEEXPORT command fails.

Zone and data set sharing considerations

The following identifies the phases of ZONEEXPORT processing and the zones and data sets SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, "Sharing SMP/E data sets," on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

DLIB zone
Read without enqueue.

Note: The global zone and the zone specified in the SET BOUNDARY command are opened for read access only.

2. Zone export processing

Global zone
Read with shared enqueue.

If `PURGE(INDEX)` is specified, this is opened for update with exclusive enqueue.

Target zone
Read with shared enqueue.

If PURGE is specified, this is opened for update with exclusive enqueue.

DLIB zone
Read with shared enqueue.

If PURGE is specified, this is opened for update with exclusive enqueue.

Note: The zone specified in the previous SET command is accessed.

3. Termination

All resources are freed.

Chapter 31. The ZONEIMPORT command

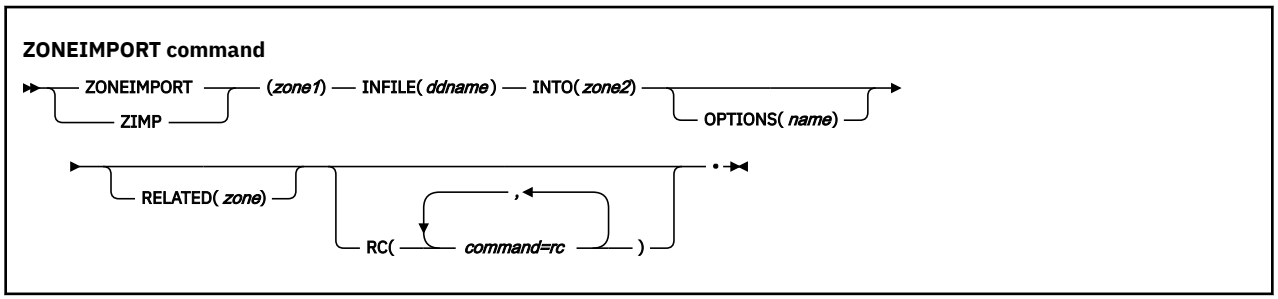
The ZONEIMPORT command loads an exported zone from a sequential data set into a specified distribution, target, or global zone. If you are importing a target or distribution zone, you can also change the zone name. (You cannot change the name of a global zone that is being imported.) Generally, you can import a zone only into the same type of zone. However, you can import a distribution zone into a target zone. You might want to do this after system generation, when the target zone is being initialized.

The input data for the ZONEIMPORT command must be the output from a ZONEEXPORT command. ZONEIMPORT processing of this data is similar to Access Method Services REPRO processing of this same data.

Zones for SET BOUNDARY

For the ZONEIMPORT command, the SET BOUNDARY command must specify where to load the zone. This name must match the name of the receiving zone specified on the INTO operand of the ZONEIMPORT command.

Syntax



Operands

zone1
specifies the name of the input zone to be imported. This is the name of the zone that was exported.

Note:

1. If you are importing a global zone, you must specify GLOBAL.
2. If you are importing a target or distribution zone, you can keep the same zone name or rename the zone:
   - If you are keeping the same zone name, you must specify the same zone here and on the SET BOUNDARY command.
   - If you are renaming the zone, this zone name is different from the one specified on the SET BOUNDARY command.

INFILE
specifies the name of the DD statement that defines the input data set created by the ZONEEXPORT command. You can import only a data set that was created by the ZONEEXPORT command.

Note: INFILE can also be specified as IFILE.

INTO
specifies the name of the receiving zone. This must match the name specified on the SET BOUNDARY command. If the INTO operand specifies a global zone, or you are importing a target or distribution zone to a new CSI data set, you should make sure you have allocated the CSI data set before you run

the ZONEIMPORT command, and initialize it with a GIMZPOOL record. **Do not** add any other entries or subentries to the zone.

Note: If the INTO operand specifies a target or distribution zone, the global zone must already contain a ZONEINDEX subentry specifying the name of that zone and the name of the CSI data set containing that zone.

OPTIONS
specifies the name of the OPTIONS entry to use for processing the receiving zone. The OPTIONS subentry in the zone definition of the receiving zone is set to the specified name.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEIMPORT command.

Before SMP/E processes the ZONEIMPORT command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEIMPORT command. Otherwise, the ZONEIMPORT command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEIMPORT command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RELATED
specifies the name of the target or distribution zone to which the receiving zone is related. The RELATED subentry in the zone definition of the receiving zone is set to the specified name.

Data sets used

The following data sets might be needed to run the ZONEIMPORT command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

infile	SMPLOG	SMPOUT	SMPSNAP
SMPCNTL	SMPLOGA	SMPRPT	*zone*
SMPCSI			

Note:

1. *zone* represents the DD statements required for each distribution zone or target zone referred to in this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
2. *infile* represents the DD statement required for the input data set. It is pointed to by the INFILE operand.

Usage notes

If the imported zone contains any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the imported zone and any identified cross-zones are properly related through cross-zone subentries. This corrective action is similar to what you would do after using ZONECOPY. For more information, see "Updating cross-zone subentries" on page 408.

Output

The File Allocation report is produced during ZONEIMPORT processing. It is described in Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the ZONEIMPORT command.

Example 1: Importing a distribution zone into a target zone

Assume that you have exported a distribution zone named IMPDLB into file IMPFILE, and now you want to use this zone to create a target zone named IMPTGT in an existing data set, SMP.IMPORT.CSI. You need an IMPFILE DD statement to point to the exported file, and an SMPCSI DD statement to point to the CSI data set that contains the global zone. The following SMP/E commands create the new zone:

```
SET        BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                       /* Define zone index.       */.
ADD        GZONE            /*                          */
           ZINDEX(          /*                          */
                  (IMPTGT,SMP.IMPORT.CSI,TARGET)    /*  */
                 )          /*                          */
                            /*                          */.
ENDUCL                      /*                          */.
SET        BDY(IMPTGT)      /* Set to new zone.         */.
ZIMP       (IMPDLB)         /* Import DLIB zone IMPDLB  */
           INFILE(IMPFILE)  /* into target zone IMPTGT. */
           INTO(IMPTGT)     /*                          */
           RELATED(IMPDLB1) /* Add related zone.        */.
```

After the ZONEIMPORT operation, a new global zone ZONEINDEX entry has been created to indicate that zone IMPTGT is in SMP.IMPORT.CSI and has a related zone, IMPDLB1. Note that you changed both the name and type of the zone when you imported it.

Example 2: Importing a global zone

Assume that you need to copy your global zone into another system for testing. You must first use ZONEEXPORT to unload the global zone data to a file (see Chapter 30, “The ZONEEXPORT command,” on page 427 for more information). Then, you must allocate a new CSI data set and initialize it with only a ZPOOL record. There must not be any other records in this new data set.

Assume that you exported the global zone to a file, whose ddname is GOUTFILE, and you allocated a new CSI data set, called SMP.IMPORT.GLOBAL.CSI, for the global zone. You need a GOUTFILE DD statement to point to the exported file and an SMPCSI DD statement to point to the new CSI data set, where you are importing the global zone. The following SMP/E commands import the global zone:

```
SET        BDY(GLOBAL)      /* Set to global zone.      */.
ZONEIMPORT (GLOBAL)         /* Import the global zone.  */
           INFILE(GOUTFILE) /*                          */
           INTO(GLOBAL)     /*                          */.
```

After the ZONEIMPORT operation, you have a new global zone in SMP.IMPORT.GLOBAL.CSI. The exported data remains in file GOUTFILE.

Example 3: Moving a zone to another CSI data set

Assume that you need to move a zone from one CSI data set to another. You can do this with the ZONEEXPORT (described in Chapter 30, “The ZONEEXPORT command,” on page 427) and ZONEIMPORT commands. First, you export the zone to a sequential data set. In the process of exporting the zone, you also delete the zone and its zone index by specifying PURGE(INDEX). Next, you define a new zone index pointing to the new CSI data set and then import the zone.

The following SMP/E commands move a zone from one CSI data set to another CSI data set:

```
SET        BDY (MVSTZ)      /* Set to zone to export.   */.
ZONEEXPORT (MVSTZ)          /* Export zone MVSTZ.       */
           OUTFILE(EXPORT1) /* DD statement for output. */
           PURGE(INDEX)     /* Delete zone MVSTZ        */
                            /* and its zone index.      */.

SET        BDY(GLOBAL)      /* Set to global zone.      */.
UCLIN                       /* Define new zone index.   */.
ADD        GZONE            /*                          */
           ZINDEX(          /*                          */
                  (MVSTZ,SMP.IMPORT.CSI,TARGET)     /*  */
                )           /*                          */
                            /*                          */.
ENDUCL                      /*                          */.
SET        BDY(MVSTZ)       /* Set to new zone.         */.
ZONEIMPORT (MVSTZ)          /* Import zone MVSTZ        */
           INFILE(EXPORT1)  /* from specified DD        */
           INTO(MVSTZ)      /* into target zone MVSTZ.  */.
```

After the ZONEEXPORT and ZONEIMPORT operations, you have moved zone MVSTZ into SMP.IMPORT.CSI.

Processing

The ZONEIMPORT command loads an exported zone into a specified distribution, target, or global zone.

Before importing the zone, SMP/E checks that the correct parameters were entered and that the import request is valid. If any of the checks fail, SMP/E issues an error message, and the ZONEIMPORT command fails. SMP/E checks that:

- The zone name on the SET BOUNDARY command matches the receiving zone name.
- Either the zone types of the exported and receiving zones match, or a distribution zone is being loaded into a target zone.
- The receiving zone does not already exist in the CSI data set.
- If the receiving zone is a global zone, no other zone is defined in the CSI.
- No cross-zone subentry from the input zone refers to a zone with the same name as the receiving zone.

If all the validity checking is successful, the zone can be imported. SMP/E opens the input data set, reads the zone definition record, and checks its validity. SMP/E opens the receiving zone for update processing. If a target or distribution zone is being imported and its zone name does not match the name of the receiving zone, SMP/E renames the zone being imported.

SMP/E checks for these situations:

- If the imported zone contains cross-zone subentries, SMP/E issues warning messages.
- If SMP/E cannot open the input data set or the receiving zone, it issues an error message, and the ZONEIMPORT command fails.
- If the zone name or type was changed, SMP/E updates the zone definition record. If the receiving zone is not the global zone, SMP/E writes this record to the receiving zone. SMP/E then reads the rest of the records from the input data set and writes them to the receiving zone using sequential reads and writes. When it reaches the record containing hex FFFF, SMP/E has finished importing the zone. It does not write this record to the receiving zone.
- If SMP/E reaches the end of the file before reading the hex FFFF record, it issues an error message and the ZONEIMPORT command fails.
- If you specified the OPTIONS or RELATED operand, SMP/E either writes the new entries to the receiving zone, if none already exist, or replaces the existing entries with the new ones and writes the new entries to the receiving zone.

SMP/E then closes the input data set and the receiving zone.

Zone and data set sharing considerations

The following identifies the phase of ZONEIMPORT processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: The zone specified in the SET command is accessed.

2. Zone import processing

   Global zone
   Update with exclusive enqueue.

   Target zone
   Update with exclusive enqueue.

   DLIB zone
   Update with exclusive enqueue.

   Note: The zone specified in the SET command is accessed.

3. Termination

   All resources are freed.

Chapter 32. The ZONEMERGE command

The ZONEMERGE command can be used to:

- **Copy** one zone to another (that is, merge an existing zone into a null zone). For example, you can use ZONEMERGE to:
  - Copy a distribution zone to a new target zone after a full system generation.
  - Prime a new distribution zone or target zone with data from an existing distribution zone or target zone before you build a new system.
- **Merge** two existing zones together.

  This function of ZONEMERGE should be used with caution. It should be used only to merge two zones that have absolutely no intersections (such as no common modules, macros, source, load modules, or SYSMODs). For further cautions on using ZONEMERGE to merge zones, see "Usage notes" on page 439.

Note:

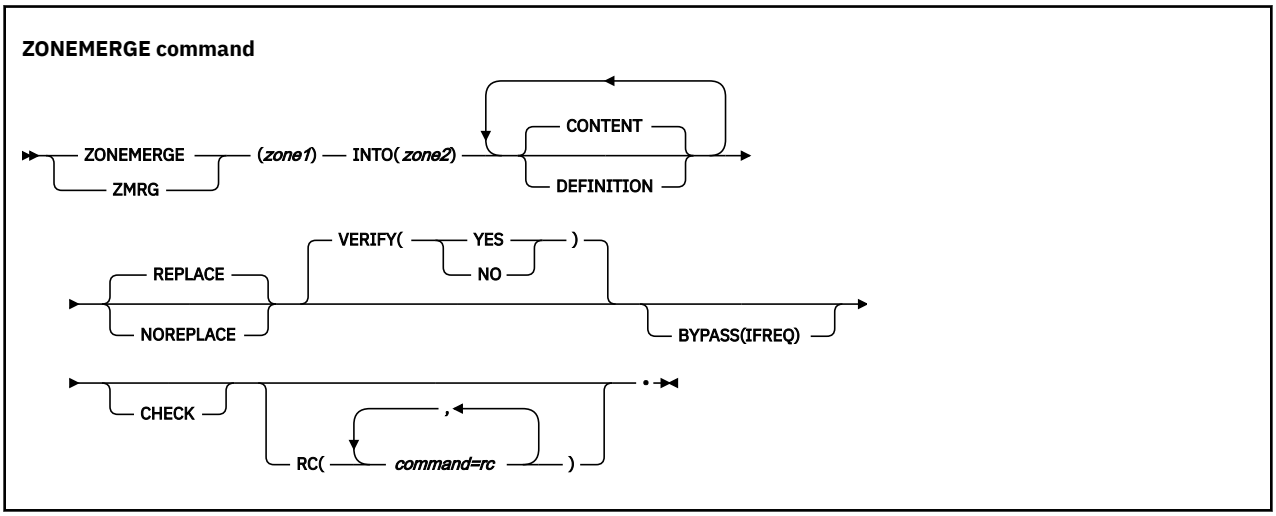
1. The zones processed by the ZONEMERGE command can either be in the same CSI data set or in different ones.
2. The ZONEMERGE command is the only method you should use to merge zones within the same CSI data set.
3. **Do not use the access method services REPRO command to merge CSI data sets.**

   Each CSI data set contains special records that record the names of the zones within that CSI, and an encoded value for that name. That encoded value is stored in each of the records in the CSI associated with that zone. Multiple CSI data sets can have the same encoded value for different zones. When ZONEMERGE is used to merge zones of a CSI, SMP/E can resolve the encoded names and assign new values to the data in each record. When access method services REPRO is used, however, the data is merged together; the result is an unusable CSI data set.
4. ZONEMERGE merges data only in CSI data sets. It does not affect the target or distribution libraries. After using the ZONEMERGE command, therefore, you must ensure that the target library or the distribution library is coordinated with the resulting target or distribution zone.

Zones for SET BOUNDARY

For the ZONEMERGE command, the SET BOUNDARY command must specify the name of the target zone or distribution zone into which the data is to be merged. This name must match the name of the zone specified on the INTO operand of the ZONEMERGE command.

Syntax



Operands

zone1
specifies the name of the zone from which the data to be merged is to be obtained (the originating zone).

BYPASS(IFREQ)
indicates that if SMP/E is performing verification processing for ZONEMERGE (VERIFY(YES) is specified or defaulted), then SMP/E should ignore any conditional requisite SYSMODs that are missing. BYPASS(IFREQ) is not used by default.

CHECK
indicates SMP/E should not update the destination zone. Instead it should test for merge conflicts and produce a ZONEMERGE Report indicating what would have happened if CHECK were not specified. CHECK is not used by default.

CONTENT
specifies that the content entries in the zone should be copied. Content entries include SYSMOD entries, element entries, and LMOD entries. If neither CONTENT nor DEFINITION is specified, CONTENT is the default.

Note: CONTENT can also be specified as CON.

DEFINITION
specifies that the DDDEF entries in the zone should be copied.

Note: DEFINITION can also be specified as DEF.

INTO
specifies the zone into which the data from the originating zone is merged (the destination zone). This operand is required.

A ZONEINDEX subentry for the destination zone must be specified in the global zone. Also, if you are merging into a new CSI data set, you should make sure you have allocated the CSI data set before you run the ZONEMERGE command, and initialize it with a GIMZPOOL record.

NOREPLACE
specifies that if the destination zone contains the same entries as the originating zone, the originating zone entries **are not** to overlay the destination zone entries.

Note: NOREPLACE is mutually exclusive with REPLACE.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONEMERGE command.

Before SMP/E processes the ZONEMERGE command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONEMERGE command. Otherwise, the ZONEMERGE command fails. For more information about the RC operand, see Appendix A, "Processing the SMP/E RC operand," on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONEMERGE command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

REPLACE
specifies that if the destination zone contains the same entries as the originating zone, the originating zone entries **are** to overlay the destination zone entries. This is the default.

Note: REPLACE is mutually exclusive with NOREPLACE.

VERIFY
indicates whether verification processing for ZONEMERGE should be performed. If `VERIFY(YES)` is specified, then SMP/E will verify the `CONTENT` type entries in the originating zone (SYSMODs, elements, and LMODs) can be merged into the destination zone without conflicts and without creating incompatible SYSMOD relationships or missing SYSMOD requisites. If `VERIFY(NO)` is specified, then SMP/E will not verify the `CONTENT` entries before merging. `VERIFY(YES)` is the default.

Data sets used

The following data sets might be needed to run the ZONEMERGE command. They may be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference* .

SMPCNTL	SMPLOG	SMPOUT	SMPSNAP
SMPCSI	SMPLOGA	SMPRPT	*zone*

Note: *zone* represents the DD statements required for each distribution zone or target zone used by this command. If the DD statements are not specified, the data sets are dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. A DD statement is required for both the originating zone and the destination zone. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

- Use the ZONEMERGE command with caution. You can get unexpected results if the zones being merged have entries with the same name, or if you mistakenly specified REPLACE instead of NOREPLACE (or vice versa). For example, suppose both ZONE1 and ZONE2 contain an entry for module IFBMOD01. In ZONE1, the RMID is UZ12345, and in ZONE2, the RMID is UZ12346. Assume UZ12346 is at a higher service level than UZ12345. If ZONE2 were merged into ZONE1 with the REPLACE option, but the target library still contained the version of the module from UZ12345, the resulting entry for module IFBMOD01 in ZONE1 would reflect the incorrect service level of the module.
- If the originating zone contains any cross-zone subentries, SMP/E issues warning messages. You need to determine what corrective action is needed to ensure that the cross-zone information for the merged entries and the identified cross-zones is correct, and that all zones are properly connected. This corrective action is similar to what you would do after using ZONECOPY. For more information, see "Updating cross-zone subentries" on page 408.
- During CONTENT processing, the SREL subentry found in the ZONE definition entry of the original zone will be merged into the ZONE definition entry of the destination zone

Output

Two reports are produced during ZONEMERGE processing:

- File Allocation report
- ZONEMERGE report

These reports are described in Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the ZONEMERGE command.

Note: For most cases where ZONEMERGE can be used to copy zones, you can also use the access method services REPRO command and ZONERENAME to achieve the same result. For examples of ZONERENAME, see “Examples” on page 452.

Example 1: Creating new target zone after system generation

After performing a full system generation, you need to create a new target zone describing the new set of target libraries created during the system generation process. The following example illustrates the steps necessary to do this.

The first step is to define the new target zone. Assume the previous target zone was named TGT1, the new target zone is to be named TGT2 (generated from distribution zone DLB1), and the OPTIONS entry to be used is MVSOPT. Both target zones are to exist in SMPE.SMPCSI.CSI. The following SMP/E commands define the new target zone:

```
SET      BDY(GLOBAL)         /* Set to global.            */.
UCLIN                        /* UCLIN to define new zone.*/.
ADD      GZONE               /*                           */
         ZONEINDEX(          /* Define zone in CSI.       */
                   (TGT2,SMPE.SMPCSI.CSI,TARGET)    /*  */
                  )          /*                           */
                             /* Define new zone.          */.
ENDUCL                       /*                           */.
SET      BDY(TGT2)           /* Set to new zone.          */.
UCLIN                        /* Add definition entry.     */.
ADD      TZONE(TGT2)         /* Define zone.              */
         RELATED(DLB1)       /* Same info as              */
         OPTIONS(MVSOPT)     /* in old zone.              */
         SREL(Z038)          /*                           */.
ENDUCL                       /*                           */.
```

Now that the new target zone has been defined, you should use the ZONEMERGE command to copy the entries from the old target zone that do not reflect system structure information. This can be done by using the DEFINITION operand of the ZONEMERGE command. The only entries that are copied in this case are the DDDEF entries, and these should not have changed during the system generation process. The following SMP/E commands copy the DDDEF entries:

```
SET      BDY(TGT2)           /* Set to new zone.          */.
ZONEMERGE(TGT1)              /* Merge from TGT1           */
         INTO(TGT2)          /* into new zone TGT2.       */
         DEFINITION          /* Definition only.          */.
```

Note: If you have changed the unit or VOLSER of the target volumes, and the DDDEF entries describing those libraries also contains the unit or volume information (that is, the DDDEF entries did not assume that the data sets were cataloged), these entries must be modified with UCLIN to reflect the new data. The following is an example of how to modify the DDDEF entries to change both the unit and volume information (assume old unit and volume were 3330 and TGTVOL and new information is 3380 and TGTPCK):

```
SET      BDY(TGT2)           /* Set to new target zone.  */.
UCLIN                        /*                          */.
REP      DDDEF(MACLIB)       /* MACLIB changed to        */
         UNIT(3380)          /* unit 3380,               */
```

```
         VOLUME(TGTPCK)     /* volume TGTPCK.           */
                            /* dsname not changed.      */.
ENDUCL                      /*                          */.
```

At this point, you must copy the distribution zone to the new target zone so SMP/E can determine the function and service levels of all of the distribution library elements. This copy can be done as follows:

```
SET      BDY(TGT2)          /* Set to new target zone.  */.
ZONEMERGE(DLB1)             /* Copy from DLIB           */
         INTO(TGT2)         /* into new target system,  */
         CONTENT            /* but only the element     */
                            /* and SYSMOD entries.      */.
```

Now that the target zone is defined and primed with the nonstructure entries from the previous target zone, you now have to prime the new target zone with the structure information about the entries in the new target libraries. You can do this with the JCLIN command of SMP/E, using the stage 1 output as input. Assuming the stage 1 output was saved in data set STG1.TGT2.CNTL and an SMP/E procedure similar to SMPPROC, as described in the "Sample SMP/E Cataloged Procedure" appendix in *z/OS SMP/E Reference*, is available, the following job primes the new target zone:

```
//JOB1     JOB 'accounting info',MSGLEVEL=(1,1)
//STEP1    EXEC SMPPROC
//SMP.SMPJCLIN DD DSN=STG1.TGT2.CNTL,DISP=SHR
//SYSIN    DD *
SET      BDY(TGT2)          /* Set to new target.       */.
JCLIN                       /* Update zone.             */.
LIST                        /* List zone.               */.
/*
```

The new target zone, TGT2, is now ready to be used for the installation of service or new function. When this system has been tested and the old level, TGT1, is no longer required, you can delete it by use of the ZONEDELETE command, as follows:

```
SET      BDY(TGT1)          /* Set to old target.       */.
ZONEDELETE                  /*                          */
         TZONE(TGT1)        /* Delete it.               */.
```

Example 2: Creating a test target system

Assume that you have a target zone named TSTTGT1, and that you want to create a backup copy of that zone, to be named BKPTGT1, before installing a new product. For this example, assume the libraries themselves have already been backed up. Also assume the CSI describing the TSTTGT1 target zone is SMPE.SMPCSI.CSI, and the CSI that will contain the backup zone BKPTGT1 is SYS1.BACKUP.SMPCSI.CSI. The following set of commands creates the backup zone:

```
SET      BDY(GLOBAL)        /* Set to global.           */.
UCLIN                       /* UCLIN to define new zone.*/.
ADD      GZONE              /*                          */
         ZONEINDEX(         /* Define zone in CSI.      */
                   (BKPTGT1,SYS1.BACKUP.SMPCSI.CSI,TARGET)
                  )         /*                          */
                            /* Define new zone.         */.
ENDUCL                      /*                          */.
SET      BDY(BKPTGT1)       /* Set to new zone.         */.
UCLIN                       /* Add definition entry.    */.
ADD      TZONE(BKPTGT1)     /* Define zone.             */
         RELATED(TSTDLB1)   /* Same info as             */
         OPTIONS(MVSOPT1)   /* in old zone.             */
         SREL(Z038)         /*                          */.
ENDUCL                      /*                          */.
ZONEMERGE(TSTTGT1)          /* Merge(copy) TSTTGT1      */
         INTO(BKPTGT1)      /* into new zone,           */
         CONTENT            /* all type entries.        */
         DEFINITION         /*                          */
                            /* Replace/noreplace is not
                               necessary -- destination
                               zone is empty.           */.
LIST                        /* List the new zone.       */.
```

Example 3: Creating a test distribution system

Assume that you have a distribution zone named TSTDLB1, and you want to make a copy of it, to be named TSTDLB2, for use in some sort of testing. You first have to make copies of the libraries associated with that distribution zone. After doing that, you can use SMP/E to make a copy of the distribution zone. You can use the following set of commands to do this:

```
SET      BDY(GLOBAL)        /* Set to global.           */.
UCLIN                       /* UCLIN to define new zone.*/.
ADD      GZONE              /*                          */
         ZONEINDEX(         /* Define zone in CSI.      */
                   (TSTDLB2,SMPE.SMPCSI.CSI,DLIB) /*    */
                  )         /*                          */
                            /* Define new zone.         */.
ENDUCL                      /*                          */.
SET      BDY(TSTDLB2)       /* Set to new zone.         */.
UCLIN                       /* Add definition entry.    */.
ADD      DZONE(TSTDLB2)     /* Define zone.             */
         RELATED(TSTTGT1)   /* Same info as             */
         OPTIONS(MVSOPT1)   /* in old zone.             */
         SREL(Z038)         /*                          */.
ENDUCL                      /*                          */.
ZONEMERGE(TSTDLB1)          /* Merge(copy) TSTDLB1      */
         INTO(TSTDLB2)      /* into new zone,           */
         CONTENT            /* all type entries.        */
         DEFINITION         /*                          */
                            /* Replace/noreplace is not
                               necessary -- destination
                               zone is empty.           */.
LIST                        /* List the new zone.       */.
```

Processing

With the ZONEMERGE command, you can merge a specified distribution zone or target zone into another specified target zone or distribution zone. REPLACE is the default if neither REPLACE nor NOREPLACE is specified. After the merge operation is complete, the originating distribution zone or target zone still exists in the CSI data set.

SMP/E supports the following variations of merging:

- Distribution zone to distribution zone
- Target zone to target zone
- Distribution zone to target zone

Note: Any attempt to merge a target zone to a distribution zone causes an error.

The syntax refers to two types of zone entries: DEFINITION and CONTENT. DEFINITION entries are set up by the user to define data sets used by SMP/E; CONTENT entries are created by SMP/E. If neither CONTENT nor DEFINITION is specified, the default is to merge only the CONTENT type entries. This supports the distribution zone to target zone copy after a system generation when the desire is to recopy the structure of the system, but not the definition. The following defines how the various entries are categorized:

- DEFINITION type entries
  - DDDEF entries
- CONTENT type entries
  - ASSEM entries
  - Element entries
  - DLIB entries
  - LMOD entries
  - SYSMOD entries

When the ZONEMERGE command is encountered, SMP/E first ensures that both zones have been previously defined. The ZONEMERGE command does not create a new zone. If either one has not been defined, an error message is issued, and the command is not processed. To fix this problem, check the zone names specified, and either change the incorrect name, or define the missing zone by using UCLIN.

SMP/E then checks to make sure the zone type (that is, target or distribution), specified in the global zone ZONEINDEX, matches the type specified in the ZONEMERGE command. If not, an error message is issued, and the ZONEMERGE is not done. To fix this problem, check to ensure that the zone names specified are correct and that the zone types specified in the global zone ZONEINDEX are correct; fix whatever discrepancy was found, and rerun the command.

To perform the actual merge, SMP/E sequentially reads through both zones. For each entry in the originating zone, SMP/E checks the entry type (DEFINITION, CONTENT, or both) and looks at the operands specified on the ZONEMERGE command to determine whether the entry should be processed.

- If VERIFY(YES) was specified or defaulted and CONTENT entries are being processed (CONTENT was specified or defaulted), SMP/E verifies that the zones can be merged without conflicts and without creating incompatible SYSMOD relationships or missing SYSMOD requisites. See “SYSMOD verification processing” on page 444. SMP/E also verifies that elements and LMODs can be merged without conflict. See “Element and LMOD verification processing” on page 445 . If no conflicts are detected during verification processing, the CONTENT entries are copied from the originating zone to the destination zone. If conflicts are detected during verification processing, the ZONEMERGE commands ends and no entries are merged.
- SMP/E does no verification of the DEFINITION entries. Also, if VERIFY(NO) is specified, SMP/E does no verification of the CONTENT entries. In this case, SMP/E checks to see whether an entry exists in the destination zone.
  - If an entry is not found, the entry from the originating zone is stored.
  - If an entry is found and REPLACE was specified, the entry from the destination zone is deleted, and the entry from the originating zone is stored.
  - If an entry is found and NOREPLACE was specified, processing continues with the next entry.

Notes:

- If the CHECK operand was specified, all processing and messaging is the same, except no entries are deleted or stored in the destination zone.
- Any conditional requisite subentries that exist in either the originating zone or the destination zone are preserved during the merge of a SYSMOD entry. See “Preserving CIFREQ subentries” on page 445.

If the originating zone contained cross-zone subentries, SMP/E issues warning messages. SMP/E determines what cross-zone information to merge from MOD and LMOD entries in the originating zone, as follows:

- If any cross-zone subentries refer to a zone with the same name as the receiving zone, that information is not copied to the receiving zone, and an error message is issued. SMP/E still attempts to merge the rest of the data in the zone.
- If the entry in the originating zone contains nothing but cross-zone subentries, SMP/E does not copy the cross-zone subentries to the receiving zone.
- If the entry in the receiving zone contains nothing but cross-zone subentries, SMP/E merges the entry from the originating zone into the receiving zone, regardless of whether the REPLACE operand was specified. Cross-zone subentries from the originating zone are added to those from the receiving zone. SMP/E issues messages indicating the source of each of the cross-zone subentries in the merged entry.
- Otherwise, the entries are merged only if REPLACE was specified. In this case, SMP/E merges the entry from the originating zone into the receiving zone, replacing all the information in the receiving zone (except the cross-zone subentries) with information from the originating zone. Cross-zone subentries from the originating zone are added to those from the receiving zone. SMP/E issues messages to indicate the source of each of the cross-zone subentries in the merged entry.
- If any cross-zone subentries were merged into the receiving zone, SMP/E adds TIEDTO subentries for the cross-zones to the receiving zone's TARGETZONE entry.

When you merge zones, you can end up with an LMOD entry with a MODDEL subentry for a module and a MOD entry for that same module. For example, suppose you have a zone with an LMOD entry that has a MODDEL subentry containing module MOD005, and you merge that zone with another zone that has a MOD entry for MOD005. It might seem contradictory, but the result is a zone having an LMOD entry with a MODDEL subentry containing MOD005 and a MOD entry for MOD005. If you install a SYSMOD containing MOD005, module MOD005 is removed from the MODDEL subentry.

SYSMOD verification processing

If SYSMODs are to be merged (the CONTENT operand was specified or defaulted), and VERIFY(YES) was specified or defaulted, the SYSMOD verification phase of the **ZONEMERGE** command ensures conditional requisite SYSMODs are not missing, and ensures negative prerequisites, installed, deleted and superseded SYSMODs are not present.

SYSMODs are analyzed in hierarchical order, and SMP/E will not continue analyzing SYSMODs whose parent FMID has already been determined to have a merge conflict.

SMP/E analyzes all SYSMOD entries in the originating zone, in groups using this order:

1. Base FUNCTIONs
2. Dependent FUNCTIONs
3. PTFs, APARs, and USERMODs

First, base FUNCTION SYSMODs in the originating zone are analyzed. After all base FUNCTION SYSMODs in the originating zone have been analyzed, then SMP/E analyzes the dependent FUNCTIONs in the originating zone, but only if no errors have been detected for the dependent FUNCTION SYSMOD's base. If an error was detected for a base FUNCTION, then no checking is performed for that base's dependent FUNCTIONs.

After all dependent FUNCTION SYSMODs in the originating zone have been analyzed, then SMP/E analyzes the PTFs, APARs, and USERMODs in the originating zone, but only if no errors have been detected for the FUNCTION SYSMOD that matches the FMID of those PTFs, APARs, and USERMODs. If an error was detected for a FUNCTION, then no checking is performed for that FUNCTION's PTFs, APARs, and USERMODs.

After all other SYSMODs have been analyzed, SYSMOD entries that have no FMIDs are analyzed. These are "dummy" SYSMOD entries, for SYSMODs that have been deleted, or superseded. For each SYSMOD in the originating zone that has been deleted or superseded and whose entry has no FMID, if a matching SYSMOD entry is found in the destination zone, regardless of that SYSMODs current state (installed, error, deleted, or superseded), a conflict is reported.

All SYSMODs that have not been deleted or superseded are analyzed as described in the following sections.

Ensure installed, deleted, and superseded SYSMODs are not present

The **ZONEMERGE** command ensures SYSMODs installed in the originating zone are not already installed in the destination zone. Specifically, for each SYSMOD being analyzed, SMP/E looks for a same named SYSMOD entry in the destination zone. If a matching SYSMOD entry is found in the destination zone (installed, deleted or superseded), a conflict is reported. If a matching SYSMOD was not found in the destination zone, then SMP/E analyzes all SYSMODs that it deletes or supersedes, if any. Specifically, SMP/E finds all DELETE and SUPING subentries from the SYSMOD entry in the originating zone. For each subentry value, SMP/E looks for SYSMOD entries in the destination zone with the same name as the DELETE and SUPING subentries. If any matching SYSMODs are found in the destination zone, regardless of that SYSMODs current state (installed, error, deleted, or superseded), a conflict is reported.

Ensure NPREs are not present

The **ZONEMERGE** command ensures SYSMODs are not merged from or into a zone if it has a negative prerequisite (NPRE) condition with another SYSMOD in the other zone. Specifically, for each SYSMOD being analyzed in the originating zone, regardless of the SYSMOD's current state (installed, error,

superseded, or deleted), SMP/E finds all the SYSMOD's NPRE subentries. It looks for SYSMOD entries with the same name as the NPRE subentry in the destination zone. If any matching SYSMODs are found in the destination zone, regardless of that SYSMOD's current state, a conflict is reported.

Likewise, SMP/E analyzes all of the SYSMOD entries in the destination zone, regardless of the SYSMOD's current state (installed, error, superseded, or deleted), to find all NPRE subentries. It then looks for SYSMOD entries with the same name as the NPRE subentry in the originating zone. If any matching SYSMODs are found in the originating zone, regardless of the SYSMOD's current state, a conflict is reported.

Ensure conditional requisites are not missing

The **ZONEMERGE** command ensures no conditional requisite SYSMODs will be missing in the destination zone. More specifically,

1. For each Function SYSMOD being analyzed in the originating zone that is not in error and has not been deleted or superseded, SMP/E searches for CIFREQ subentries in the destination zone with a matching FMID (that is, the FMID specified on the originating ++IF statement matches the Function SYSMOD name). For each matching CIFREQ subentry in the destination zone, SMP/E ensures the specified requisite is in either the originating zone or the destination zone. If a requisite SYSMOD is in the originating zone or the destination zone (and is not in error and not deleted), the conditional requisite SYSMOD will be satisfied after the merge operation completes.

   However, if the requisite is not in the originating zone or the destination zone, then if BYPASS(IFREQ) was not specified, a conflict is reported. If BYPASS(IFREQ) was specified, a warning is noted and merging may continue.
2. For each SYSMOD being analyzed in the originating zone (FUNCTIONS, PTFs, APARs, and USERMODs) that is not in error and has not been deleted or superseded, SMP/E finds CIFREQ subentries also in the originating zone with a matching causer (that is, the SYSMOD that contained the originating ++IF statement matches the name of the SYSMOD being analyzed). For each matching CIFREQ subentry, SMP/E determines if the Function SYSMOD for the FMID specified on the originating ++IF statement is in the destination zone. If the FMID is in the destination zone (is not in error and has not been deleted or superseded), SMP/E ensures the specified requisite is in either the destination zone or the originating zone. If the requisite is in the destination zone or the originating zone (and is not in error and not deleted), the conditional requisite will be satisfied after the merge operation completes.

   However, if the Function SYSMOD for the specified FMID is in the destination zone but the requisite is not in the destination zone or the originating zone, then if BYPASS(IFREQ) was not specified, a conflict is reported. If BYPASS(IFREQ) was specified, a warning is noted and merging may continue.

Element and LMOD verification processing

If no SYSMOD verification errors were detected, and if CONTENT type entries are being merged and if VERIFY(YES) was specified or defaulted, then the ZONEMERGE command will verify elements, and LMODs can be merged into the destination zone without conflicts.

For each element and LMOD in the originating zone, SMP/E will determine if an entry with the same name and type exists in the destination zone.

1. If an entry with the same name and type is not found, then the entry from the originating zone is merged into the destination zone.
2. If an entry with the same name and type is found, then the entry is not copied to the destination zone and a conflict is reported.

Preserving CIFREQ subentries

To support merging CIFREQ subentries with SYSMOD entries, the CIFREQ subentries are considered independent of the rest of the SYSMOD entry. That is, the presence of CIFREQ subentries does not affect whether a SYSMOD entry can be merged or replaced.

For each SYSMOD entry in the originating zone that contains more than just CIFREQ subentries, if the CONTENT operand was specified on the **ZONEMERGE** command, then the processing is as follows:

- If a SYSMOD entry with the same name is not found in the destination zone, the entry from the originating zone (including any CIFREQ subentries from the originating zone) is stored in the destination zone.
- If a SYSMOD entry with the same name is found in the destination zone, and the entry in the destination zone contains only CIFREQ subentries, then the entry from the originating zone (including any CIFREQ subentries from the originating zone) is stored in the destination zone. This preserves the existing CIFREQ subentries in the destination zone.
- If a SYSMOD entry with the same name is found in the destination zone, and the entry in the destination zone contains more than only CIFREQ subentries, then one of the following occur:
  - If REPLACE was specified, the existing entry (less the existing CIFREQ subentries) in the destination zone is deleted, and the entry from the originating zone (including any CIFREQ subentries from the originating zone) is stored in the destination zone. This preserves any CIFREQ subentries in the destination zone.
  - If NOREPLACE was specified, no changes are made to the entry in the destination zone, and processing continues with the next SYSMOD entry.

For each SYSMOD entry in the originating zone that contains only CIFREQ subentries, if CONTENT was specified, then the CIFREQ subentries from the originating zone are always stored into the destination zone, regardless of whether a same named SYSMOD entry exists in the destination zone or not, or whether an existing SYSMOD entry in the destination zone has CIFREQ subentries or not. This way, the SYSMOD entries in the destination zone will be preserved and merged with CIFREQ subentries from the originating zone.

Zone and data set sharing considerations

The following identifies the phases of ZONEMERGE processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note:

   a. Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

   b. Both the “from” and “to” zones are accessed at this time.

2. ZONEMERGE processing

   Target zone
   Read with shared enqueue.

   Target zone
   Update with exclusive enqueue.

   DLIB zone
   Read with shared enqueue.

   DLIB zone
   Update with exclusive enqueue.

   Note:

a. Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

b. The “from” zone is accessed for read with shared enqueue; the “to” zone is accessed for update with exclusive enqueue.

3. Termination

   All resources are freed.

Chapter 33. The ZONERENAME command

With the ZONERENAME command, you can assign a new name to an existing target zone or distribution zone in an SMPCSI data set. You can also:

- Change a distribution zone to a target zone.
- Change the default OPTIONS entry name.
- Change the name of the related zone.
- Have SMP/E add a ZONEINDEX subentry for the renamed zone, if it is in a different CSI from the one containing the "old" name of the zone.

Note: ZONERENAME does not rename the SMPCSI data set containing the zone. It only renames the SMP/E zone inside the SMPCSI data set.

You may want to use the ZONERENAME command in some of the following situations:

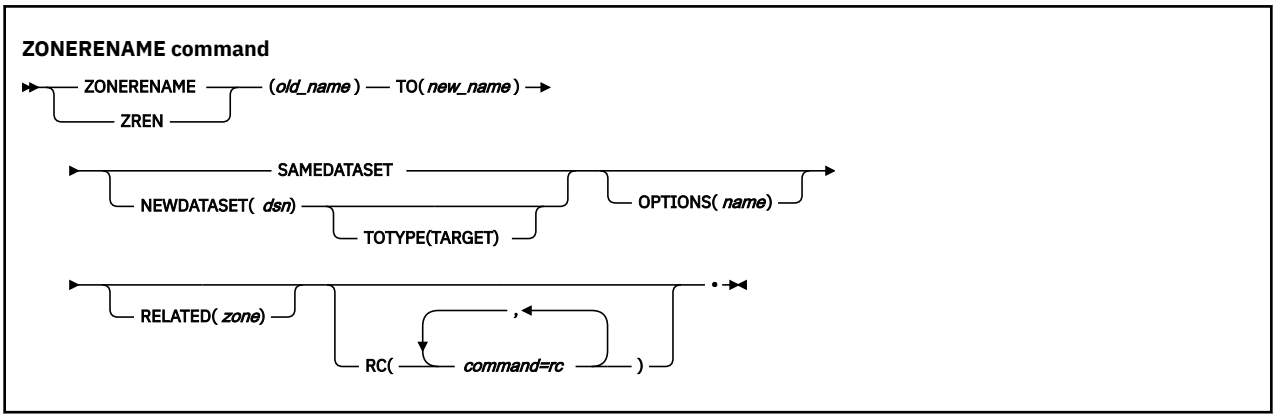
- If the current name of a zone does not conform to naming conventions of the establishment controlling the zone, and you want to assign a new name to the zone without going through the overhead required to do a ZONEMERGE followed by a ZONEDELETE.
- If you create a new set of target libraries during system generation and need a target zone to describe them, you have two choices. You can use the ZONEMERGE command to create the target zone (see “Examples” on page 440 for further information), or you can make a copy of the CSI containing the distribution zone. Then, using the ZONERENAME command, you can rename the distribution zone in the CSI copy to your new target zone name, change the zone type to TARGET, and connect it to the original distribution zone.
- If you have made a copy of a CSI data set and want to access the copied zones through an existing global zone. Assuming that the zones in the original CSI are already accessed through that global zone, you cannot also access the zones in the copy, because there can be only one ZONEINDEX entry in the global zone for each unique zone name.

  In this case, you want to rename the zone in the CSI copy, and then connect the renamed zone in your global zone. This procedure can be used to make a working copy of a zone under a different name for testing purposes without the overhead required to do a ZONEMERGE.

Zones for SET BOUNDARY

For the ZONERENAME command, the SET BOUNDARY command must specify the global zone.

Syntax



Operands

old-name
specifies the name of the zone to be renamed.

Note:

1. The specified name cannot be GLOBAL.
2. The old zone name cannot be the same as the new zone name.

TO
specifies the new name for the zone. This operand is required.

Note:

1. The new zone name cannot be GLOBAL.
2. The new zone name cannot be the same as the old zone name.

NEWDATASET
indicates that the zone to be renamed and the SMPCSI containing it are not yet defined by a zone index in the global zone. (For example, there might be a zone index for the old zone name, but not for the data set specified by NEWDATASET, or there might not be any zone index at all for the old zone name.) SMP/E will rename the zone in the indicated data set, and then create a zone index for the new zone name and the SMPCSI data set containing that zone.

You must specify either NEWDATASET or SAMEDATASET.

Note:

1. The indicated data set must be an existing SMPCSI containing the zone to be renamed. ZONERENAME does not create the SMPCSI data set for you.
2. The data set must not be the same as the one currently defined in the zone index for the old zone name.
3. The data set name must conform to the naming conventions for a CSI data set and therefore must have a low-level qualifier of `.CSI`. The data set name can contain no more than 44 characters.
4. NEWDATASET tells SMP/E to rename only the zone in the indicated data set. It does not rename the SMPCSI data set containing that zone. You cannot use the ZONERENAME command to rename SMPCSI data sets.

OPTIONS
specifies the name of the OPTIONS entry to use for processing the renamed zone. The OPTIONS subentry in the zone definition of the RENAMED zone is set to the specified name. If `OPTIONS` is not specified, the OPTIONS subentry in the definition of the renamed zone is not changed.

RC
changes the maximum return codes allowed for the specified commands. These return codes determine whether SMP/E can process the ZONERENAME command.

Before SMP/E processes the ZONERENAME command, it checks whether the return codes for the specified commands are less than or equal to the values specified on the RC operand. If so, SMP/E can process the ZONERENAME command. Otherwise, the ZONERENAME command fails. For more information about the RC operand, see Appendix A, “Processing the SMP/E RC operand,” on page 541.

Note:

1. The RC operand must be the **last** operand specified on the command.
2. If you do specify the RC operand, return codes for commands not specified do not affect processing for the ZONERENAME command. Therefore, if you use the RC operand, you must specify every command whose return code you want SMP/E to check.

RELATED
specifies the name of the target or distribution zone to which the renamed zone is related. The RELATED subentry in the zone definition of the renamed zone is set to the specified name. If RELATED is not specified, the RELATED subentry in the renamed zone's definition is not changed.

SAMEDATASET
indicates that the zone to be renamed is already defined by a zone index in the global zone, and is to be renamed in its current data set. The zone index is be changed to indicate the new zone name.

You must specify either SAMEDATASET or NEWDATASET.

Note: SAMEDATASET is mutually exclusive with TOTYPE(TARGET).

TOTYPE(TARGET)
indicates that the zone to be renamed is currently a distribution zone in a copy of an SMP/E CSI data set and is to be changed to a target zone. After the rename operation, the GLOBALZONE ZONEINDEX record indicates that this is now a target zone and the zone definition for the zone is updated to indicate that it is a target zone.

Note:

1. TOTYPE is mutually exclusive with SAMEDATASET.
2. TOTYPE is needed only if you are changing a target zone to a distribution zone. For example, suppose you did a system generation and copied the distribution zone to initialize the target zone. You could use the ZONERENAME command to rename the copied distribution zone and change the type to target.

Data sets used

The following data sets might be needed to run the ZONERENAME command. They can be defined by DD statements or, usually, by DDDEF entries. For more information about these data sets, see *z/OS SMP/E Reference*.

newzonename	SMPCSI	SMPLOGA	SMPRPT
oldzonename	SMPLOG	SMPOUT	SMPSNAP
SMPCNTL			

Note:

1. *newzonename* represents the DD statement required for the new zone name. If it is not specified, the data set is dynamically allocated according to the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.
2. *oldzonename* represents the DD statement required for the zone to be renamed by this command. If it is not specified, the data set is dynamically allocated using the ZONEINDEX information in the GLOBALZONE entry. Also note that, while DD statements may be used to override the ZONEINDEX information, they are not a substitute for a zoneindex. A zoneindex is always required for a zone.

Usage notes

- The *oldzonename* CSI data set need not be mounted when the NEWDATASET operand is used. All operations are performed against the global zone data set and the CSI data set specified as the value of the NEWDATASET operand.
- APPID and ACCID values are not updated in the global zone SYSMOD entries for the renamed zone.
- DDDEF entries in the renamed zone are not changed. To change information in these entries, you must use UCLIN, ZONEEDIT, or the administration dialogs.
- It is your responsibility to ensure that the proper data sets are used for the renamed zone. A partial list to consider is the SMPSCDS, SMPMTS, SMPLOG, SMPSTS, and any target or distribution library data sets.

- If a zone in a copy of a CSI data set is to be renamed, you should be aware that data for other zones in the copied data set remains in the data set, wasting space. Therefore, if you plan to make a copy of a CSI data set and rename the zone therein, you should either use the ZONEDELETE command to delete all the other zones or, when setting up the CSI data sets, ensure that each CSI data set contains only one zone.
- If the zone being renamed contains any TIEDTO subentries for cross-zones, SMP/E issues messages with information to help you update the cross-zones with the new zone name. For information that can help you determine the action to take, see “Updating cross-zone subentries” on page 408.

Output

The File Allocation report is produced during ZONERENAME processing. It is described in Chapter 34, “SMP/E reports,” on page 457.

Examples

The following examples are provided to help you use the ZONERENAME command.

Example 1: Renaming an existing zone

The simplest use of ZONERENAME is to just assign a new name to an existing zone. For example, assume that responsibility for maintaining the production system has been transferred from department E17 to department C87. The conventions for naming a zone in this establishment is department number plus four characters that help describe the zone's content. You want to change the zone name from E17SYSP to C87SYSP. Assume no changes are required for DDDEF entries in the zone to be renamed, because the target libraries being maintained are not changing.

Before the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone E17SYSP is in data set SMPE.CSI. There is no ZONEINDEX yet for C87SYSP; it is created by the ZONERENAME command. You now perform the ZONERENAME using the following set of SMP/E commands:

```
SET      BDY(GLOBAL)        /* Set to global.           */.
ZONERENAME(E17SYSP)         /* Rename zone E17SYSP      */
         TO(C87SYSP)        /* to new name C87SYSP      */
         SAMEDATASET        /* within the same CSI.     */
                            /* TOTYPE not required      */
                            /* because the zone type    */
                            /* remains the same.        */.
```

After the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone C87SYSP is on data set SMPE.CSI, and the ZONEINDEX entry for E17SYSP has been removed. The zone definition for E17SYSP in data set SMPE.CSI has been changed so it is now the zone definition for C87SYSP.

Example 2: Creating a target zone from a distribution zone

A common use of ZONERENAME involves creating a new target zone after having performed a system generation.

Let us Assume that you have a distribution zone named SYSDLIB in data set SMPE.CSI. The distribution libraries described by this zone have been used to perform a system generation, and now you want to create a target zone describing the content of your new operating system. You want to call your new target zone SYSPROD. You also want to change the default OPTIONS entry for the renamed zone to SYSOPTS. Further, you want to indicate that the renamed zone is related to the SYSDLIB zone; that is, SYSDLIB is the distribution zone controlling the distribution libraries for the target zone.

First, you make a copy of the SMP.CSI data set containing the distribution zone SYSDLIB (SMPE.CSI.) using the access method services REPRO command. Let us assume the name of the copied data set is PROD.CSI.

Before the ZONERENAME operation, the GLOBALZONE ZONEINDEX indicates that zone SYSDLIB is on data set SMPE.CSI. There is no GLOBALZONE ZONEINDEX for SYSPROD. You can now use the following

ZONERENAME command to rename the copy of SYSDLIB in PROD.CSI to SYSPROD, change the zone type to TARGET, and modify the OPTIONS and RELATED subentries:

```
SET      BDY(GLOBAL)        /* Set to global.           */.
ZONERENAME(SYSDLIB)         /* Rename zone SYSDLIB      */
         TO(SYSPROD)        /* to new name SYSPROD in a */
         NEWDATASET(        /* copy of a CSI data set   */
                    PROD.CSI   /* named PROD.CSI.       */
                   )        /*                          */
         TOTYPE(TARGET)     /* Change to a target type. */
         OPTIONS(SYSOPTS)   /* Change default OPTIONS.  */
         RELATED(SYSDLIB)   /* Related to DLIB zone.    */.
```

After the ZONERENAME operation, the GLOBALZONE ZONEINDEX still indicates that zone SYSDLIB is on data set SMPE.CSI; it has not been removed. A new GLOBALZONE ZONEINDEX entry has been created indicating that zone SYSPROD is on data set PROD.CSI. If the zone definition for SYSPROD is listed, the user sees that the default OPTIONS value is SYSOPTS and that the target zone is related to distribution zone SYSDLIB.

Other than the zone definition, the contents of the renamed zone are unchanged. You must make changes to the DDDEF entries in SYSPROD as required.

After performing these steps, you have a new target zone containing information describing the various modules, macros, and source as they exist on the distribution libraries. The additional information required in the target zone is the description of how the various distribution library elements are combined and installed into the target zone libraries. This information is added to the target zone by using the JCLIN command, with stage 1 system generation output as input to SMP/E.

Also, if you are using the dynamic allocation facility in SMP/E, you may have to add or modify some of the DDDEF entries in the new target zone. For an example of priming the new target zone after a full system generation, see “Examples” on page 440.

Example 3: Creating a duplicate copy of a CSI data set

ZONERENAME can help you make a duplicate copy of a CSI, such as for backup or test. For example, suppose you have a CSI named SMPE.SYSPROD.CSI containing a target zone named SYSTGT and a DLIB zone named SYSDLB. You plan to install a new release of z/OS, but before doing so, you want to create a copy of this CSI as backup. The new CSI is to be named SMPE.SYSBKUP.CSI, the copy of SYSTGT is to be called SYSTGTB, and the copy of SYSDLB is to be called SYSDLBB.

Follow these steps to make a backup copy of your CSI:

1. Use access method services (AMS) to define the new CSI data set, named SMPE.SYSBKUP.CSI. Do **not** initialize this new CSI with GIMZPOOL; you are going to copy an existing CSI data set into it.
2. Use AMS REPRO to copy the original CSI data set, SMPE.SYSPROD.CSI, to the newly defined data set, SMPE.SYSBKUP.CSI.

   At this point, the zones in SMPE.SYSBKUP.CSI still have the same names as the zones in SMPE.SYSPROD.CSI. Unlike the zones in SMPE.SYSPROD.CSI, however, none of the zones in SMPE.SYSBKUP.CSI are defined by GLOBALZONE ZONEINDEX subentries.
3. Use the ZONERENAME command to:
   - Rename the zones in SMPE.SYSBKUP.CSI.
   - Add ZONEINDEX subentries to point to those zones.
   - Update the RELATED zone subentries for the zones in SMPE.SYSBKUP.CSI.

   Here is an example:

```
SET      BDY(GLOBAL)        /* Set to global.           */.
ZONERENAME(SYSTGT)          /* Rename zone SYSTGT       */
         TO(SYSTGTB)        /* to new name SYSTGTB in   */
         NEWDATASET(        /* new CSI data set         */
         SMPE.SYSBKUP.CSI)  /* SMPE.SYSBKUP.CSI.        */
         RELATED(SYSDLBB)   /* Related zone is SYSDLBB. */.
ZONERENAME(SYSDLB)          /* Rename zone SYSDLB       */
```

```
          TO(SYSDLBB)        /* to new name SYSDLBB in   */
          NEWDATASET(        /* new CSI data set         */
          SMPE.SYSBKUP.CSI)  /* SMPE.SYSBKUP.CSI.        */
          RELATED(SYSTGTB)   /* Related zone is SYSTGTB. */.
```

After the ZONERENAME operation, all the information about the zones in the old CSI (SMPE.SYSPROD.CSI) remains unchanged, and the following changes have been made for the zones in the new CSI (SMPE.SYSBKUP.CSI):

- The zones in SMPE.SYSBKUP.CSI have been renamed to SYSTGTB and SYSDLBB.
- New GLOBALZONE ZONEINDEX subentries point to zones SYSTGTB and SYSDLBB on data set SMPE.SYSBKUP.CSI.
- The TARGETZONE entry for SYSTGTB indicates that the related DLIB zone is SYSDLBB, and the DLIBZONE entry for SYSDLBB indicates that the related target zone is SYSTGTB.

Other than the zone definition entries, the content of the renamed zones remain unchanged.

4. Add or change DDDEF entries as appropriate in the SYSTGTB and SYSDLBB zones.

Processing

With the ZONERENAME command, you can rename a specified distribution or target zone. SMP/E supports the following variations of renaming:

- Distribution zone to distribution zone
- Target zone to target zone
- Distribution zone to target zone

The rename operation itself is very simple. Before doing it, however, SMP/E makes sure the correct parameters have been entered and that the rename request is valid. If any of the checks fail, an error message is issued, and the rename operation is terminated. The following checks are made:

- *oldzonename* and *newzonename* must not be the same.
- The new zone must not already exist in the data set it is to reside in.
- A GLOBALZONE ZONEINDEX entry for the new zone name must not already exist.
- If TOTYPE(TARGET) is specified:
  - The GLOBALZONE ZONEINDEX entry for the old zone name must be a distribution zone.
  - The NEWDATASET operand must also be specified, and the SAMEDATASET operand must **not** be specified.
- If NEWDATASET is specified, the data set name specified in the NEWDATASET operand must not be the same as the data set value for the old zone name.
- If SAMEDATASET is specified, a GLOBALZONE ZONEINDEX entry must already exist for the old zone name.
- No cross-zone subentry from the zone being renamed refers to the new zone name.

If all validity checking is successful, the renaming can be done. The following operations are done to rename a zone:

- A GLOBALZONE ZONEINDEX entry is added for the new zone name. The data set name is the same as either the value in the old zone (if SAMEDATASET was specified) or the name specified in the NEWDATASET operand. The zone type is the same as the old zone type unless the TOTYPE(TARGET) operand is coded, in which case the type is set to TARGET.
- A zone definition entry (either TARGETZONE or DLIBZONE entry) is created for the new zone; the zone definition for the old zone is taken as a base.

  If the RELATED operand or the OPTIONS operand, or both, were specified, that information is used in the zone definition entry; otherwise, that data remains as is in the old zone.
- The old zone definition entry is deleted.

- If the SAMEDATASET operand is specified, the ZONEINDEX entry for the old zone is deleted.

Zone and data set sharing considerations

The following identifies the phases of ZONERENAME processing and the zones and data sets that SMP/E may require for exclusive or shared use during each phase. For more information about command phases and data set sharing in SMP/E, see Appendix B, “Sharing SMP/E data sets,” on page 543.

1. Initialization

   Global zone
   Read without enqueue.

   Target zone
   Read without enqueue.

   DLIB zone
   Read without enqueue.

   Note: Either the target zone or the distribution zone accessed during this phase is the name specified on the TO operand (that is, the new name of the zone). Except for an error condition, this zone should not exist.

2. ZONERENAME processing

   Global zone
   Update with exclusive enqueue.

   Target zone
   Update with exclusive enqueue.

   DLIB zone
   Update with exclusive enqueue.

   Note:

   a. Either the target zone or the distribution zone is accessed, according to the zone type specified in the previous SET command.

   b. The zone accessed in this phase is the zone to be renamed.

3. Termination

   All resources are freed.

Chapter 34. SMP/E reports

This chapter describes all the reports produced by SMP/E. The following chart lists these reports and the page on which each is found. The heading of the report indicates the service level of SMP/E that is installed on your system. The SMP/E service level appears as SMP/E *nn.nn*. For example, SMP/E 36.*nn* is V3R6.0 service level *nn*.

Report	Page
BUILDMCS entry summary report	“BUILDMCS entry summary report” on page 458
BUILDMCS function summary report	“BUILDMCS function summary report” on page 460
Bypassed HOLD reason report	“Bypassed HOLD reason report” on page 461
Causer SYSMOD summary report	“Causer SYSMOD summary report” on page 463
CLEANUP summary report	“CLEANUP summary report” on page 464
Cross-zone requisite SYSMOD report	“Cross-zone requisite SYSMOD report” on page 465
Cross-zone summary report	“Cross-zone summary report” on page 468
Deleted SYSMOD report	“Deleted SYSMOD report” on page 471
Element summary report	“Element summary report” on page 472
Exception SYSMOD report	“Exception SYSMOD report” on page 476
File allocation report	“File allocation report” on page 479
GENERATE summary report	“GENERATE summary report” on page 482
GZONEMERGE report	“GZONEMERGE report” on page 485
JCLIN cross-reference report	“JCLIN cross-reference report” on page 489
JCLIN summary report	“JCLIN summary report” on page 490
LINK LMODS summary report	“LINK LMODS summary report” on page 493
LIST summary report	“LIST summary report” on page 495
Missing FIXCAT SYSMOD report	“Missing FIXCAT SYSMOD report” on page 496
MOVE/RENAME/DELETE report	“MOVE/RENAME/DELETE report” on page 498
RECEIVE exception SYSMOD data report	“RECEIVE exception SYSMOD data report” on page 503
RECEIVE summary report	“RECEIVE summary report” on page 505
RECEIVE product summary report	“Receive product summary report” on page 510
REJECT summary report	“REJECT summary report” on page 512
Summary of bypassed and unresolved HOLD reason report	“Summary of bypassed and unresolved HOLD reason report” on page 520
SOURCEID report	“SOURCEID report” on page 518
SYSMOD comparison HOLDDATA report	“SYSMOD comparison HOLDDATA report” on page 523
SYSMOD comparison report	“SYSMOD comparison report” on page 521

Report	Page
SYSMOD regression report	“SYSMOD regression report” on page 526
SYSMOD status report	“SYSMOD status report” on page 527
UNLOAD summary report	“UNLOAD summary report” on page 530
Unresolved HOLD reason report	“Unresolved HOLD reason report” on page 531
ZONEEDIT summary report	“ZONEEDIT summary report” on page 536
ZONEMERGE report	“ZONEMERGE report” on page 538

BUILDMCS entry summary report

This report, produced by the BUILDMCS command, summarizes the processing done for every eligible entry. The report lists the entries alphabetically by type and, within types, alphabetically by name. SMP/E produces one report for each superseding function it creates.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO GLOBAL ZONE         DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BUILDMCS SUMMARY REPORT FOR FMID fmid

ENTRY      ENTRY     ENTRY           CURRENT
TYPE       NAME      STATUS          RMID           COMMENTS

aaaaaaaa   bbbbbbbb  ccccccccccc     ddddddd    eeeeeeeeeeeeeeeeeeeeeee…
aaaaaaaa   bbbbbbbb  ccccccccccc     ddddddd    eeeeeeeeeeeeeeeeeeeeeee…
                                                eeeeeeeeeeeeeeeeeeeeeee…
aaaaaaaa   bbbbbbbb  ccccccccccc     ddddddd    eeeeeeeeeeeeeeeeeeeeeee…
```

Figure 31. BUILDMCS entry summary report: standard format

These are the fields in the report:

fmid
is an FMID that was specified on the FORFMID operand.

ENTRY TYPE
is the entry type: ASSEM, data element, DDDEF, DLIB, hierarchical file system element, LMOD, MAC, MOD, and SRC.

ENTRY NAME
is the name of the entry.

ENTRY STATUS
describes whether the entry was selected for inclusion in the superseding function. It might be one of the following entries:

NOT SELECTED
This entry was not selected for inclusion in the superseding function. See the COMMENTS for reasons the entry may not be selected.

REQUIRED
This is for DDDEF entries. It identifies that this DDDEF is required to be defined to the system where this superseding function is to be received, applied, and accepted. See the COMMENTS field for information on zone types that require this definition.

SELECTED
This entry was selected for inclusion in the superseding function. See the COMMENTS field for more information that is important about the inclusion of this entry.

CURRENT RMID
is the RMID that appears in the element entry at the end of command processing. It appears only if the element is processed successfully.

COMMENTS
if present, provides additional information about the entry. The following comments can appear:

ENTRY NOT FOUND
The specified entry was needed for processing, but is not defined in the set-to zone. This is an error condition.

If the specified entry is a DDDEF, the values for the element MCS FROMDS operands will contain only the distribution library ddname. To correct this, the user must take one of the following actions:

1. Add the missing DDDEF entry and re-run the BUILDMCS command, or
2. Update the created MCS to specify the distribution library data set names on the element MCS FROMDS operands.

ELEMENT NOT FOUND - ASSUMED DELETED
The function or an associated SYSMOD contains an element that is not defined in the set-to zone. This is not an error, but SMP/E assumes the element has been deleted.

CURRENT FMID IS yyyyyyy
The function or associated SYSMOD contains an element that is defined in the set-to zone as belonging to FMID yyyyyyy.

yyyyyyy
FMID that currently owns the entry.

LMOD CONTAINS MODDELS
The LMOD entry contains MODDEL subentries indicating that modules were part of this load module, but have been deleted.

LMOD CONTAINS CROSS-ZONE MODS
The LMOD entry contains cross-zone modules.

LMOD HAS MODS FROM MULTIPLE FMIDS
The LMOD entry contains modules from more than the specified FMID.

MOD HAS CROSS-ZONE LMODS
The MOD entry contains cross-zone load modules.

MOD HAS NO LMODS
The module is defined as an entry belonging to the specified FMID, however, the MOD does not exist in any load modules.

REQUIRED FOR CALLLIBS IN TARGET ZONE
The DDDEF is needed for CALLLIBS processing and must be defined in the target zone.

REQUIRED FOR LMOD SIDE DECK LIBRARY IN TARGET ZONE
The DDDEF is needed for the load module's side deck library and must be defined in the target zone.

REQUIRED FOR LMOD UTILITY INPUT IN TARGET ZONE
The DDDEF is needed for the load module's utility input and must be defined in the target zone.

REQUIRED IN TARGET AND DISTRIBUTION ZONE
DDDEF must be defined in both the target and distribution zone.

REQUIRED IN TARGET ZONE
DDDEF must be defined in the target zone.

Example: BUILDMCS entry summary report

```
PAGE nnnn  - NOW SET TO GLOBAL ZONE         DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BUILDMCS ENTRY SUMMARY REPORT FOR FMID HMP1800

ENTRY      ENTRY     ENTRY           CURRENT
TYPE       NAME      STATUS          RMID           COMMENTS

CLIST      CIDCGADS  SELECTED        UR43934
CLIST      CIDCGBRO  SELECTED        HMP1800
CLIST      CIDCGCLN  SELECTED        UR44925
CLIST      LOSTENTY  NOT SELECTED                   ENTRY NOT FOUND
⋮
CLIST      CIDCSVAT  SELECTED        UR43787
DDDEF      AGIMCLS0  REQUIRED                       REQUIRED IN TARGET AND DISTRIBUTION ZONE
DDDEF      AGIMLMD0  REQUIRED                       REQUIRED IN TARGET AND DISTRIBUTION ZONE
⋮
DDDEF      SGIMLMD0  REQUIRED                       REQUIRED IN TARGET ZONE
LMOD       CIDXGADB  SELECTED
LMOD       CIDXGADS  SELECTED
LMOD       CIDXGALC  SELECTED
⋮
MAC        FAUXMAC   NOT SELECTED                   ELEMENT NOT FOUND - ASSUMED DELETED
MAC        GIMBSTRP  SELECTED        UR41504
MAC        GIMMPUXP  SELECTED        HMP1800
MAC        GIMZPOOL  SELECTED        HMP1800
MAC        SGAMA402  SELECTED        HMP1800
MAC        SGGIMSMP  SELECTED        HMP1800
MAC        SGHMA402  SELECTED        HMP1800
MOD        CIDXGACK  SELECTED        HMP1800
MOD        CIDXGADB  SELECTED        HMP1800
MOD        CIDXGADS  SELECTED        HMP1800
⋮
MOD        GIMZUDRV  SELECTED        HMP1800
PARM       GIMOPCDE  SELECTED        HMP1800
SAMP       GIMSAMPU  SELECTED        HMP1800
SAMP       GIMZPDFT  SELECTED        UR44780
```

Figure 32. BUILDMCS entry summary report: sample report

BUILDMCS function summary report

This report, produced by the BUILDMCS command, summarizes the processing done for every FMID specified on the BUILDMCS command. The report lists the functions alphabetically by base function. Dependent functions are listed alphabetically after its base function. The associated SYSMODS listed for each function are also listed in alphabetical order.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BUILDMCS FUNCTION SUMMARY REPORT

NOTE:  '*' INDICATES THE ASSOCIATED SYSMOD IS IN ERROR

FUNCTION  STATUS        FMID      ASSOCIATED SYSMODS

aaaaaaa   bbbbbbbbbbbb  ccccccc   ddddddd…
                                  ddddddd…
aaaaaaa   bbbbbbbbbbbb  ccccccc   ddddddd…
aaaaaaa   bbbbbbbbbbbb  ccccccc   ddddddd…
```

Figure 33. BUILDMCS function summary report

These are the fields in the report:

FUNCTION
identifies the function SYSMOD that was processed.

STATUS
describes the validity of the function. It can be one of the following fields:

DELETED
The specified function was deleted by another SYSMOD.

ERROR
The specified function has a status of error in its SYSMOD entry.

NOT FOUND
The specified function was not defined in the set-to zone.

NOT SELECTED
The specified function is not a function SYSMOD.

SELECTED
The specified function was valid for BUILDMCS processing.

SUPERSEDED
The specified function was included and completely replaced by another SYSMOD.

FMID
base FMID associated with the specified dependent function. If the specified function is a base function, the FMID field is blank.

ASSOCIATED SYSMODS
lists installed SYSMODs whose FMID matches the specified function. These installed SYSMODs will be included in the superseding function. The SYSMODs are preceded by an '*' when the associated SYSMOD is in error.

Example: BUILDMCS function summary report

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BUILDMCS FUNCTION SUMMARY REPORT

NOTE:  '*' INDICATES THE ASSOCIATED SYSMOD IS IN ERROR

FUNCTION  STATUS        FMID      ASSOCIATED SYSMODS

HMP1700   DELETED
JMP1701   DELETED
HMP1800   SELECTED                 UR41007   UR41531   UR41534   UR41820   UR41937   UR41949   UR42025   UR42152
                                   UR42277   UR42341   UR42393   UR42497   UR42499   UR42558   UR42726   UR43011
                                   UR43171   UR43350   UR43492   UR43715   UR43934   UR43968   UR44005   UR44036
                                   UR44178   UR44291   UR44433  *UR44593  *UR44780
JMP1801   SELECTED      HMP1800   UR41008   UR41507   UR41823   UR41940   UR41956   UR42028   UR42155   UR42280
                                   UR42342   UR42396   UR42496   UR42561   UR42729   UR43015   UR43175   UR43354
                                   UR43791   UR43936   UR44038   UR44435
```

Figure 34. BUILDMCS Function Summary Report: Sample Report

Bypassed HOLD reason report

This report is produced at the completion of APPLY or ACCEPT processing to identify HOLD reason IDs that were bypassed on the APPLY or ACCEPT command. When this report is presented for a non-CHECK mode run, it provides confirmation about which HOLD conditions were bypassed.

You can use this report for planning a non-CHECK mode run by taking one of these actions:

- Run an APPLY or ACCEPT in CHECK mode and bypass all HOLDSYS and HOLDUSER conditions.
- The "Bypassed HOLD Reason Report" is produced for this run and contains all the HOLDs that would be bypassed for SYSMODs that would be applied or accepted successfully.
- Review the report and the information contained in the HOLDDATA. From this (and possibly other investigation) you can determine what actions must be taken and which HOLDs can be safely bypassed for the non-CHECK mode run.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BYPASSED HOLD REASON REPORT FOR xxxxxxx yyyyy PROCESSING

TYPE      REASON ID   FMID      SYSMOD    ++HOLD DATA
------    ---------   -------   -------   -----------------------------------------------------------------------

aaaaaa    bbbbbbb     ccccccc   ddddddd   eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                      ccccccc   ddddddd   eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
          bbbbbbb     ccccccc   ddddddd   eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
aaaaaa    bbbbbbb     ccccccc   ddddddd   eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                          eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

Figure 35. Bypassed HOLD reason report: standard format

These are the fields in the report:

xxxxxx
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
is CHECK, if CHECK was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

TYPE
is the type of ++HOLD for the following reason ID.

ERROR
indicates an error hold.

FIXCAT
indicates a fix category hold.

SYSTEM
indicates a system hold.

USER
indicates a user hold.

REASON ID
is a list of one or more HOLD reason IDs that were bypassed for the named SYSMOD.

FMID
is the FMID of the SYSMOD in the next field of the report.

SYSMOD
is the ID of a SYSMOD that had some HOLD reason ID bypassed.

++HOLD DATA
refers to one of the following MCS:

- the ++HOLD MCS that was received for the HOLD reason ID. This data may span several lines and is provided for each reason ID, or
- an indication that the ++HOLD MCS has been suppressed, for the reason IDs specified in the Suppress HOLDDATA (SUPPHOLD) subentry of the active OPTIONS entry

Example: Bypassed HOLD reason report

Note: A0 and DOC appear as SYSTEM reason IDs in both the "Unresolved HOLD Reason Report" and the "Bypassed HOLD Reason Report". This could occur only if you bypassed the SYSTEM holds for specific SYSMODs.

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BYPASSED HOLD REASON REPORT FOR APPLY CHECK PROCESSING

TYPE    REASON ID  FMID     SYSMOD   ++HOLD DATA

------  ---------  -------  -------  ------------------------------------------------------------------------

SYSTEM  ACTION     HXY0022  UZ06853  ++HOLD (UZ06853) FMID(HXY0022) SYSTEM REASON(ACTION)
                                      COMMENT(INCREASE TARGET DATA SET SIZE OTHERWISE
                                      YOU WILL X37 DUE TO INCREASED LMOD SIZES).

        ACTION     JXY0033  UZ64781  ++HOLD (UZ64781) FMID(JXY0033) SYSTEM REASON(ACTION)
                                      COMMENT(SYSTEM MUST BE IPLED TO ACTIVATE THE CHANGE).

        AO         HXY0022  UZ06444  ++HOLD (UZ06444) FMID(HXY0022) SYSTEM REASON(AO)
                                      COMMENT(MUST MASSAGE AUTO OPS DUE TO NEW MESSAGES).

        DOC        HXY0022  UZ06444  ++HOLD (UZ06444) FMID(HXY0022) SYSTEM REASON(DOC)
                                      COMMENT(NEW MESSAGES).

        DOC        JXY0033  UZ64781  ++HOLD (UZ64781) FMID(JXY0033) SYSTEM REASON(DOC)
                                      COMMENT(PARMLIB MEMBER FORMAT CHANGE).
```

Figure 36. Bypassed HOLD Reason Report: Sample Report

Example: Bypassed HOLD reason report

Figure 37 on page 463 is an illustration of a page of a possible Bypassed HOLD Reason Report. The following example had an active OPTIONS entry for zone SYSP001. In this active OPTIONS entry, the subentry SUPPHOLD was specified for the DOC Reason ID.

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

BYPASSED HOLD REASON REPORT FOR APPLY CHECK PROCESSING


TYPE      REASON ID  FMID    SYSMOD   ++HOLD DATA
----      ---------  ----    ------   ------------------------------------------------------------------------
SYSTEM    ACTION     HXY0022 UZ06853  ++HOLD (UZ06853) FMID(HXY0022) SYSTEM REASON(ACTION)
                                        COMMENT(INCREASE TARGET DATA SET SIZE OTHERWISE
                                        YOU WILL X37 DUE TO INCREASED LMOD SIZES).

          AO         HXY0022 UZ06444  ++HOLD (UZ06444) FMID(HXY0022) SYSTEM REASON(AO)
                                        COMMENT(MUST MASSAGE AUTO OPS DUE TO NEW MESSAGES).

          DOC        HXY0022          * ONE OR MORE RESTART HOLDS ARE SUPPRESSED FOR HXY0022.
```

Figure 37. Bypassed HOLD reason report: sample report - suppressed HOLDDATA

Causer SYSMOD summary report

To reduce the work needed to determine which errors caused SYSMODs to fail, SMP/E performs root cause analysis for the ACCEPT, APPLY, and RESTORE commands to identify causer SYSMODs. A causer SYSMOD is a SYSMOD whose failure has led to the failure of other SYSMODs. The types of errors SMP/E analyzes to determine causer SYSMODs include the following situations:

- Held SYSMODs
- Missing requisite SYSMODs
- Utility program failures: copy, update, assembler, link-edit utility, zap
- Out-of-space conditions: *x*37 ABENDs
- Missing DD statements and other allocation errors
- ID errors (a SYSMOD does not supersede or specify as a prerequisite an RMID or a UMID of an element)
- JCLIN errors (syntax errors)

The Causer SYSMOD Summary report lists the causer SYSMODs and a summary of the related messages describing the errors that need to be fixed to successfully process the SYSMODs. (Subsequent messages generated because of these initial errors are not included in the report.)

This report is produced during the processing of APPLY, ACCEPT, and RESTORE commands. It is **not** produced when there are no errors or when there are certain types of errors, such as unusual VSAM errors (as indicated by GIM443*xx* messages).

Format and explanation of data

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

CAUSER SYSMOD SUMMARY REPORT FOR xxxxxxx PROCESSING

CAUSER     FMID     MESSAGE ID  PAGE    ERROR DESCRIPTION AND POSSIBLE CAUSES
aaaaaaa    bbbbbbb  ccccccccc     dd    eeeeeeee…
aaaaaaa    bbbbbbb  ccccccccc     dd    eeeeeeee…
aaaaaaa    bbbbbbb  ccccccccc     dd    eeeeeeee…
aaaaaaa    bbbbbbb  ccccccccc     dd    eeeeeeee…
aaaaaaa    bbbbbbb  ccccccccc     dd    eeeeeeee…
```

Figure 38. Causer SYSMOD summary report: standard format

These are the fields in the report:

XXXXXXXX
is the SMP/E command being processed: APPLY, ACCEPT, or RESTORE.

CAUSER
identifies the SYSMOD whose failure led to the failure of other SYSMODs.

FMID
identifies the FMID for the causer SYSMOD when that information is available. Otherwise, the field is blank.

MESSAGE ID
is the message identification number for the message describing the error that caused this SYSMOD to fail (8-character alphanumeric string plus a 1-character severity value).

PAGE
identifies the page number in SMPOUT output where the SYSMOD failure message is located.

ERROR DESCRIPTION AND POSSIBLE CAUSES
provides a summary of the error and, when feasible, a list of possible causes of the error.

Example: Causer SYSMOD summary report for APPLY processing

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

CAUSER SYSMOD SUMMARY REPORT FOR APPLY PROCESSING

CAUSER     FMID     MESSAGE ID  PAGE    ERROR DESCRIPTION AND POSSIBLE CAUSES

AZ00001             GIM37301E      1    THIS SYSMOD HAS MORE THAN ONE APPLICABLE ++VER.

EYC0437    HAE1500  GIM24001I     30    ASSEMBLER ERROR FOR MODULE IATDMER IN LIBRARY LINKLIB FOR SYSMOD
                                        EYC0437. THE SEQUENCE NUMBER IS 00001.
                                        --- POSSIBLE CAUSES ---
                                        1. THE CORRECT DEFAULT UTILITY RETURN CODE WAS NOT SPECIFIED.
                                        2. THE OPTIONS ENTRY DOES NOT CONTAIN THE CORRECT UTILITY ENTRY.

UP42613    HAE1500  GIM35902I      6    SYSTEM HOLD DEP WAS NOT RESOLVED.

UY12605    HAE1500  GIM40501E     30    THE DISTLIB IN ++MOD BLMGDBWR IS DIFFERENT FROM THE ELEMENT ENTRY IN THE ZONE.
                    GIM40501E     30    THE DISTLIB IN ++MOD BLMGDBWI IS DIFFERENT FROM THE ELEMENT ENTRY IN THE ZONE.

UZ62368    JTC2412  GIM35901I      2    ERROR HOLD AZ71745 IS NOT RESOLVED.
                    GIM35901I      2    ERROR HOLD AZ75533 IS NOT RESOLVED.

UZ75098    HJE2330  GIM35909I     27    COREQUISITE SYSMOD UZ75006 WAS EXCLUDED.
```

Figure 39. Causer SYSMOD summary report: sample report for APPLY

CLEANUP summary report

This report is produced during CLEANUP processing to summarize the elements or aliases deleted from each target zone work data set. If no entries were deleted (for example, there were no entries or an abend occurred before report data was collected), the CLEANUP report states `THERE WAS NO CLEANUP DONE`.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                CLEANUP SUMMARY REPORT

NOTE: * - INDICATES THE ENTRY IS AN ALIAS OF THE PRECEDING MAJOR MEMBER NAME

DDNAME   ENTRY DELETED

aaaaaaa  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb
aaaaaaa  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb
aaaaaaa  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb
aaaaaaa  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb  bbbbbbbb
```

Figure 40. CLEANUP summary report: standard format

These are the fields in the report:

DDNAME
is the ddname of the data set from which data was deleted: SMPLTS, SMPMTS, SMPSCDS, or SMPSTS. Each ddname is shown only once.

ENTRY DELETED
is a list of the elements or alias names deleted from the data set. If there is an alias name, it follows the real name of the member and is preceded by an asterisk.

Example: CLEANUP summary report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                CLEANUP SUMMARY REPORT

NOTE: * - INDICATES THE ENTRY IS AN ALIAS OF THE PRECEDING MAJOR MEMBER NAME

DDNAME   ENTRY DELETED

SMPSCDS  UZ00010   UZ00020
SMPMTS   MAC02     MAC03
SMPSTS   SRC11     SRC12
```

Figure 41. CLEANUP summary report: sample report

Cross-zone requisite SYSMOD report

This report is produced:

- At the completion of REPORT CROSSZONE processing.
- For APPLY and ACCEPT commands, when a zone group has been established for APPLY or ACCEPT command processing.

Report for REPORT CROSSZONE processing

For REPORT CROSSZONE processing, the report shows the affected zones, the requisites that must be installed in each zone, and the SYSMODs that contained ++IF statements naming the requisites. The requisite SYSMODs are listed in alphanumerical order by SYSMOD ID for each FMID in each zone.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                CROSSZONE REQUISITE SYSMOD REPORT FOR xxxxxx

 ZONE      REQUIRES            CAUSER
 NAME     FMID     SYSMOD   RECEIVED  SYSMOD   FMID     ZONE

 zzzzzzz  fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww
          fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww

 zzzzzzz  fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww
          fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww
```

Figure 42. Cross-zone requisite SYSMOD report: standard format for REPORT CROSSZONE

These are the fields in the report:

xxxxxx
is the command to be used to install the SYSMODs in the report: ACCEPT if the report was done for distribution zones, or APPLY if the report was done for target zones.

ZONE NAME
is the zone where requisite SYSMODs must be installed.

REQUIRES FMID
is the FMID for which requisite SYSMODs must be installed.

REQUIRES SYSMOD
is the ID of a requisite SYSMOD. If there are no requisite SYSMODs for a particular zone, you see NONE in the REQUIRES SYSMOD field.

RECEIVED
indicates whether the requisite SYSMOD has been received. YES in the RECEIVED field means the SYSMOD has been received, and NO means it has not. You must receive all requisite SYSMODs before installing them.

CAUSER SYSMOD
is the ID of the SYSMOD that contained the ++IF statement for the requisite.

CAUSER FMID
is the FMID for the causer SYSMOD. If UNKNOWN is in the CAUSER FMID field, the causer SYSMOD was installed with an obsolete release of SMP/E. This is not an error.

CAUSER ZONE
is the zone that contained the causer SYSMOD.

Example: Cross-zone requisite SYSMOD report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT
OUTPUT

                  CROSSZONE REQUISITE SYSMOD REPORT FOR APPLY

 ZONE              REQUIRES           CAUSER
 NAME     FMID     SYSMOD   RECEIVED  SYSMOD   FMID     ZONE

 TZONE01           NONE

 TZONE23  HJS5502  UZ00027  YES       UZ00025  EJS5502  TZONE43
          HJS5502  UZ00028  NO        UZ00026  EJS5502  TZONE43

 TZONE43  EJS1201  UZ00023  YES       UZ00011  EBB1202  TZONE01
          EJS1201  UZ00024  YES       UZ00013  EBB1202  TZONE01
```

Figure 43. Cross-Zone Requisite SYSMOD Report: Sample Report for REPORT CROSSZONE

Report for APPLY and ACCEPT processing

For APPLY and ACCEPT processing, the report shows all the zones in the current zone group, the requisites that must be installed in each zone, and the SYSMODs that contained ++IF statements naming the requisites. If a zone group has been established for the APPLY or ACCEPT command, this report immediately follows the SYSMOD Status Report. The requisite SYSMODs are listed in alphanumerical order by SYSMOD ID for each FMID in each zone.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                CROSSZONE REQUISITE SYSMOD REPORT FOR xxxxxx PROCESSING

                NOTE:  CROSS-ZONE REQUISITES CREATED BY THE CURRENT COMMAND ARE PRECEDED BY '*'.

 ZONE      REQUIRES            CAUSER
 NAME     FMID     SYSMOD   RECEIVED  SYSMOD   FMID     ZONE

 zzzzzzz  fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww
          fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww

 zzzzzzz  fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww
          fffffff  sssssss  xxx       ttttttt  rrrrrrr  wwwwwww
```

Figure 44. Cross-zone requisite SYSMOD report: standard format for APPLY and ACCEPT

These are the fields in the report:

xxxxxx
is the name of the command (APPLY or ACCEPT) that caused the report to be produced.

ZONE NAME
is the zone where requisite SYSMODs must be installed.

REQUIRES FMID
is the FMID for which requisite SYSMODs must be installed.

REQUIRES SYSMOD
is the ID of a requisite SYSMOD. If there are no requisite SYSMODs for a particular zone, you see NONE in the REQUIRES SYSMOD field. If the need for a requisite SYSMOD is newly created by the current APPLY or ACCEPT command, an asterisk (*) precedes the SYSMOD ID.

RECEIVED
indicates whether the requisite SYSMOD has been received. YES in the RECEIVED field means the SYSMOD has been received, and NO means it has not. You must receive all requisite SYSMODs before installing them.

CAUSER SYSMOD
is the ID of the SYSMOD that contained the ++IF statement for the requisite.

CAUSER FMID
is the FMID for the causer SYSMOD. If UNKNOWN is in the CAUSER FMID field, the causer SYSMOD was installed with an obsolete release of SMP/E. This is not an error.

CAUSER ZONE
is the zone that contained the causer SYSMOD.

Example: Cross-zone requisite SYSMOD report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                CROSSZONE REQUISITE SYSMOD REPORT FOR APPLY PROCESSING

                NOTE:  CROSS-ZONE REQUISITES CREATED BY THE CURRENT COMMAND ARE PRECEDED BY '*'.

 ZONE      REQUIRES           CAUSER
 NAME     FMID     SYSMOD   RECEIVED  SYSMOD   FMID     ZONE

 TZONE01           NONE

 TZONE23  HJS5502  UZ00027  YES       UZ00025  EJS5502  TZONE43
          HJS5502  UZ00028  NO        UZ00026  EJS5502  TZONE43

 TZONE43  EJS1201  UZ00023  YES       UZ00011  EBB1202  TZONE01
          EJS1201 *UZ00024  YES       UZ00013  EBB1202  TZONE01
```

Figure 45. Cross-Zone Requisite SYSMOD Report: Sample Report for APPLY

Cross-zone summary report

This report, produced during APPLY and RESTORE processing, summarizes the cross-zone work that has been done, the cross-zone work that has not been done, and why. The cross-zone work resulting from renamed LMODs is summarized in the Move/Rename/Delete report.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                            CROSS-ZONE SUMMARY REPORT FOR xxxxxx yyyyyy PROCESSING

NOTE: SEE THE MOVE/RENAME/DELETE REPORT FOR A SUMMARY OF CROSS-ZONE WORK CAUSED BY RENAMED LMODS

CROSS            LMOD                                        LKED
ZONE    LMOD     SYSLIB   MODULE   _________ACTION_________  RC  ____________REASON_____________   SYSMOD

aaaaaaa bbbbbbbb ccccccc  ddddddd  eeeeeeeeeeeeeeeeeeeeeee fff  gggggggggggggggggggggggggggggggg   hhhhhhh
```

Figure 46. Cross-zone summary report

These are the fields in the report:

xxxxxxx
is the SMP/E command being processed: APPLY or RESTORE.

yyyyyyy
is CHECK, if CHECK was specified on the command. Otherwise, this field is blank.

CROSS-ZONE
is the cross-zone name.

LMOD
is the name of the cross-zone LMOD being updated.

LMOD SYSLIB
is the SYSLIB for the cross-zone LMOD being updated.

Note: SMPLTS appears in this column when a module is linked into a base version of a load module in the SMPLTS data set.

MODULE
is the name of the changed module from the set-to zone that caused the cross-zone LMOD to be updated.

ACTION
indicates whether the cross-zone work was done.

MOD LINKED INTO LMOD
The module from the set-to zone was replaced in the cross-zone LMOD.

MOD NOT LINKED INTO LMOD
The module from the set-to zone was not replaced in the cross-zone LMOD.

Note: If the SYSLIB value is not filled in and the load module exists in two SYSLIBs, the module from the set-to zone may have been successfully link-edited to the load module's first SYSLIB. Use the output from the link-edit utility to determine whether the link-edit was successful for the load module's first SYSLIB.

ZAP LINKED INTO LMOD
The zapped module from the set-to zone has been replaced in the cross-zone LMOD.

ZAP NOT LINKED INTO LMOD
The zapped module from the set-to zone has not been replaced in the cross-zone LMOD.

Note: If the SYSLIB value is not filled in and the load module exists in two SYSLIBs, the module from the set-to zone may have been successfully link-edited to the load module's first SYSLIB. Use the output from the link-edit utility to determine whether the link-edit was successful for the load module's first SYSLIB.

MOD DELETED FROM LMOD
The module has been successfully deleted from the cross-zone LMOD.

MOD NOT DELETED FROM LMOD
The module has not been deleted from the cross-zone LMOD.

Note: If the SYSLIB value is not filled in and the load module exists in two SYSLIBs, the module from the set-to zone may have been successfully link-edited to the load module's first SYSLIB. Use the output from the link-edit utility to determine whether the link-edit was successful for the load module's first SYSLIB.

LKED RC
is the return code from the link-edit utility.

REASON
is the reason the cross-zone work was not done.

ABEND
The cross-zone work could not be done because of an abend. You should use a combination of the LINK MODULE command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

Note: If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

ASSEMBLY NOT AVAILABLE
The assembled module was not available, because the assembly was not done. SMP/E determined that no LMODs from the set-to zone needed the assembled module and, therefore, did not do the assembly work. To determine the action you need to take, see the programmer response for message GIM69136W.

CANNOT ALLOCATE CALLLIBS
The cross-zone work for the LMOD could not be done, because the CALLLIBS libraries could not be allocated. Fix the allocation error, and then use the LINK MODULE command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

CANNOT ALLOCATE SIDE DECK LIBRARY
The cross-zone work for the LMOD could not be done, because the side deck library could not be allocated. Fix the allocation error, and then use the LINK MODULE command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

CANNOT ALLOCATE SYSLIB *ddname*
The cross-zone work for the LMOD could not be done, because its SYSLIB could not be allocated. Fix the allocation error, and then use the LINK MODULE command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

CANNOT ALLOCATE UTIN LIBRARY
The cross-zone work for the LMOD could not be done, because a utility input library could not be allocated. Fix the allocation error, and then use the LINK MODULE command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

CROSS-ZONE PROCESSING DEFERRED
The cross-zone work could not be done, because the cross-zone XZLINK subentry was set to DEFER. You should use a combination of the LINK MODULE command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

Note: If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

CSI DATA SET UNAVAILABLE
The cross-zone work could not be done, because the CSI data set containing the cross-zone was not available. You should use a combination of the LINK MODULE command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

Note: If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

ERROR FOUND FOR ZONE
The cross-zone work could not be done, because of an error found while processing the cross-zone. You should use a combination of the LINK MODULE command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

Note: If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

LMOD DOES NOT REFER TO MOD
The module from the set-to zone has an XZLMOD subentry, which means it was once part of the cross-zone LMOD. However, the cross-zone LMOD was not updated to include the updated module, because the LMOD does not have an XZMOD subentry for the module. If the cross-zone LMOD no longer needs the MOD, no action is required. Otherwise, use the LINK MODULE command to link the module into the cross-zone LMOD.

LMOD NOT IN SYSLIB *ddname*
The cross-zone work for the LMOD could not be done, because the LMOD does not exist in its SYSLIB. If the SYSLIB is incorrect, use UCLIN to correct it; then use the LINK MODULE command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

LMOD NOT IN ZONE
The cross-zone work for the LMOD was not done, because the LMOD not longer exists in the cross-zone. No action is required to complete the cross-zone work.

MODULE NO LONGER BEING PROCESSED
The cross-zone work was not done, because the module from the set-to zone is no longer selected to be updated. No action is required to complete the cross-zone work. It is automatically done once the module from the set-to zone has been successfully updated by a subsequent APPLY or RESTORE command.

SEVERE ERROR ENCOUNTERED
The cross-zone work could not be done, because a severe error was encountered. You should use a combination of the LINK MODULE command, the UCLIN command, or the link-edit utility outside of SMP/E to complete the cross-zone work.

Note: If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

SMPLTS PROCESSING FAILED
The cross-zone work for the LMOD was attempted, but the link-edit into the SMPLTS was unsuccessful. Determine the cause for the link-edit failure, and then use the LINK MODULE command or the link-edit utility outside of SMP/E to complete the cross-zone work for the LMOD.

USABLE COPY OF ZAP NOT FOUND
The zap could not be included in the cross-zone LMOD, because a single-CSECT LMOD containing just the zapped module could not be found. Find a usable copy of the module with the zap, and link it into the cross-zone LMOD using either the LINK MODULE command or the link-edit utility outside of SMP/E. If you cannot find a usable copy, use the ZAP utility to update the cross-zone LMOD.

SYSMOD
is ID of the SYSMOD causing the cross-zone work to be scheduled.

Example: Cross-zone summary report for APPLY processing

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                          CROSS-ZONE SUMMARY REPORT FOR APPLY PROCESSING

CROSS            LMOD                                          LKED
ZONE    LMOD     SYSLIB   MODULE   _________ACTION_________     RC      ______________REASON_____________   SYSMOD

XZONE5  XLMOD1   LOADLIBR MOD110   MOD DELETED FROM LMOD        00      N/A                                 APP1003

ZONE10  LMOD8    N/A      MOD110   MOD NOT DELETED FROM LMOD    N/A     ERROR FOUND FOR ZONE                APP1003

ZONE9   LMOD7    N/A      MOD110   MOD NOT DELETED FROM LMOD    N/A     ERROR FOUND FOR ZONE                APP1003
```

Figure 47. Cross-zone summary report: sample report for APPLY

Deleted SYSMOD report

This report is produced at the completion of APPLY and ACCEPT processing when SMP/E has processed a SYSMOD with a ++VER DELETE statement. The report shows the function SYSMODs that have been deleted, and all the service SYSMODs that were applicable to those functions.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                DELETED FUNCTION REPORT FOR xxxxxxxx yyyyy PROCESSING

SYSMOD         DELETED THE
CAUSING THE    FOLLOWING SYSMODS
DELETION       TYPE       SYSMOD


aaaaaaa        bbbbbbbb   ccccccc  ccccccc  ccccccc
               bbbbbbbb   ccccccc  ccccccc  ccccccc

aaaaaaa        bbbbbbbb   ccccccc
```

Figure 48. Deleted SYSMOD report: standard format

These are the fields in the report:

xxxxxxxx
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
is CHECK if the CHECK operand was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

SYSMOD CAUSING THE DELETION
identifies the SYSMOD containing the ++VER DELETE statement.

DELETED THE FOLLOWING SYSMODS
identifies the type and SYSMOD ID of all the SYSMODs that were deleted. The type can be APAR, FUNCTION, PTF, or USERMOD. The SYSMOD IDs are listed from left to right next to the type. Each

PTF, APAR, and USERMOD listed in the TYPE field belongs to the function SYSMOD listed immediately above it.

When the type is `FUNCTION`, the value of the SYSMOD is listed in the following chart:

SYSMOD ID only
The SYSMOD has been installed on the target or distribution libraries and is specified in the DELETE operand list of the ++VER statement for the deleting SYSMOD.

SYSMOD ID followed by FMID(*sysmod_id*)
The SYSMOD is a dependent function that has been implicitly deleted. FMID identifies the associated base function that has been explicitly deleted.

SYSMOD ID followed by `NOT PREVIOUSLY INSTALLED`
The SYSMOD has been specified in the DELETE operand list of the ++VER statement for the deleting SYSMOD, but has not been installed on your target or distribution libraries.

SYSMOD ID followed by PREVIOUSLY DELETED
The SYSMOD has been specified in the DELETE operand list of the ++VER statement for the deleting SYSMOD, but has been previously deleted by another function SYSMOD.

SYSMODs that have been previously deleted may remain as entries on the target zone or distribution zone if they are specified in the SUP operand list either of the deleting function SYSMOD or of another SYSMOD that has been processed concurrently.

Example: Deleted SYSMOD report for APPLY

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                DELETED FUNCTION REPORT FOR APPLY PROCESSING

SYSMOD        DELETED THE
CAUSING THE   FOLLOWING SYSMODS
DELETION      TYPE       SYSMOD


FYZ3000       FUNCTION   FYZ1000
              PTF        UZ00111  UZ00123  UZ00135
              USERMOD    MY11111

              FUNCTION   GYZ1010  FMID (FYZ1000)
              PTF        UZ00112  UZ00124 UZ00136
              USERMOD    MY11112

              FUNCTION   GYZ1020  FMID (GYZ1010)
              PTF        UZ00142  UZ00164
              APAR       AZ12345

              FUNCTION   FYZ2000  NOT PREVIOUSLY INSTALLED
```

Figure 49. Deleted SYSMOD report: sample report for APPLY

Element summary report

This report is produced at the completion of APPLY, ACCEPT, and RESTORE processing to describe the status of the libraries that were updated for each macro, source, module, or data element. Elements are grouped by element type, in the following order and are listed alphabetically under each element type:

1. Macros (++MAC and ++MACUPD statements)
2. Source (++SRC and ++SRCUPD statements)
3. Modules (++MOD and ++ZAP statements)
4. Data elements (++*element* statements)
5. Hierarchical file system elements (++*hfs-element* statements)

The report is not generated when processing for all selected SYSMODs stops before any elements are selected.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                 ELEMENT SUMMARY REPORT FOR xxxxxxxx yyyyy PROCESSING

ELEMENT  ELEMENT  ELEMENT  CURRENT CURRENT DISTLIB SYSLIB  ASSEM    LOAD     LMOD    SYSMOD   SYSMOD
TYPE     NAME     STATUS   FMID    RMID    LIBRARY LIBRARY NAMES    MODULE   SYSLIB  NAME     STATUS


aaaaaaaa bbbbbbbb cccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhhh iiiiiiii jjjjj   kkkkkkk  lllllll
aaaaaaaa bbbbbbbb cccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhhh iiiiiiii jjjjj   kkkkkkk  lllllll
aaaaaaaa bbbbbbbb cccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhhh iiiiiiii jjjjj   kkkkkkk  lllllll
aaaaaaaa bbbbbbbb cccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhhh iiiiiiii jjjjj   kkkkkkk  lllllll
```

Figure 50. Element summary report: standard format

These are the fields in the report:

xxxxxxxx
is the SMP/E command being processed: APPLY, ACCEPT, or RESTORE.

yyyyy
is CHECK if the CHECK operand was specified on the command. Otherwise, this field is blank.

ELEMENT TYPE
is the element type: MAC, MOD, SRC, MUPD, SUPD, S/ZAP, or one of the data element or hierarchical file system element types shown in the "SMP/E Modification Control Statements" chapter in *z/OS SMP/E Reference*.

ELEMENT NAME
is the element name.

ELEMENT STATUS
describes what happened to the element.

APPLIED, ACCEPTED, or RESTORED
The element was processed successfully.

BYPASS
SMP/E detected an error while checking MODIDs. However, because `BYPASS(ID)` was specified, the element was processed.

DELETED
The element was selected and deleted. A element may be deleted if:

- DELETE was specified on the element MCS.
- The supplying SYSMOD was deleted. (A SYSMOD that was applied or accepted specified the supplying SYSMOD on the ++VER DELETE statement.)
- The supplying SYSMOD was restored.

DLIB ERR
The DISTLIB value on the MCS does not match the DISTLIB value in the target or distribution zone element entry. The element was not processed, and the SYSMOD status is NOGO.

INCMPLT
Element processing is incomplete because of some failure. No libraries were updated.

ID ERR
SMP/E detected an error while checking MODIDs. The element was not processed. See the messages in SMPOUT to determine the error.

NOGO
The element was not processed if the SYSMOD status is also NOGO. If the SYSMOD status is ERROR, the element may have been processed. Check the messages in SMPOUT for the status of the library in which the element resides.

NOT SEL
This version of the element was not selected. Following are the reasons an element might not be selected.

- **Multiple versions of the element.** If multiple versions of the same element are being processed concurrently, a superior version may have been chosen from another SYSMOD.

  A module might not be selected if a macro or source caused a higher level of the module to be assembled.

  If none of the versions of an element are selected, a superior version already exists on the target system.
- **FMID mismatch.** Often, when an element is not selected, its FMID did not match the FMID of the element on the target system. The selection and exclusion of elements is discussed in ""Processing" on page 77" on pages "Processing" on page 28 and "Processing" on page 77.
- **No target library.** When an element has no target library (as described in message GIM43401W), the status is NOT SEL. The element is not selected for update to any target libraries.
- **Owning SYSMOD marked NOGO.** An element can be marked NOT SEL if the owning SYSMOD is marked NOGO before processing is finished for that element.

Note: NOT SEL refers to processing in the set-to zone. An element with a status of NOT SEL can be updated in cross-zone LMODs.

SRC SEL
Because the source version of the module was selected, the object version (++MOD) was not processed. For example, when a source or macro is changed, the source may be reassembled to create the updated object module.

CURRENT FMID
is the FMID that appears in the element entry at the end of command processing. It appears only if the element is processed successfully.

CURRENT RMID
is the RMID that appears in the element entry at the end of command processing. It appears only if the element is processed successfully.

DISTLIB LIBRARY
is the ddname of the distribution library containing the element.

SYSLIB LIBRARY
is the ddname of the target library containing the element.

ASSEM NAMES
is a list of SRC or ASSEM modules assembled as a result of a macro or source modification. This field is not present for ACCEPT processing. It is present for RESTORE processing only if the MAC entry contains a GENASM subentry.

LOAD MODULE
is a list of load modules that were link-edited or copied using the module in the ELEMENT NAME field. This field is not present for ACCEPT processing.

Note: For S/ZAP elements, this list shows the load modules that included the module specified on the ++ZAP MCS.

If two operands were used on the IMASPZAP NAME statement to limit the load modules that were updated, this list shows all the load modules that the element is part of. Check the utility completion messages (GIM237*xx*) to determine which load modules were updated.

LMOD SYSLIB
is a list of target libraries that contained the load module in the LOAD MODULE field and that were updated during APPLY or RESTORE processing. This field is set to the DISTLIB value for ACCEPT processing.

Note: SMPLTS appears in this column when a module is linked into a base version of a load module in the SMPLTS data set.

SYSMOD NAME
identifies the SYSMODs that changed the element in the ELEMENT NAME field.

SYSMOD STATUS
is the status of the SYSMOD in the SYSMOD NAME field.

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

DELETED
The SYSMOD was explicitly or implicitly deleted.

ERROR
SYSMOD processing stopped after some target libraries or SMP/E libraries were updated, but before the SYSMOD was completely processed. A SYSMOD is completely processed when all its elements have been processed and all of the SYSMOD's requisites have been completely processed. See SMPOUT to determine the cause of the error.

Note: ERROR does not appear in the SYSMOD status field when the CHECK operand is specified on the command.

EXCLUDED
The SYSMOD was specified on the EXCLUDE operand.

HELD
The SYSMOD was held because one or more of HOLD reason IDs were not resolved.

INCMPLT
SYSMOD processing is incomplete because of some failure. No target libraries were updated.

NOGO
The SYSMOD was not processed before any updates. This can happen when a related SYSMOD has an error. See SMPOUT to determine the cause of the error.

NOGO(E)
SYSMOD processing stopped because a required SYSMOD was excluded.

NOGO(H)
SYSMOD processing stopped because a required SYSMOD was held.

SUPD
The SYSMOD is superseded by one or more SYSMODs being processed. The superseding SYSMODs are shown in the REQUISITE AND SUPBY SYSMODS field in the SYSMOD status report.

Example: APPLY CHECK element summary report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

ELEMENT SUMMARY REPORT FOR APPLY CHECK PROCESSING

ELEMENT  ELEMENT  ELEMENT  CURRENT   CURRENT   DISTLIB  SYSLIB   ASSEM  LOAD    LMOD       SYSMOD   SYSMOD
TYPE     NAME     STATUS   FMID      RMID      LIBRARY  LIBRARY  NAMES  MODULE  SYSLIB     NAME     STATUS

MAC      MAC1     APPLIED  F000000   F000000            MACLIB                             F000000  APPLIED
SRC      SRC1     APPLIED  F000000   F000000            SRCLIB                             F000000  APPLIED
MOD      MOD1     APPLIED  F000000   P000001                                   MOD1    LINKLIB    P000001  APPLIED
                  NOT SEL                                                                F000000  APPLIED
MOD      MOD2     APPLIED  F000000   P000002                                   MOD2    LINKLIB    P000002  APPLIED
                  NOT SEL                                                                F000000  APPLIED
.
CLIST    ZCLIST   APPLIED  F000000   F000000            SXYZLIST                           F000000  APPLIED
PARM     BPARM    APPLIED  F000000   F000000            SXYZPARM                           F000000  APPLIED
HFS      HFSEL1   APPLIED  HFSFUNC   HFSPTF1   APOSIXL1 HFSTGT1                            HFSPTF1  APPLIED
HFS      HFSEL2   APPLIED  HFSFUNC   HFSPTF2   APOSIXL2 HFSTGT2                            HFSPTF2  APPLIED
```

Figure 51. Element summary report: sample report for APPLY CHECK

Exception SYSMOD report

This report is produced at the completion of REPORT ERRSYSMODS processing during which exception SYSMOD checking was done and HOLDERROR reason IDs were not resolved for SYSMODs installed in the specified zone. The report shows the exception SYSMODs that were previously installed, the HOLDERROR reason IDs (APAR numbers) that have made them exception SYSMODs, resolving SYSMODs that have not yet been installed, and the hold class and hold symptoms for each APAR.

The exception SYSMOD reports produced by a given REPORT ERRSYSMODs command are arranged in alphanumerical order by zone name. Each report begins on a new page. The information gathered for each zone is sorted in ascending order, first by FMID, then by SYSMOD name, then APAR number, and finally by resolving SYSMOD.

Format and explanation of data

The Exception SYSMOD Report contains one or two sections for each zone. The first section (shown in Figure 52 on page 476) lists, by FMID, the held, installed SYSMODS and their resolving SYSMODs. If any of the resolving SYSMODs are held, there will be a second section (shown in Figure 53 on page 476), for the zone that lists, by FMID, the resolving SYSMOD for each held SYSMOD. The number of APARs against each installed FMID and the number of resolving SYSMODs against each APAR are saved for later display in the summary section (shown in Figure 54 on page 478).

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE xxxxxxx  DATE: mm/dd/yy - mm/dd/yy

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

aaaaaaa  bbbbbbb  ccccccc  ddddddd eeeeee fff        ggggggg hhhhhhhh
```

Figure 52. Exception SYSMOD report: standard format (first section)

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE xxxxxxx  DATE: mm/dd/yy - mm/dd/yy

FIXES FOR HELD SYSMODS

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

aaaaaaa  bbbbbbb  ccccccc  ddddddd eeeeee fff        ggggggg hhhhhhhh
```

Figure 53. Exception SYSMOD Report: Standard Format (Second Section)

These are the fields in the report:

xxxxxxx
is the name of the zone being reported on. A separate report is provided for each zone specified on the ZONES operand of the REPORT ERRSYSMODS command.

DATE: *vv/vv/vv - ww/ww/ww*
shows the beginning and ending dates of the HOLDDATA used to create this report. The dates appear as *mm/dd/yy*, where *mm* is the month (01 to 12), *dd* is the day (01 to 31), and *yy* is the year (00 to 99).

- *vv/vv/vv* is the beginning date of the HOLDDATA used to create this report.

  If the BEGINDATE operand was specified on the REPORT ERRSYSMODS command, that value appears in this field. Otherwise, the word THROUGH appears.
- *ww/ww/ww* is the ending date of the HOLDDATA used to create this report.

If the ENDDATE operand was specified on the REPORT ERRSYSMODS command, that value appears in this field. Otherwise, the current date or the IPL date (if any) specified on the EXEC statement for GIMSMP is used.

If neither BEGINDATE nor ENDDATE was specified on the REPORT ERRSYSMODS command, no dates appear in this field.

HOLD FMID
is the FMID of the ++HOLD MCS that was received for the HOLDERROR reason ID.

SYSMOD NAME
is the ID of a SYSMOD installed in the specified zone that meets these conditions:

- For target and distribution zones, it is an installed SYSMOD for which ++HOLD statements were received later and whose ERROR reason IDs have not yet been resolved.
- For the global zone, it is a received SYSMOD for which ++HOLD statements with ERROR reason IDs have been received.

If there are no SYSMODs to report on in the zone, you see ***NONE in the SYSMOD NAME field, and the remaining fields are blank.

APAR NUMBER
is a list of one or more HOLDERROR reason IDs (APAR numbers) that caused the installed SYSMOD to become an exception SYSMOD.

RESOLVING SYSMODS NAMES
is the SYSMOD that will fix the problem causing the hold. It will either be the fix for the held SYSMOD or ***NONE, if there is no known fix.

RESOLVING SYSMODS STATUS
can have the following status values:

- GOOD

  The resolving SYSMOD is not held. This does not mean that the SYSMOD has been received. The RESOLVING SYSMOD RECEIVED column will provide that information. GOOD indicates that the resolving SYSMOD has no known problems.

- ERREL

  The resolving SYSMOD is held with a hold class of ERREL. ERREL indicates that, although the resolving SYSMOD is held, the problem that it resolves is more critical.

- HELD

  Resolving SYSMOD is held.

RESOLVING SYSMOD RECEIVED
indicates whether the resolving SYSMOD is received (YES) or not received (NO).

HOLD CLASS
is the hold class specified on the CLASS operand of the ++HOLD MCS that was being installed.

HOLD SYMPTOMS
field contains a description of the problem associated with the held SYSMOD. It may contain 3-character symbols representing various symptoms of the problem, a text description, or a combination of symptom symbols and text. The following symbols may appear. For a complete list of the symbols, see Enhanced HOLDDATA for z/OS (service.software.ibm.com/holdata/390holddata.html).

DAL
Data Loss

FUL
Function Loss

IPL
Requires IPL

PRV
Pervasive Problem

PRF
Performance Problem

Only the first 57 characters of the SYMP field will be displayed in the Exception SYSMOD Report.

Summary section

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT SUMMARY           DATE: mm/dd/yy - mm/dd/yy

ZONE     FMID        TOTAL APARS        TOTAL RESOLVING
                    AGAINST FMID      SYSMODS AGAINST FMID

aaaaaaa  bbbbbbb        nnnnn               nnnnn
```

Figure 54. Exception SYSMOD report: standard format (summary section)

ZONE
shows each requested zone that contained a held SYSMOD.

FMID
list the FMIDs that have APARS against them. If there are no FMIDs with APARs in a requested zone, then this field is set to ***NONE.

TOTAL APARS AGAINST FMID
indicates the total number of APARs that have error holds against an installed FMID.

TOTAL RESOLVING SYSMODS AGAINST FMID
indicates the total number of resolving SYSMODs found against an installed FMID or a SYSMOD of the installed FMID. The number includes all APARs and all PTFs (both received and not received; held and not held) against the FMID or a SYSMOD of the FMID.

Example: Exception SYSMOD report

The following sample report shows an Exception SYSMOD Report for two zones. The first zone has a report section for resolvers for held, installed SYSMODs, (Figure 55 on page 478) and a second section for resolvers for held resolvers (Figure 56 on page 479). Notice that the SYMP and CLASS data is only shown for the first entry for each APAR. Also, the installed, held SYSMOD name is not repeated for each APAR against it. The second zone only contains a section for resolvers to held SYSMODs (Figure 57 on page 479). It does not have a section for resolvers because none of the resolvers are held. Figure 58 on page 479 shows the Summary Section.

These are the additional fields in the Summary Section for Figure 54 on page 478:

```
PAGE 0001  - NOW SET TO GLOBAL ZONE          DATE 04/23/07  TIME 16:08:43  SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE TGT1     DATE: 02/01/07 - 04/23/07

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

HMJ4102  HMJ4102  AN78422  AN78422 GOOD   YES        HIPER   IPL,FAILS WITH E37 ABEND
                          UW31189 HELD   YES
                          UW32001 GOOD   YES
                  AN80332  AN80332 GOOD   YES        HIPER   DAL,PRV,FUL
                          UW37822 GOOD   YES
                  AN80501  UW38922 HELD   YES        HIPER   PRV
HQA5140  HQA5140  AN90012  AN90012 GOOD   YES        HIPER   DAL
                          UW42146 HELD   YES
```

Figure 55. Exception SYSMOD report: sample report (first zone section 1)

```
PAGE 0002  - NOW SET TO GLOBAL ZONE          DATE 04/23/07  TIME 16:08:43  SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE TGT1     DATE: 02/01/07 - 04/23/07

FIXES FOR HELD RESOLVING SYSMODS

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

HMJ4102  UW31189  AN80203  UW32213 GOOD   YES        PE
                          UW36378 HELD   NO
                          UW36402 GOOD   YES
         UW36378  AN81345  AN81345 GOOD   YES        PE
                          UW37011 GOOD   NO
         UW38922  AN81401  UW39013 ERREL  YES        PE
HQA5140  UW42146  AN90025  UW43610 GOOD   NO         PE
```

Figure 56. Exception SYSMOD report: sample report (first zone section 2)

```
PAGE 0003  - NOW SET TO GLOBAL ZONE          DATE 04/23/07  TIME 16:08:43  SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT FOR ZONE DZONE1   DATE: 02/01/07 - 04/23/07

HOLD     SYSMOD   APAR     ---RESOLVING SYSMOD----   HOLD    HOLD
FMID     NAME     NUMBER   NAME    STATUS RECEIVED   CLASS   SYMPTOMS

HBB1200  HBB1200  AY16920  UW17276 ERREL  YES
                  AZ66402  UW16250 GOOD   YES        HIPER   INSTALL AS SOON AS POSSIBLE
```

Figure 57. Exception SYSMOD report: sample report (second zone)

```
PAGE 0004  - NOW SET TO GLOBAL ZONE          DATE 04/23/07  TIME 16:08:43  SMP/E 36.nn SMPRPT OUTPUT

EXCEPTION SYSMOD REPORT SUMMARY           DATE: 02/01/07 - 04/23/07

ZONE     FMID        TOTAL APARS        TOTAL RESOLVING
                    AGAINST FMID      SYSMODS AGAINST FMID

TGT1     HMJ4120            6                  12
         HQA5140            2                   3

DZONE1   HBB1200            2                   2
```

Figure 58. Exception SYSMOD report: sample report (summary section)

File allocation report

This report is produced at the completion of each SMP/E command (except SET and RESETRC) to identify the DD statements used for that command. It shows how each DD statement was obtained and contains information about the DD statement. When filling in the report fields for a data set, SMP/E relies on the system information for that data set, which is stored in the job file control block (JFCB). The report is arranged alphanumerically by ddname. (Regardless of how the SMPTLIB data sets are allocated, they do not appear in the File Allocation report.)

If an error occurs before SMP/E has collected the necessary information about the command's DD statements, the report is not produced.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SMP/E xxxxxxxx FILE ALLOCATION REPORT

ZONE    DDNAME   DDDEFNAM SMPDDNAM TYPE    --------------DATA SET OR PATH-------------- VOLSER UNIT     STATUS

aaaaaaa bbbbbbbb cccccccc dddddddd eeeeeee ffffffffffffffffffffffffffffffffffffffffffff gggggg hhhhhhhh iii
aaaaaaa bbbbbbbb cccccccc dddddddd eeeeeee ffffffffffffffffffffffffffffffffffffffffffff gggggg hhhhhhhh iii
aaaaaaa bbbbbbbb cccccccc dddddddd eeeeeee ffffffffffffffffffffffffffffffffffffffffffff gggggg hhhhhhhh iii
aaaaaaa bbbbbbbb cccccccc dddddddd eeeeeee ffffffffffffffffffffffffffffffffffffffffffff gggggg hhhhhhhh iii
aaaaaaa bbbbbbbb cccccccc dddddddd eeeeeee ffffffffffffffffffffffffffffffffffffffffffff gggggg hhhhhhhh iii
aaaaaaa bbbbbbbb cccccccc dddddddd eeeeeee ffffffffffffffffffffffffffffffffffffffffffff gggggg hhhhhhhh iii
```

Figure 59. File allocation report: standard format

These are the fields in the report:

xxxxxxx
is the SMP/E command being processed.

ZONE

- For the APPLY and RESTORE commands, the ZONE field contains the name of the cross-zone whose DDDEFs have been used for the allocation of the library. An SMP-generated DD is used instead of the DDNAME to keep all currently allocated DDs unique.
- For the LINK MODULE command, the ZONE field contains the name of the FROMZONE or DLIB zone related to the FROMZONE whose DDDEF has been used to allocate the library.

DDNAME
is the ddname of the DD statement used. It can be either a user-specified ddname or one that was generated by SMP/E.

DDDEFNAM
is the name of the DDDEF entry that was used to allocate the DDNAME data set. This field is filled in only if SMP/E dynamically allocated the DD statement. It is blank for:

- Background processing when the JCL supplied a DD statement
- Foreground processing when the data set was preallocated using the TSO ALLOCATE command

The SMPPTS entry in the sample report is an example of a DD statement specified by the user (its DDDEFNAM field is blank).

SMPDDNAM
is filled in only for concatenated DD statements that were dynamically allocated by SMP/E. When a DDDEF entry lists data sets to be concatenated, SMP/E dynamically allocates the individual members and assigns each data set a unique ddname. This name is shown in the SMPDDNAM field. SMP/E then requests that these individual ddnames be concatenated and assigned the ddname indicated in the DDNAME field. The SYSLIB entry in the sample report is an example of concatenated DD statements.

Note:

1. For concatenated DD statements specified in the JCL, only the information from the first concatenated library is displayed.
2. This field is also used to show the data sets in a SYSLIB allocation for a load module, as indicated by the CALLLIBS subentry list in the corresponding LMOD entry. Each ddname in the CALLLIBS list is displayed, along with the information from its corresponding DDDEF entry and its SMP/E-generated ddname. When the CALLLIBS subentry list contains more than one name, SMP/E allocates them as a concatenation and gives the concatenation a generated ddname.

TYPE
is either the type of data set that was allocated or the reason the data set was not allocated. TYPE may also be set when more than one type of information is being generated for a file. In this case, the TYPE field is used to qualify the information for the file.

ERROR
An error occurred when SMP/E tried to dynamically allocate the specified ddname. See the error messages in SMPOUT to determine the cause of the error.

NODDF
There was no DDDEF entry in the current zone; so SMP/E could not dynamically allocate the requested DD statement.

The SMPRPT entry in the sample report is an example of a DD statement that was not found. In this case, because SMP/E could not allocate the SMPRPT DD statement, it had to write all reports to the SMPOUT DD statement.

NTFND
The JCL did not contain the required DD statement. SMP/E tries to dynamically allocate the DD statement using the DDDEF entries.

PATH
The DATA SET OR PATH field of the report is filled in with the path in a UNIX file system to which the file was allocated.

PATHHFS
The DATA SET OR PATH field of the report is filled in with the name of the HFS data set containing the path to which the file was allocated.

The PATHHFS line is generated for all commands in which a path in a UNIX file system has been allocated.

PERM
The DD statement specified a permanent data set.

SYMHFS
The DATA SET OR PATH field of the report is filled in with the name of the HFS data set containing a symbolic link for an element or load module that was updated in the path to which the file was allocated.

Any SYMHFS lines will follow the PATHHFS line for an allocated file.

The SYMHFS lines will only be generated during APPLY, LINK LMOD, LINK MODULE, or RESTORE processing. In APPLY, LINK LMOD, LINK MODULE, or RESTORE, the elements and load modules updated or deleted from libraries allocated to a path will now be tracked. A list of all symbolic links for these elements or load modules will be derived. The physical HFS data set containing the symbolic link is then derived. A list of data sets containing symbolic links is thus determined. Any duplicates are removed from the list. In addition, if the data set name matches the data set name in the PATHHFS line it is also removed from the list. A SYMHFS line is written to the report for all remaining data sets.

SYSIO
The DD statement specified a SYSIN or SYSOUT data set.

The SMPOUT entry in the sample report is an example of a SYSIO data set. The data set name in this sample is the temporary JES2 data set name (shown as `JES2.A.B...`).

TEMP
The DD statement specified a temporary data set.

The entries for SMPWRK1 through SMPWRK6 in the sample report are examples of temporary data sets allocated by SMP/E.

VIO
The DD statement specified is a VIO data set.

The entries for SYSUT1 through SYSUT4 in the sample report are examples of VIO data sets specified by the user; their DDDEFNAM fields are blank.

DATA SET OR PATH
is the name of the data set or path specified in the DDDEF entry or in the JCL.

- For data sets, the full data set name (up to 44 characters) is displayed.
- For concatenated DD statements specified in the JCL, only the information from the first concatenated library is displayed.
- For paths, the full pathname (up to 255 characters) is displayed. The pathname is enclosed by apostrophes. If the pathname plus the enclosing apostrophes is greater than 44 characters, the pathname spans several lines.

VOLSER
identifies the volume specified in the DDDEF entry, the JCL, or the catalog.

UNIT
identifies the unit where the data set resides. This field is filled in if SMP/E dynamically allocated the DD statement and the DDDEF entry contained the unit information. Otherwise, this field is blank.

STATUS
is the status of the data set: NEW, OLD, MOD, or SHR.

Example: File allocation report for APPLY

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SMP/E APPLY    FILE ALLOCATION REPORT

ZONE    DDNAME   DDDEFNAM SMPDDNAM TYPE  --------------DATA SET OR PATH-------------- VOLSER UNIT     STATUS

        BPXLIB1                   PATH    '/etc/bin/'
                                  PATHHFS OMVS.BIN
                                  SYMHFS  OMVS.BIN2
                                  SYMHFS  OMVS.BIN3
        BPXLIB2  BPXLIB2          PATH    '/etc/bin2/'
                                  PATHHFS OMVS.BIN2
        LINKLIB  LINKLIB          PERM  SYS1.LINKLIB                                SYSRES          OLD
        SMPCNTL                   SYSIO JES2.A.B…
        SMPLOG   SMPLOG           PERM  SYS1.SMPLOG                                 SYSRES          OLD
        SMPLTS   SMPLTS           PERM  SYS1.SMPLTS                                 SYSRES          SHR
        SMPMTS   SMPMTS           PERM  SYS1.SMPMTS                                 SYSRES          OLD
        SMPOUT   SMPOUT           SYSIO JES2.A.B…                                                 MOD
        SMPPTS                    PERM  SYS1.SMPPTS                                 SMPVOL          OLD
        SMPRPT   SMPRPT           NODDF
        SMPSCDS  SMPSCDS          PERM  SYS1.SMPSCDS                                SYSRES          OLD
        SMPSTS   SMPSTS           PERM  SYS1.SMPSTS                                 SYSRES          OLD
        SMPWRK1  SMPWRK1          TEMP                                              SCR001          NEW
        SMPWRK2  SMPWRK2          TEMP                                              SCR001          NEW
        SMPWRK3  SMPWRK3          TEMP                                              SCR001          NEW
        SMPWRK4  SMPWRK4          TEMP                                              SCR001          NEW
        SMPWRK6  SMPWRK6          TEMP                                              SCR001          NEW
        SMP00001
                 BPXLIB3  SMP00001 PATH  '/etc/bin/calllib/'
                                  PATHHFS OMVS.BIN3
        SMP00002
                 PLIBASE  SMP00002 PERM  SYS1.PLIBASE                                                SHR
                 CSSLIB   SMP00003 PERM  SYS1.CSSLIB                                                 SHR
        SVCLIB   SVCLIB           PERM  SYS1.SVCLIB                                 SYSRES          OLD
        SYSLIB   SYSLIB
                 SMPMTS   SYSLIB  PERM  SYS1.SMPMTS                                 SYSRES          OLD
                 MACLIB   SMP00004 PERM  SYS1.MACLIB                                SYSRES          OLD
                 AMACLIB  SMP00005 PERM  SYS1.AMACLIB                               DLIB01          OLD
        SYSPRNT  SYSPRNT          SYSIO JES2.A.B…                                                 MOD
        SYSUT1                    VIO                                                             NEW
        SYSUT2                    VIO                                                             NEW
        SYSUT3                    VIO                                                             NEW
        SYSUT4                    VIO                                                             NEW
OPNMVS  SMP00006
OPNMVS           BPXLIB1  SMP00006 PATH  '/etc/bin/cross/'
                                  PATHHFS OMVS.BIN4
XTZONE1 SMP00007
XTZONE1          PLIBASE  SMP00007 PERM  SYS1.XTZONE1.PLIBASE                                      SHR
XTZONE1          CSSLIB   SMP00008 PERM  SYS1.XTZONE1.CSSLIB                                       SHR
```

Figure 60. File allocation report: sample report for APPLY

GENERATE summary report

This report is produced during GENERATE processing to summarize the jobs that have been created.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                GENERATE SUMMARY REPORT

JOBNAME  STEPNAME UTILITY  SYSLIB   MEMBER   TYPE     DISTLIB  MEMBER   FMID

aaaaaaaa bbbbbbbb ccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhh iiiiii
aaaaaaaa bbbbbbbb ccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhh iiiiii
aaaaaaaa bbbbbbbb ccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhh iiiiii
aaaaaaaa bbbbbbbb ccccccc ddddddd eeeeeee fffffff ggggggg hhhhhhh iiiiii
```

Figure 61. GENERATE summary report: standard format

These are the fields in the report:

JOBNAME
is the job name that was generated. The job name is displayed once for the first step of each job. If the job is continued on another page of the report, the job name is repeated at the top of the new page.

STEPNAME
is the step name that was generated. Each step name is displayed only once. If the step is continued on another page of the report, the step name is repeated at the top of the new page.

UTILITY
is the name of the utility to be called. The utility program name is displayed once for each step.

SYSLIB
is the name of the target library that will be updated. The target library ddname is displayed at the start of each step and whenever it changes within that step.

MEMBER
is the member name of the element in the target library. The member name is displayed once for each member in the target library.

TYPE
is the element type of the SYSLIB member.

Note: ASSEM appears only for assembly steps, indicating that the source of the assembly input was the target zone ASSEM entry. If ASSEM is in the TYPE field, no DISTLIB value is displayed.

DISTLIB
is the name of the distribution library from which the element will be obtained. The DISTLIB field is displayed on each line (except for assembly lines for ASSEM entries).

MEMBER
is the member name of the element in the distribution library.

FMID
is the FMID of the function that owns the element.

Examples

The following sample reports are provided:

- "Example 1: No load modules with a SYSLIB allocation" on page 483
- "Example 2: Load modules with a SYSLIB allocation" on page 484

Example 1: No load modules with a SYSLIB allocation

This example shows the kind of information that is normally provided on the GENERATE Summary report.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                GENERATE SUMMARY REPORT

JOBNAME  STEPNAME UTILITY  SYSLIB   MEMBER   TYPE     DISTLIB  MEMBER   FMID

COPYJOB  COPYSTEP IEBCOPY  MACLIB   MAC11    MAC      AMACLIB  MAC11    PAA1100
                           MAC12    MAC      AMACLIB  MAC12    PAA1100
                           SRCLIB   SRC11    SOURCE   ASRCLIB  SRC11    PAA1100
                           SRC12    SOURCE   ASRCLIB  SRC12    PAA1100
                           LPALIB   MOD01    LMOD     AOS12    MOD01    PAA1100
                           MOD02    LMOD     AOS13    MOD02    PAA1100
DEIINST  ISRCLIB  GIMIAP   ISRCLIB  ISRFC01  CLIST    AISRCLIB ISRFC01  PAA1100
         JPNHPLB  GIMIAP   JPNHPLB  HELPK01  HELPJPN  AJPNHPLB HELPK01  PAA1100
                           HELPK02  HELPJPN  AJPNHPLB HELPK02  PAA1100
         FRAMSLB  GIMIAP   FRAMSLB  MSGFRA01 MSGFRA   AFRAMSLB MSGFRA01 PAA1100
         DEUMSLB  GIMIAP   DEUMSLB  MSGDEU01 MSGDEU   ADEUMSLB MSGDEU01 PAA1100
                           MSGDEU02 MSGDEU   ADEUMSLB MSGDEU02 PAA1100
                           MSGDEU03 MSGDEU   ADEUMSLB MSGDEU03 PAA1100
         PARMLIB  GIMIAP   PARMLIB  PARMXX01 PARM     APARMLIB PARMXX01 PAA1100
         ISPPLIB  GIMIAP   ISPPLIB  ISP@PRIM PNLENG   AIPPLIBB ISP@PRIM PAA1100
         ITASPLB  GIMIAP   ITASPLB  SAMPIT07 SAMPITA  AITASPLB SAMPIT07 PAA1100
         ESPTXLB  GIMIAP   ESPTXLB  TXTESP66 TEXTESP  AESPTXLB TXTESP66 PAA1100
LINKLIB  ASSM0001 IEUASM            ASSEM01  ASSEM              ASSEM01  PAA1100
         ASSM0002 IEUASM            ASSEM02  ASSEM              ASSEM02  PAA1100
         ASSM0003 IEUASM            SRCMOD01 SOURCE   SRCDLIB   SRCMOD01 PAA1100
         ASSM0004 IEUASM            SRCMOD02 SOURCE   SRCDLIB   SRCMOD02 PAA1100
         LINK0001 IEWL     LINKLIB  LMOD01   LMOD     AOS12    MOD21    PAA1100
                                                      AOS12    MOD22    PAA1100
                                                      AOS12    MOD23    PAA1100
         LINK0002 IEWL     LINKLIB  LMOD02   LMOD     SMPPUNCH ASSEM01  PAA1100
                                                      SMPPUNCH ASSEM02  PAA1100
                                                      SMPPUNCH SRCMOD01 PAA1100
                                                      SMPPUNCH SRCMOD02 PAA1100
SVCLIB   LINK0001 IEWL     SVCLIB   LMOD50   LMOD     DN554    MOD50    PAA1100
                           LMOD51   LMOD     DN554    MOD51    PAA1100
```

Figure 62. GENERATE summary report: sample report

Example 2: Load modules with a SYSLIB allocation

This example shows the kind of information that is provided on the GENERATE Summary report when load modules have a SYSLIB allocation (the LMOD entry contains a CALLLIBS subentry list). The SMPLTS job is created only when it is necessary to link-edit the base version of such load modules into the SMPLTS data set. The LKSYSLIB job link-edits the executable version of the load modules into the target libraries.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT
OUTPUT

                GENERATE SUMMARY REPORT

JOBNAME  STEPNAME UTILITY  SYSLIB   MEMBER   TYPE     DISTLIB  MEMBER   FMID

COPYJOB  COPYSTEP IEBCOPY  SRCLIB   MSAPL03  SOURCE   DSRCLIB  MSAPL03  FGENSRC
HFSINST  SLIB04   GIMIAP   SLIB04   ELEM1    HFS      DLIB04   ELEM1    FUNC001
                           SLIB04   ELEM10   HFS      DLIB04   ELEM10   FUNC003
DEIINST  SLIB04   GIMIAP   SLIB04   ELEM2    CLIST    DLIB04   ELEM2    FUNC001
                           SLIB04   ELEM20   CLIST    DLIB04   ELEM20   FUNC003
LPALIB   LINK0001 IEWL     LPALIB   FTNLMOD1 LMOD     DLIB1    FTNMOD1  FGENFTN
                                                     DLIB1    FTNMOD2  FGENFTN
SMPLTS   ASSM0001 ASMA90                    ASSEM              MSAPL01
         ASSM0002 ASMA90                    ASSEM              MSAPL02
         LINK0001 IEWL     SMPLTS   LMOD3    LMOD     DLIB1    MOD1     FGEN001
                                                     DLIB1    MOD2     FGEN001
                                                     DLIB1    MOD3     FGEN001
                           SMPLTS   LMOD4    LMOD     DLIB1    MOD1     FGEN001
                                                     DLIB1    MOD2     FGEN001
                                                     DLIB1    MOD3     FGEN001
         LINK0002 IEWL     SMPLTS   LMOD1    LMOD     DLIB1    MOD1     FGEN001
                                                     DLIB1    MOD2     FGEN001
                                                     DLIB1    MOD3     FGEN001
                           SMPLTS   LMOD2    LMOD     DLIB1    MOD1     FGEN001
                                                     DLIB1    MOD2     FGEN001
                                                     DLIB1    MOD3     FGEN001
                           SMPLTS   PLILMOD1 LMOD     DLIB1    PLIMOD1  FGENPL1
                                                     DLIB1    PLIMOD2  FGENPL1
         LINK0003 IEWL     SMPLTS   LAPLSRC1 LMOD     SYSPUNCH MSAPL01
                                                     SYSPUNCH MSAPL02
                                                     DLIB1    MSAPL03  FGENSRC

LKSYSLIB ASSM0001 ASMA90                    ASSEM              MSAPL01
```

```
          ASSM0002 ASMA90                     ASSEM             MSAPL02
          LINK0001 IEWL     CSSLIB   CSSLMOD1 LMOD     DLIB1    CSSMOD1  FGEN001
                                                       DLIB1    CSSMOD2  FGEN001
                                                       DLIB1    CSSMOD3  FGEN001
                            CSSLIB   CSSLMOD2 LMOD     DLIB1    CSSMOD1  FGEN001
                                                       DLIB1    CSSMOD2  FGEN001
          LINK0002 IEWL     FORTLIB  FTNLMOD1 LMOD     DLIB1    FTNMOD1  FGENFTN
                                                       DLIB1    FTNMOD2  FGENFTN
          LINK0003 IEWL     LINKLIB  LMOD1    LMOD     DLIB1    MOD1     FGEN001
                                                       DLIB1    MOD2     FGEN001
                                                       DLIB1    MOD3     FGEN001
                            LINKLIB  LMOD2    LMOD     DLIB1    MOD1     FGEN001
                                                       DLIB1    MOD2     FGEN001
                                                       DLIB1    MOD3     FGEN001

          LINK0004 IEWL     LINKLIB  LAPLSRC1 LMOD     SYSPUNCH MSAPL01
                                                       SYSPUNCH MSAPL02
                                                       DLIB1    MSAPL03  FGENSRC

          LINK0005 IEWL     LPALIB   LMOD3    LMOD     DLIB1    MOD1     FGEN001
                                                       DLIB1    MOD2     FGEN001
                                                       DLIB1    MOD3     FGEN001
                            LPALIB   LMOD4    LMOD     DLIB1    MOD1     FGEN001
                                                       DLIB1    MOD2     FGEN001
                                                       DLIB1    MOD3     FGEN001
```

```
          LINK0006 IEWL     LPALIB   LMOD1    LMOD     DLIB1    MOD1     FGEN001
                                                       DLIB1    MOD2     FGEN001
                                                       DLIB1    MOD3     FGEN001
                            LPALIB   LMOD2    LMOD     DLIB1    MOD1     FGEN001
                                                       DLIB1    MOD2     FGEN001
                                                       DLIB1    MOD3     FGEN001
          LINK0007 IEWL     LPALIB   LAPLSRC1 LMOD     SYSPUNCH MSAPL01
                                                       SYSPUNCH MSAPL02
                                                       DLIB1    MSAPL03  FGENSRC
          LINK0008 IEWL     PLILIB   PLILMOD1 LMOD     DLIB1    PLIMOD1  FGENPL1
                                                       DLIB1    PLIMOD2  FGENPL1
```

GZONEMERGE report

This report is produced during GZONEMERGE processing to show the entries that were merged. If an error occurs and command processing stops, you can use this report to determine how far the merge operation has been completed. This report is arranged by entry type and, within entry type, alphanumerically by entry name. If no entries were merged, the GZONEMERGE report states `NO ENTRIES MERGED. NO APPLICABLE ENTRIES IN FROM GLOBAL ZONE.`

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                      SMP/E GZONEMERGE REPORT

ENTRY         ENTRY              SUBENTRY      SUBENTRY
TYPE          NAME               TYPE          VALUE     ACTION        REASON

aaaaaaaaaaaa  bbbbbbbbbbbbbbbb  cccccccccccc  dddddddd  eeeeeeeeeeee  ffffffff
aaaaaaaaaaaa  bbbbbbbbbbbbbbbb  cccccccccccc  dddddddd  eeeeeeeeeeee  ffffffff
aaaaaaaaaaaa  bbbbbbbbbbbbbbbb  cccccccccccc  dddddddd  eeeeeeeeeeee  ffffffff
aaaaaaaaaaaa  bbbbbbbbbbbbbbbb  cccccccccccc  dddddddd  eeeeeeeeeeee  ffffffff
aaaaaaaaaaaa  bbbbbbbbbbbbbbbb  cccccccccccc  dddddddd  eeeeeeeeeeee  ffffffff
aaaaaaaaaaaa  bbbbbbbbbbbbbbbb  cccccccccccc  dddddddd  eeeeeeeeeeee  ffffffff
```

Figure 63. GZONEMERGE report: standard format

These are the fields in the report:

ENTRY TYPE

is the entry type.

Value

Description

DDDEF
designates a DDDEF entry

FEATURE
designates the FEATURE entry

FMIDSET
designates an FMIDSET entry

GLOBALZONE
designates a GLOBALZONE entry

HOLDDATA(E)
designates a HOLDDATA entry with a hold category of ERROR

HOLDDATA(F)
designates a HOLDDATA entry with a hold category of FIXCAT

HOLDDATA(S)
designates a HOLDDATA entry with a hold category of SYSTEM

HOLDDATA(U)
designates a HOLDDATA entry with a hold category of USER

OPTIONS
designates an OPTIONS entry

ORDER
designates an ORDER entry

PRODUCT
designates the PRODUCT entry

SYSMOD
designates a SYSMOD entry

UTILITY
designates a UTILITY entry

ZONESET
designates a ZONESET entry

ENTRY NAME
is the name of the entry. This field is blank for the GLOBALZONE entry. This field contains the SYSMOD ID specified on the ++HOLD MCS when the entry is HOLDDATA(E), HOLDDATA(F), HOLDDATA(S), or HOLDDATA(U).

SUBENTRY TYPE
is generally blank but contains a subentry designation when the ENTRY TYPE is GLOBALZONE or one of the HOLDDATA types. For a GLOBALZONE entry, the following values may appear:

- FMID
- OPTIONS
- SREL
- ZDESC
- ZONEINDEX

For a HOLDDATA type entry, the only value that can appear is HOLDREASON.

SUBENTRY VALUE
is the value of the field identified by the SUBENTRY TYPE field. This field is generally blank but contains a value for the following subentry types:

- FMID
- OPTIONS
- SREL
- ZDESC

- ZONEINDEX
- HOLDREASON

When HOLDREASON is the subentry type, the subentry value is the *reason-id* specified in the HOLDDATA entry. For other subentry types, the subentry value is one that was found in the GLOBALZONE entry in the originating global zone.

ACTION
describes what SMP/E did with that entry.

MERGED
The specified entry or subentry was in the FROM GLOBAL ZONE but not in the TO GLOBAL ZONE; so SMP/E added it to the TO GLOBAL ZONE.

NOT FOUND
The specified entry was not found in the FROM GLOBAL ZONE.

NOT MERGED
The specified entry or subentry was in both the FROM GLOBAL ZONE and the TO GLOBAL ZONE.

NOT SELECTED
The specified entry was not a valid FMIDSET name.

RENAMED
The specified entry was renamed.

REPLACED
The specified entry was in both the FROM GLOBAL ZONE and the TO GLOBAL ZONE. Because the level of the FEATURE, PRODUCT, or SYSMOD was higher in the originating global zone than was in the destination global zone, the entry was replaced.

REASON
is the reason associated with the action.

The following table depicts the various ACTION values with their associated REASON values and an explanation of each.

Table 27. GZONEMERGE report REASON values

ACTION	REASON	Explanation
MERGED	blank	The entry was merged from the originating global zone into the destination global zone.
NOT FOUND	FORFMID VALUE NOT FOUND IN GZONE FMID LIST	The entry was not found in the originating global zone's GZONE FMID subentry, but was selected previously by SMP/E as a valid FMID on the FORFMID operand.
NOT MERGED	DUPLICATE ENTRY NAME IN TO GLOBAL ZONE	The entry was found in both the originating global zone and the destination global zone.
NOT SELECTED	FORFMID VALUE NOT A VALID FMIDSET NAME	The value specified on the FORFMID operand was not a valid FMIDSET name in the originating global zone.
REPLACED	REWORK LEVEL IS GREATER IN FROM GLOBAL ZONE.	The SYSMOD, FEATURE, or PRODUCT entry was found in both the originating and destination global zones. Because the level of the SYSMOD, FEATURE, or PRODUCT was higher in the originating global zone than was in the destination global zone, the entry was replaced.

The following table depicts the various ACTION values with their associated REASON values when the ENTRY TYPE is GLOBALZONE. All other reason values are covered in Table 27 on page 487 and Table 29 on page 488.

Table 28. GZONEMERGE report REASON values for GZONE entry and subentries

Subentry of GZONE entry	ACTION	REASON	Explanation
FMID	NOT MERGED	DUPLICATE SUBENTRY VALUE IN TO GZONE ENTRY	The subentry value was found in both the originating global zone and the destination global zone.
OPTIONS	NOT MERGED	DUPLICATE SUBENTRY VALUE IN TO GZONE ENTRY	The subentry value was found in both the originating global zone and the destination global zone.
SREL	NOT MERGED	DUPLICATE SUBENTRY VALUE IN TO GZONE ENTRY	The subentry value was found in both the originating global zone and the destination global zone.
ZDESC	NOT MERGED	DUPLICATE SUBENTRY VALUE IN TO GZONE ENTRY	The subentry value was found in both the originating global zone and the destination global zone.
ZONEINDEX	NOT MERGED	DUPLICATE ZONE INDEX NAME WITH DIFFERENT DSN	The subentry value was found in both the originating global zone and the destination global zone. This occurs when the *name* of the zone index is the same in both global zones, but the associated *dsn* is different.

The following table depicts the various ACTION values with their associated REASON values when the ENTRY TYPE is ORDER. All other reason values are covered in Table 27 on page 487 and Table 28 on page 488.

Table 29. GZONEMERGE report REASON values for ORDER entry

ACTION	REASON	Explanation
NOT MERGED	DUPLICATE ORDERID IN TO GLOBAL ZONE	The ORDERID subentry value was found in an existing ORDER entry in the destination zone.
NOT MERGED	UNIQUE ORDER ENTRY NAME CANNOT BE GENERATED	An ORDER entry was found in the destination zone which has the same name as the ORDER entry in the originating zone. SMP/E was not able to generate a unique ORDER entry name in the destination zone.
RENAMED	DUPLICATE ENTRY*origname* IN TO GLOBAL ZONE	An ORDER entry was found in the destination zone which has the same name as the ORDER entry in the originating zone. SMP/E was able to generate a new ORDER entry name to be used in the destination zone.

Examples

The following sample report is provided:

- “Example: Merge global zone entries” on page 488

Example: Merge global zone entries

Assume that you are merging entries into an existing global zone CSI data set. Figure 64 on page 489 is an example of the GZONEMERGE report when merging global zone entries to illustrate the type of information that is contained in the report.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                    SMP/E GZONEMERGE REPORT

ENTRY            ENTRY          SUBENTRY       SUBENTRY
TYPE             NAME           TYPE           VALUE        ACTION       REASON

DDDEF            SMPPTS                                     NOT MERGED   DUPLICATE ENTRY NAME IN TO GLOBAL ZONE
FEATURE          OS3250BA                                   MERGED
FEATURE          OS3260BA                                   NOT MERGED   DUPLICATE ENTRY NAME IN TO GLOBAL ZONE
FEATURE          OS3270BA                                   REPLACED     REWORK LEVEL IS GREATER IN FROM GLOBAL ZONE
FMIDSET          FMIDSET2                                   MERGED
FMIDSET          FMSET1                                     MERGED
GLOBALZONE                      FMID           FUNC001      MERGED
GLOBALZONE                      FMID           FUNC002      MERGED
GLOBALZONE                      FMID           FUNC003      MERGED
GLOBALZONE                      SREL           Z038         NOT MERGED   DUPLICATE SUBENTRY VALUE IN TO GZONE ENTRY
GLOBALZONE                      ZONEINDEX      DLBXZ01      MERGED
GLOBALZONE                      ZONEINDEX      DLBZONE      MERGED
GLOBALZONE                      ZONEINDEX      TGTXZ01      MERGED
GLOBALZONE                      ZONEINDEX      TGTZONE      MERGED
ORDER            ORD00001                                   MERGED
ORDER            ORD00002                                   MERGED
ORDER            ORD00003                                   MERGED
ORDER            ORD00004                                   NOT MERGED   DUPLICATE ORDERID IN TO GLOBAL ZONE
ORDER            ORD00005                                   MERGED
ORDER            ORD00006                                   NOT MERGED   DUPLICATE ORDERID IN TO GLOBAL ZONE
ORDER            ORD00007                                   MERGED
ORDER            ORD00008                                   MERGED
HOLDDATA(E)         UK24680        HOLDREASON     AK34256      MERGED
HOLDDATA(E)      UK13579        HOLDREASON     AK45367      MERGED
HOLDDATA(F)      HBB7730        HOLDREASON     AK18603      MERGED
HOLDDATA(F)      HBB7730        HOLDREASON     AO23456      NOT MERGED   DUPLICATE HOLDDATA ENTRY IN TO GLOBAL ZONE
HOLDDATA(S)      UK24680        HOLDREASON     UCLIN        MERGED
HOLDDATA(S)      UK13579        HOLDREASON     DOC          MERGED
HOLDDATA(S)      UK56478        HOLDREASON     GEN          MERGED
HOLDDATA(U)      UK24680        HOLDREASON     NORESON      MERGED
PRODUCT          5645-001,01.02.00                          NOT MERGED     DUPLICATE ENTRY NAME IN TO GLOBAL ZONE
PRODUCT          5645-001,01.03.00                          MERGED
PRODUCT          5647-A01,02.05.00                          MERGED
SYSMOD           APAR003                                    MERGED
SYSMOD           FUNC001                                    MERGED
SYSMOD           FUNC002                                    MERGED
SYSMOD           FUNC003                                    MERGED
SYSMOD           PTF0001                                    MERGED
SYSMOD           PTF0002                                    NOT MERGED   MCS ENTRY NOT FOUND
SYSMOD           PTF0021                                    NOT MERGED   MCS ENTRY NOT FOUND
SYSMOD           USRMOD3                                    MERGED
ZONESET          ZONEST1                                    MERGED
ZONESET          ZONEST2                                    MERGED
```

Figure 64. GZONEMERGE report: sample report

JCLIN cross-reference report

This report is produced during JCLIN processing to show where to place specific entries in the JCLIN Summary report.

For inline JCLIN processing, the report title line is:

```
JCLIN CROSS REFERENCE REPORT FOR SYSMOD nnnnnnnn
```

where *nnnnnnnn* is the ID of the SYSMOD containing the inline JCLIN. Several of these reports can be produced during APPLY or ACCEPT, one for each SYSMOD with inline JCLIN.

Note: If SMP/E runs out of storage during JCLIN processing, the JCLIN Cross-Reference report may not be produced.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                JCLIN CROSS REFERENCE REPORT

  TYPE     NAME     WHERE USED     TYPE     NAME     WHERE USED

aaaaaaaa bbbbbbbb nnnn nnnn nnnn   aaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn   aaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn   aaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn   aaaaaaaa bbbbbbbb nnnn nnnn nnnn
aaaaaaaa bbbbbbbb nnnn nnnn nnnn   aaaaaaaa bbbbbbbb nnnn nnnn nnnn
```

Figure 65. JCLIN cross-reference report: standard format

These are the fields in the report:

TYPE
is the entry type. The possible entry types, listed in the order in which they appear, are:

- ASSEM
- MAC
- LMOD
- MOD
- SRC
- DLIB

NAME
is the entry name.

WHERE USED
is a list of the pages in the JCLIN Summary report where changes for the entry are noted.

Example: JCLIN cross-reference report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                JCLIN CROSS REFERENCE REPORT

  TYPE     NAME     WHERE USED     TYPE     NAME     WHERE USED

MOD      GIMMPGTA 0001 0003 0004
                  0010 0015 0016
                  0100
MOD      GIMMPGTB 0001 0003 0004
                  0010 0015 0016
                  0100
MOD      GIMMPG01 0001 0003
MOD      GIMMPG02 0001 0003
MOD      GIMMPG03 0001 0003
MOD      GIMMPG04 0001 0003
```

Figure 66. JCLIN cross-reference report: sample report

JCLIN summary report

This report is produced during JCLIN processing to summarize the changes that have been made.

For inline JCLIN processing, the report title line is:

```
JCLIN SUMMARY REPORT FOR SYSMOD nnnnnnnn
```

where *nnnnnnnn* is the ID of the SYSMOD containing the inline JCLIN. Several of these reports can be produced during APPLY or ACCEPT, one for each SYSMOD with inline JCLIN.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                JCLIN SUMMARY REPORT

         JCL INFORMATION                 SMP/E ACTION TAKEN

JOBNAME   STEPNAME  UTILITY   TYPE  MEMBER    TYPE  ACTION   CHANGES MADE

aaaaaaaa  bbbbbbbb  cccccccc  dddd  eeeeeeee  ffff  gggggggg  hhhhhhhhhhhh
aaaaaaaa  bbbbbbbb  cccccccc  dddd  eeeeeeee  ffff  gggggggg  hhhhhhhhhhhh
aaaaaaaa  bbbbbbbb  cccccccc  dddd  eeeeeeee  ffff  gggggggg  hhhhhhhhhhhh
aaaaaaaa  bbbbbbbb  cccccccc  dddd  eeeeeeee  ffff  gggggggg  hhhhhhhhhhhh
```

Figure 67. JCLIN summary report: standard format

These are the fields in the report:

JOBNAME
is the name of the current job being analyzed. The job name is displayed once for the first step of each job. If the job is continued on another page of the report, the job name is repeated at the top of the new page.

STEPNAME
is the name of the current step being analyzed. Each step name is displayed only once. If the step is continued on another page of the report, the step name is repeated at the top of the new page.

UTILITY
is the name of the utility program or catalogued procedure being analyzed. The utility program name is displayed once for each step.

TYPE
is the utility type, which can be:

ASSM
Assembly step

COPY
Copy step

LKED
Link-edit step

UNKN
Ignored step (the utility was not recognized)

UPDT
Update step

MEMBER
is the name of the entry that was modified. The entry name is displayed once for each changed entry.

TYPE
is the entry type: ASSEM, DLIB, LMOD, MAC, MOD, or SRC.

ACTION
is the type of action taken:

ADDED
No entry existed, but one was added.

UPDATED
An existing entry was updated.

DELETED
An existing entry was deleted.

CHANGES MADE
describes the type of changes made to the entry, which may be one or more of the following. *xxxxxxxx* is the value of the field, and *yyyyyyyy* is the new value of the field if it was changed.

- NO UPDATES REQUIRED

  This can appear for all types of entries and JCLIN steps. It indicates that the same JCLIN was previously processed, and because there were no changes in the JCLIN, SMP/E did not need to change any entries as a result of processing the JCLIN. Here are some instances when this might occur:

  – A PTF included the entire product JCLIN, not just the small part that was changed or added. In this case, most of the entries need no updates, because there is no change to the related JCLIN.

  – An APPLY step completed JCLIN processing (so the entries were updated with the JCLIN changes), but a library ran out of space during subsequent processing of that same APPLY step. When the APPLY step is rerun, no JCLIN updates are necessary, because the entries have already been updated.

- ASSEM entry:

  - ASSEMBLER INPUT ADDED
  - ASSEMBLER INPUT REPLACED
- DLIB entry:
  - SYSLIB=*xxxxxxxx*
  - SYSLIB *xxxxxxxx* ADDED
  - SYSLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
- LMOD entry:
  - CALLLIBS ADDED
  - CALLLIBS REPLACED
  - COPY INDICATOR SET
  - COPY INDICATOR DELETED
  - LEPARMS ADDED
  - LEPARMS REPLACED
  - LINK-EDIT INPUT ADDED
  - LINK-EDIT INPUT REPLACED
  - LINK-EDIT INPUT DELETED
  - NOT PROCESSED
  - RC=*rc*
  - RC *rc* ADDED
  - RC *nn* REPLACED BY *rc*
  - SIDEDECKLIB=*xxxxxxxx*
  - SIDEDECKLIB *xxxxxxxx* ADDED
  - SIDEDECKLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
  - SYSLIB=*xxxxxxxx*
  - SYSLIB *xxxxxxxx* ADDED
  - SYSLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
  - UTILITY INPUT ADDED
  - UTILITY INPUT DELETED
  - UTILITY INPUT REPLACED

  Note: When SMPLTS is the output library, JCLIN link-edit steps are skipped.
- MAC entry:
  - DISTLIB=*xxxxxxxx*
  - DISTLIB *xxxxxxxx* ADDED
  - DISTLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
  - GENASM=*xxxxxxxx*
  - GENASM *xxxxxxxx* ADDED
  - MALIAS=*xxxxxxxx*
  - MALIAS *xxxxxxxx* ADDED
  - SYSLIB=*xxxxxxxx*
  - SYSLIB *xxxxxxxx* ADDED
  - SYSLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
- MOD entry:
  - DISTLIB=*xxxxxxxx*

  - DISTLIB *xxxxxxxx* ADDED
  - DISTLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
  - LMOD=*xxxxxxxx*
  - LMOD *xxxxxxxx* ADDED
  - LMOD *xxxxxxxx* DELETED
  - TALIAS=*xxxxxxxx*
  - TALIAS *xxxxxxxx* ADDED
- SRC entry:
  - DISTLIB=*xxxxxxxx*
  - DISTLIB *xxxxxxxx* ADDED
  - DISTLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*
  - SYSLIB=*xxxxxxxx*
  - SYSLIB *xxxxxxxx* ADDED
  - SYSLIB *xxxxxxxx* REPLACED BY *yyyyyyyy*

Example: JCLIN summary report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                JCLIN SUMMARY REPORT

         JCL INFORMATION                 SMP/E ACTION TAKEN

JOBNAME   STEPNAME  UTILITY   TYPE  MEMBER    TYPE  ACTION   CHANGES MADE

JOB001    STEP001   IEWL      LKED  LMOD001   LMOD  ADDED    SYSLIB=LINKLIB,
                                                              LEPARMS ADDED,
                                                              LINK-EDIT INPUT
                                                              ADDED, RC=4
                                  MOD0001   MOD   ADDED    DISTLIB=AOS12,
                                                            LMOD=LMOD001
                                  MOD0002   MOD   ADDED    DISTLIB=AOS12,
                                                            LMOD=LMOD001
                                  LMOD002   LMOD  ADDED    SYSLIB=LINKLIB,
                                                            LEPARMS ADDED,
                                                            LINK-EDIT INPUT
                                                            ADDED
                                  MOD0004   MOD   ADDED    DISTLIB=AOS12,
                                                            LMOD=LMOD001
                                  MOD0005   MOD   ADDED    DISTLIB=AOS12,
                                                            LMOD=LMOD001
          STEP002   IEBCOPY   COPY  MAC001    MACRO ADDED    DISTLIB=AMACLIB,
                                                             SYSLIB=MACLIB
                                  MAC002    MACRO ADDED    DISTLIB=AMACLIB,
                                                             SYSLIB=MACLIB
          STEP003   IDCAMS    UNKN                         NOT PROCESSED
          STEP004   IEUASM    ASSM                         NO UPDATES REQUIRED
JOB002    STEP001   IEUASM    ASSM  ASSEM001  ASSEM UPDATED  ASSEMBLER INPUT
                                                             REPLACED
                                  MAC001    MACRO UPDATED  GENASM ASSEM001
                                                             ADDED
                                  MAC002    MACRO UPDATED  GENASM ASSEM001
                                                             ADDED
          STEP002   IEBCOPY   COPY  AMACLIB   DLIB  ADDED    SYSLIB=MACLIB
          STEP003   IEBCOPY   COPY  MOD0005   MOD   UPDATED  DISTLIB AOS12
                                                              REPLACED BY
                                                              ALINKLIB
*NONAME*  LINK1     IEWBLINK  LKED  GOSLMOD1  LMOD  ADDED    SYSLIB=LOADLIB, LEPARMS ADDED, CALLLIBS ADDED,
                                                              SIDEDECKLIB=SGOSSD
                                  GOSLMOD1  MOD   ADDED    DISTLIB=AGOSDLIB, LMOD=GOSLMOD1
                                  GOSLMOD2  LMOD  ADDED    SYSLIB=LOADLIB, LEPARMS ADDED, CALLLIBS ADDED,
                                                            SIDEDECKLIB=SGOSSD
                                  GOSMOD2   MOD   ADDED    DISTLIB=AGOSDLIB, LMOD=GOSLMOD2
*NONAME*  LINK2     IEWBLINK  LKED  APPLLMOD  LMOD  ADDED    SYSLIB=LOADLIB, LEPARMS ADDED, LINK-EDI INPUT ADDED,
                                                              CALLLIBS ADDED, UTILITY INPUT ADDED
                                  APPLMOD1  MOD   ADDED    DISTLIB=APPLDLIB, LMOD=APPLLMOD
                                  APPLMOD2  MOD   ADDED    DISTLIB=APPLDLIB, LMOD=APPLLMOD
```

Figure 68. JCLIN summary report: sample report

LINK LMODS summary report

This report is produced at the completion of LINK LMODS processing, to identify which load modules were relinked.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SUMMARY REPORT FOR LINK LMODS check PROCESSING

LMOD     LMOD     LMOD   LKED   LMOD      MODULE  MODULE
NAME     SYSLIB   STATUS SEQ#   CALLLIBS  FMID    RMID    ____________MODULES LINKED INTO THE LOAD MODULE_____________

aaaaaaaa bbbbbbbb cccccc dddddd eeeeeeee1 fffffff ggggggg hhhhhhhh iiiiiiii jjjjjjjj kkkkkkkk llllllll mmmmmmmm nnnnnnnn
                                eeeeeeee2
                                eeeeeeeex
```

Figure 69. LINK LMODS summary report: standard format

These are the fields in the report:

check
is CHECK if the CHECK operand was specified. Otherwise, this field is blank.

LMOD NAME
identifies the load module being relinked.

LMOD SYSLIB
is the SYSLIB for the load module being relinked.

Note: SMPLTS appears in this column when the base version of the load module is being relinked in the SMPLTS data set.

LMOD STATUS
indicates whether LINK processing was successful for the load module. This field can be set to any of the following values:

LINKED
the load module was processed successfully. During CHECK processing, this status indicates that no problems were found when processing the load module.

ERROR
link-edit was attempted for the load module but link-edit processing failed. Review the output from the link-edit utility to determine the cause of the failure.

NOGO
link-edit was not attempted for the load module because of an error. Review the messages in the SMPOUT data set to determine the cause of the failure.

LKED SEQ#
is the sequence number assigned to the link-edit step. The sequence number can be used to find the output from the link-edit utility for this load module. This field is set to blanks during CHECK processing.

LMOD CALLLIBS
is the list of DDDEF entries that make up the LMOD entry's CALLLIBS subentry list. This field is set to blanks if the load module does not have any CALLLIBS subentries.

MODULE FMID
is the FMID for the module. If the module was assembled, the FMID is set to blanks.

MODULE RMID
is the RMID for the module. If the module was assembled, the RMID is set to RMIDASM.

MODULES LINKED INTO THE LOAD MODULE
is the names of the modules that were linked into the load module.

Note:

1. SMP/E may not need to rebuild the load module from scratch. For example, if the load module contains XZMOD and CALLLIBS subentries and the base version of the load module exists in the SMPLTS, the load module does not need to be rebuilt. For such a load module, the MODULE FMID, MODULE RMID and MODULE NAME fields in the report would be set to blanks.

2. When going to a new page in the report, the LMOD NAME, LMOD SYSLIB, LMOD STATUS and LKED SEQ# fields are repeated. If the load module exists in two SYSLIBS, the second SYSLIB is also repeated along with its associated LMOD STATUS and LKED SEQ#. The MODULE FMID and MODULE RMID fields are also repeated for the first line of the new page.

Example: LINK LMODS summary report

Figure 70 on page 495 shows that load module LSRLMOD2 was not rebuilt. LSRLMOD2 contained CALLLIBS and its base version existed in the SMPLTS data set.

```
PAGE nnnn  - NOW SET TO
TARGET ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SUMMARY REPORT FOR LINK LMODS PROCESSING

LMOD     LMOD     LMOD   LKED   LMOD      MODULE  MODULE
NAME     SYSLIB   STATUS SEQ#   CALLLIBS  FMID    RMID    ____________MODULES LINKED INTO THE LOAD MODULE________


LSRLMOD1 MIGLIB   LINKED 000001 CSSLIB            RMIDASM ASSEM001 ASEM002
                                PLIBASE   F000001 F000001 LSRMOD01 LSMOD02 LSRMOD03 LSRMOD04 LSRMOD05 LSRMOD06 LSRMOD07
                                                          LSRMOD08 LSMOD09 SRC00005
                                          UW00027 LSRMOD10
                                F000002 UQ00028 QSRMOD01 QSMOD02
LSRLMOD2 OPECLIB  LINKED 000003 CSSLIB
                                PLIBASE
LSRLMOD3 SLIB0001 LINKED 000004 CSSLIB    H000001 H000001 HSRMOD01 HSMOD02 HSRMOD03 HSRMOD04 HSRMOD05 HSRMOD06 HSRMOD07
         SLIB0002 LINKED 000005 PLIBASE                   HSRMOD08 HSMOD09
                                HFSCLIB          UV00039 HSRMOD10
```

Figure 70. LINK LMODS summary report: sample report

LIST summary report

This report is produced during LIST processing to summarize which entries were or were not found in the set-to zone.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

LIST SUMMARY FOR aaaaaaaa

ENTRY-TYPE   ENTRY-NAME          STATUS

bbbbbbbb     ccccccccccccccc   ddddddddddddddddd
bbbbbbbb     ccccccccccccccc   ddddddddddddddddd
bbbbbbbb     ccccccccccccccc   ddddddddddddddddd
bbbbbbbb     ccccccccccccccc   ddddddddddddddddd
```

Figure 71. LIST summary report: standard format

These are the fields in the report:

aaaaaaaa
is the name of the zone containing the entries being listed.

ENTRY-TYPE
is the type of entry SMP/E looked for. If no specific entries of a given type were selected, that entry type is shown only once. If several specific entries of a given type were selected, that entry type is shown for each of the selected entries.

ENTRY-NAME
is the name of an entry that was specified on the LIST command. If no specific entries were selected, this is blank.

STATUS
indicates whether any entries were found.

NOT FOUND
The specified entry or entry type was not found.

STARTS ON PAGE *nnnn*
The specified entry or entry type was found. The LIST output starts on the indicated page in the SMPLIST data set.

EMPTY ZONE – NO ENTRIES FOUND
This may appear for the LIST or LIST ALLZONES command. No entries were found in the specified zone. This is the only line in the report. The entry type and entry name fields are blank.

NO ENTRIES FOUND
This may appear for the LIST FORFMID(*name*) or LIST BACKUP(*sysmod_id*) command. No entries were found for the indicated FORFMID or SYSMOD ID values in the specified zone. This is the only line in the report. The entry type and entry name fields are blank.

Example: LIST summary report

Figure 72 on page 496 shows the LIST Summary report that can accompany the LIST output for the following command:

```
LIST MOD MAC SRC CLIST(CLIST01,CLIST02,CLIST03).
```

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

LIST SUMMARY FOR MVSTGT1

ENTRY-TYPE   ENTRY-NAME          STATUS

CLIST        CLIST01             STARTS ON PAGE 0002
CLIST        CLIST02             NOT FOUND
CLIST        CLIST03             STARTS ON PAGE 0002
MAC                              NOT FOUND
MOD                              STARTS ON PAGE 0001
SRC                              STARTS ON PAGE 0010
```

Figure 72. LIST summary report: sample report

Missing FIXCAT SYSMOD report

The missing FIXCAT SYSMOD report contains two parts. The first part of the report identifies the FIXCAT APARs that are missing as well as the SYSMODs that can resolve the missing APARs. The second part of the report is used to provide additional information about those resolving SYSMODs in the first part that are held for an error.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

MISSING FIXCAT SYSMOD REPORT FOR ZONE nnnnnnn

                              HOLD           MISSING       HELD          ______RESOLVING SYSMOD______
FIX CATEGORY      FMID        CLASS          APAR          SYSMOD        NAME       STATUS   RECEIVED
------------      ----        -----          -------       -------       ----       ------   --------
aaaaaaaaaaaa
                  bbbbbbb     ccc            ddddddd       eeeeeee       fffffff    ggggggg  hhhhhhhh
                  bbbbbbb     ccc            ddddddd       eeeeeee       fffffff    ggggggg  hhhhhhhh
```

Figure 73. Missing FIXCAT SYSMOD report, part 1

These are the fields in the first part of the report:

FIX CATEGORY
is the fix category specified on the CATEGORY operand of the ++HOLD MCS. If multiple fix category values are specified for a particular ++HOLD, that HOLD is reported once for each fix category value that matches any of the fix categories of interest specified by the user. If no missing SYSMODs exist in the zone, this field will contain ***NONE.

FMID
is the FMID of the held SYSMOD.

HOLD CLASS
is the hold class specified on the CLASS operand of the ++HOLD MCS. If the CLASS operand is not specified on the ++HOLD, this field is blank.

MISSING APAR
is the unresolved HOLD reason ID.

HELD SYSMOD
is the ID of a SYSMOD that has an unresolved HOLD condition.

RESOLVING SYSMOD NAME
is the list of SYSMODs that can resolve the HOLD reason ID. If a fixing SYSMOD is identified in the RESOLVER operand on the ++HOLD MCS, then that SYSMOD is included in this list. If other SYSMODs exist in the global zone that supersede the reason ID APAR, then they are also be included in this list. If no SYSMODs are in the list, then this report field will contain the value ***NONE.

RESOLVING SYSMOD STATUS
describes the current state of the SYSMOD that resolves the HOLD.

GOOD
The SYSMOD is not held and has no known problems.

HELD
The SYSMOD is held because one or more error HOLDs exist for it.

RESOLVING SYSMOD RECEIVED
indicates whether the resolving SYSMOD has been received and is found in the global zone or not.

Resolving SYSMODs identified in the missing FIXCAT SYSMOD report might have a status of HELD. This means one or more error HOLDs exists for them. To identify SYSMODs that resolve these error HOLDs, the second part of the missing FIXCAT SYSMOD report is produced. In addition, any of the resolving SYSMODs that are held for an error are also included in the second part of the report. If none of the resolving SYSMODs identified in the first part of the report are held for an error, then this second part of the report is not produced.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

MISSING FIXCAT SYSMOD REPORT FOR ZONE nnnnnnn                              - FIXES FOR HELD RESOLVING SYSMODS

HOLD          HELD                          ______RESOLVING SYSMOD______     HOLD
FMID          SYSMOD      APAR              NAME      STATUS    RECEIVED     CLASS
----          ------      ----              ----      ------    --------     -----
aaaaaaa       bbbbbbb     ccccccc           ddddddd   eeeeeee   ggggggggg    hhhhhhh
                                            ddddddd   eeeeeee   ggggggggg    hhhhhhh
```

Figure 74. Missing FIXCAT SYSMOD report, part 2

These are the fields in the second part of the report:

HOLD FMID
is the FMID of the held SYSMOD.

HELD SYSMOD
is the ID of a SYSMOD that has an unresolved ERROR HOLD condition.

APAR
is the ID of a SYSMOD that has an unresolved HOLD condition.

RESOLVING SYSMOD NAME
is the list of SYSMODs that can resolve the HOLD reason ID. If a fixing SYSMOD is identified in the RESOLVER operand on the ++HOLD MCS, then that SYSMOD is included in this list. If other SYSMODs exist in the global zone that supersede the reason ID APAR, then they are also be included in this list. If no SYSMODs are in the list, then this report field will contain the value ***NONE.

RESOLVING SYSMOD STATUS
describes the current state of the SYSMOD that resolves the HOLD.

GOOD
The SYSMOD is not held and has no known problems.

HELD
The SYSMOD is held because one or more error HOLDs exist for it.

RESOLVING SYSMOD RECEIVED
indicates whether the resolving SYSMOD has been received and is found in the global zone or not.

HOLD CLASS
is the hold class specified on the CLASS operand of the ++HOLD MCS.

Example: Missing FIXCAT SYSMOD report

```
PAGE nnnn  - NOW SET TO GLOBAL ZONE   DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

MISSING FIXCAT SYSMOD REPORT FOR ZONE TGT

                                HOLD       MISSING        HELD          ______RESOLVING SYSMOD______
FIX CATEGORY      FMID          CLASS      APAR           SYSMOD        NAME      STATUS    RECEIVED
------------      -------       -----      -------        -------       -------   ------    --------
IBM.Device.2094.z/OS
                  HBB7730       PSP        AA13644        HBB7730       UA27033   GOOD      NO
                                           AA14941        HBB7730       UA26443   GOOD      NO
                                           AA15968        HBB7730       UA27113   GOOD      YES
                                           AA16005        HBB7730       UA26745   GOOD      NO
                                           AA16529        HBB7730       UA26741   HELD      NO
                                           AA17268        HBB7730       UA27861   GOOD      NO
                  HRM7730       PSP        AA16458        HRM7730       UA28236   GOOD      NO

IBM.Device.zIIP
                  HBB7730       PSP        AA15968        HBB7730       UA27113   GOOD      NO
                                           AA16005        HBB7730       UA26475   GOOD      NO
                  HRM7730       PSP        AA16458        HRM7730       UA28236   GOOD      NO
                                           AA18772        HRM7730       ***NONE

IBM.Function.DataSharing.MVS/ESA
                  HBB7730       PSP        AA13335        HBB7730       UA24231   GOOD      NO
                                           AA16416        HBB7730       UA26375   GOOD      NO
                                           AA16534        HBB7730       UA29059   HELD      NO
                                                                        UA30114   GOOD      YES
                                           AA16640        HBB7730       UA27891   GOOD      NO
                                           AA17188        HBB7730       UA29175   GOOD      NO

IBM.HealthChecker
                  HBB7730       PSP        AA15593        HBB7730       UA28094   GOOD      NO
                                           AA16687        HBB7730       UA27678   GOOD      NO
                  HRF7730       PSP        AA15290        HRF7730       UA26196   GOOD      NO
                                           AA17429        HRF7730       UA28412   GOOD      NO
```

Figure 75. Missing FIXCAT SYSMOD report: sample part 1

```
PAGE nnnn  - NOW SET TO GLOBAL ZONE   DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

MISSING FIXCAT SYSMOD REPORT FOR ZONE TGT                              - FIXES FOR HELD RESOLVING SYSMODS

HOLD           HELD                        ______RESOLVING SYSMOD______          HOLD
FMID           SYSMOD        APAR      NAME     STATUS     RECEIVED        CLASS
-------        -------       -------   -------  -------    --------        -------
HBB7730        UA26741       AA22874   UA34271  GOOD       NO              PE
                                       UA36122  GOOD       YES             PE
               UA29059       AA24875   UA36625  HELD       NO              PE
               UA36625       AA43771   ***NONE                             PE
                             AA45962   ***NONE                             PE
```

Figure 76. Missing FIXCAT SYSMOD report, sample part 2

MOVE/RENAME/DELETE report

This report is produced when SMP/E has processed a SYSMOD containing ++MOVE, ++RENAME, or ++DELETE statements. It can be written for the following commands: ACCEPT, ACCEPT CHECK, APPLY, APPLY CHECK, RESTORE, and RESTORE CHECK.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

             SMP/E MOVE/RENAME/DELETE REPORT FOR xxxxxxx PROCESSING

ELEMENT   ELEMENT   SYSMOD
 TYPE      NAME      NAME    ACTION                                                         COMMENT

aaaaaaaa  bbbbbbbb  ccccccc  ddddddddddddddddddddddddddddddddddddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
aaaaaaaa  bbbbbbbb  ccccccc  ddddddddddddddddddddddddddddddddddddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
aaaaaaaa  bbbbbbbb  ccccccc  ddddddddddddddddddddddddddddddddddddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
aaaaaaaa  bbbbbbbb  ccccccc  ddddddddddddddddddddddddddddddddddddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
aaaaaaaa  bbbbbbbb  ccccccc  ddddddddddddddddddddddddddddddddddddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

Figure 77. MOVE/RENAME/DELETE report: standard format

These are the fields in the report:

xxxxxxx

is the command being processed:

- ACCEPT
- ACCEPT CHECK
- APPLY
- APPLY CHECK
- RESTORE
- RESTORE CHECK

ELEMENT TYPE

is the type of element that was changed: MAC, MOD, SRC, or LMOD.

ELEMENT NAME

is the name of an element that was changed.

SYSMOD NAME

identifies the SYSMOD containing the MCS that changed the element.

ACTION

is the type of change that was made:

ALIAS DELETED FROM SYSLIB *syslib* - *alias*

The load module alias was deleted from the indicated library during APPLY processing.

Note:

1. A long alias spans several lines.
2. An alias is enclosed in apostrophes if it contains a character other than uppercase alphabetic, numeric, national ($, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

ALIAS NOT DELETED FROM SYSLIB *syslib* - *alias*

The load module alias was not deleted from the indicated library during APPLY processing.

Note:

1. A long alias spans several lines.
2. An alias is enclosed in apostrophes if it contains a character other than uppercase alphabetic, numeric, national ($, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

CHANGED DISTLIB FROM *distlib1* TO *distlib2*

The DISTLIB subentry of the element entry was changed during APPLY or RESTORE processing.

CHANGED SYSLIB FROM *syslib1* TO *syslib2*

The SYSLIB subentry of the element entry was changed during ACCEPT processing.

DELETED FROM SYSLIB *syslib1*
The load module was deleted from the indicated library during APPLY processing.

DISTLIB NOT CHANGED FROM *distlib1* TO *distlib2*
The DISTLIB subentry of the element was not changed during APPLY or RESTORE processing.

LMOD ENTRY NOT RENAMED TO *newname*
The indicated LMOD entry was not renamed during ACCEPT processing.

LMOD ENTRY RENAMED TO *newname*
The indicated LMOD entry was renamed during ACCEPT processing.

MOD *aaaaaaaa* IN ZONE *bbbbbbb* UPDATED
The indicated cross-zone module is part of a load module renamed by the ++RENAME statement during APPLY or RESTORE processing. The cross-zone MOD entry has been updated to reflect the new load module name.

MOD *aaaaaaaa* IN ZONE *bbbbbbb* NOT UPDATED
The indicated cross-zone module is part of a load module renamed by the ++RENAME statement during APPLY or RESTORE processing. The cross-zone MOD entry was **not** updated to reflect the new load module name.

MOVED FROM DISTLIB *distlib1* TO *distlib2*
The element was moved to a different distribution library during ACCEPT processing.

MOVED FROM SYSLIB *syslib1* TO *syslib2*
The element was moved to a different target library during APPLY or RESTORE processing.

NOT DELETED FROM SYSLIB *syslib1*
The load module was not deleted from the indicated library during APPLY processing.

NOT MOVED FROM DISTLIB *distlib1* TO *distlib2*
The element was not moved during ACCEPT processing.

NOT MOVED FROM SYSLIB *syslib1* TO *syslib2*
The element or load module was not moved during APPLY or RESTORE processing.

NOT RENAMED TO *newname* IN SYSLIB *syslib1*
The load module was not renamed in the indicated library during APPLY or RESTORE processing.

RENAMED TO *newname* IN SYSLIB *syslib1*
The load module was renamed in the indicated library during APPLY or RESTORE processing.

SIDE DECK ALIAS DELETED FROM LIBRARY *ddname - alias*
The side deck alias was deleted from the indicated library during APPLY processing.

SIDE DECK DELETED FROM LIBRARY *ddname*
The side deck was deleted from the indicated library during APPLY processing.

SIDE DECK NOT RENAMED TO *newname* IN LIBRARY *ddname*
The side deck was not renamed in the indicated library during APPLY or RESTORE processing.

SIDE DECK RENAMED TO *newname* IN LIBRARY *ddname*
The side deck was renamed in the indicated library during APPLY or RESTORE processing.

SYSLIB NOT CHANGED FROM *syslib1* TO *syslib2*
The SYSLIB subentry of the element was not changed during ACCEPT processing.

SYSLIB *syslib1* DELETED FROM LMOD ENTRY
The SYSLIB subentry was deleted from the indicated LMOD entry during ACCEPT processing.

SYSLIB *syslib1* NOT DELETED FROM LMOD ENTRY
The SYSLIB subentry was not deleted from the indicated LMOD entry during ACCEPT processing.

COMMENT
provides additional information about processing:

ALIAS(ES) *alias...*
For ++DELETE processing, the indicated aliases were deleted from the target library. For ++MOVE processing, they were moved to the new target library.

Note:

1. A long alias spans several lines.
2. Multiple aliases are separated by a comma followed by a blank (", ").
3. An alias is enclosed in apostrophes if it contains a character other than uppercase alphabetic, numeric, national ($, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

ALIAS GREATER THAN 8 CHARACTERS
The ++DELETE statement attempted to delete an alias without deleting the associated load module, which resides in a PDSE. Because the alias to be deleted was more than 8 characters long, it could not be deleted.

ALIAS NOT IN SYSLIB
The alias to be deleted was not in the indicated target library.

ALREADY CHANGED
The DISTLIB or SYSLIB subentries for the element or load module has already been changed.

ALREADY DELETED
The indicated load module has already been deleted.

ALREADY MOVED
The element or load module has already been moved.

ALREADY RENAMED
The indicated load module has already been renamed.

ATTEMPTED UPDATE FAILED
Because of errors during cross-zone processing, the attempted cross-zone update was not made.

COPY FAILED
The copy for a load module or element failed during ++MOVE processing.

CSI RESOURCE UNAVAILABLE
Cross-zone processing was not done, because SMP/E could not obtain access to the CSI data set containing the cross-zone.

DEFERRED XZLINK SPECIFIED
Cross-zone processing was not done, because the XZLINK value of the cross-zone is DEFERRED.

FMID MISMATCH
The FMID of the element to be moved must match either the FMID of the SYSMOD containing the ++MOVE MCS, or an FMID specified on the ++MOVE MCS.

HAS ASSOCIATED SYMBOLIC LINKS
++MOVE or ++RENAME processing for a program element has been terminated, because the program element has symbolic links defined in its link-edit control statements.

LIBRARIES NOT AVAILABLE
Libraries needed for ++MOVE, ++RENAME, or ++DELETE processing were not available.

LINK EDIT FAILED
The link-edit for a load module or element failed during ++MOVE processing.

MISSING ALIAS(ES) *alias*...
The indicated aliases were missing from the target or distribution library, even though the element or LMOD entry indicated that they should exist.

Note:

1. A long alias spans several lines.
2. Multiple aliases are separated by a comma followed by a blank (", ").
3. An alias is enclosed in apostrophes if it contains a character other than uppercase alphabetic, numeric, national ($, #, or @), slash, plus, hyphen, period, and ampersand, or it spans several lines.

MOD DOES NOT CONTAIN XZLMOD
The LMOD from the set-to zone has an XZMOD subentry indicating that it contains the cross-zone module. SMP/E cannot update the cross-zone module's XZLMOD record to show that the LMOD

was renamed, because the cross-zone module does **not** contain an XZLMOD subentry for the renamed LMOD.

If the LMOD does contain the cross-zone module, use UCLIN to add an XZLMOD subentry for the renamed LMOD to the cross-zone MOD entry. If the LMOD does not contain the cross-zone module, use UCLIN to remove the XZMOD subentry for the cross-zone module from the LMOD entry.

Note: If you use UCLIN to update the cross-zone connections, make sure the TIEDTO subentries of the zone definition entry are still synchronized.

MOD ENTRIES UPDATED
For ++DELETE processing, the LMOD subentries in the MOD entries have been changed and the LMOD entry has been deleted. For ++RENAME processing, the LMOD subentries in the MOD entries have been changed.

NEW NAME EXISTS
An LMOD entry already exists (as either an LMOD entry or in one of the LMOD's SYSLIBs) for the new name specified on a ++RENAME statement.

NOT FOUND
The definition side deck for the named load module was not found in the side deck library, or no entry was found for the element or load module named.

NOT IN ALL SYSLIBS
The load module was not in all the target libraries indicated in the LMOD entry.

NOT IN CURRENT DISTLIB
The element was not in the distribution library named. The DISTLIB on the MCS does not match the DISTLIB subentry in the element entry.

NOT IN CURRENT SYSLIB
The element or load module was not in the target library named. The SYSLIB on the MCS does not match the SYSLIB subentry in the element or LMOD entry.

NOT INSTALLED
The element or load module named was not installed or being installed.

PREVIOUS ERROR
A previous error prevented SMP/E from updating the cross-zone MOD XZLMOD subentry to indicate that the set-to LMOD was renamed. Use UCLIN to update the XZLMOD subentry for the renamed LMOD.

SYSLIB SUBENTRY NOT FOUND
The SYSLIB subentry of the element was not changed, because the SYSLIB subentry was not found in the element entry. This situation is common during ACCEPT processing for elements that are packaged in a totally copied library.

SYSMOD TERMINATED
The SYSMOD containing the ++MOVE, ++RENAME, or ++DELETE statement previously failed.

XZLMOD SUBENTRY REPLACED
Cross-zone processing has been successful and the MOD entry has been updated with the load module name.

Note: The report is limited to reporting on the first 20 aliases processed for a particular element. For example, suppose a load module having 23 aliases is being moved by a ++MOVE statement. The first 20 aliases moved appear in the MOVE/RENAME/DELETE report.

Example: Report for APPLY processing

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

            SMP/E MOVE/RENAME/DELETE REPORT FOR APPLY PROCESSING

ELEMENT   ELEMENT   SYSMOD
 TYPE      NAME      NAME    ACTION                                                COMMENT


MAC       GIMMPIO   UR06789  MOVED FROM SYSLIB MACLIB TO PTVMACS
                             CHANGED DISTLIB FROM AMACLIB TO APVTMACS
SRC       GIMMPSR   UR01234  MOVED FROM SYSLIB SRCLIB TO NEWSRC
LMOD      GIMSMP    UR04321  MOVED FROM SYSLIB LINKLIB TO PVTLIB                   ALIAS(ES) HMASMP
LMOD      GIMSMP1   UR06420  RENAMED TO GIMTEST IN SYSLIB LINKLIB                 MOD ENTRIES UPDATED
                             RENAMED TO GIMTEST IN SYSLIB PVTLIB
LMOD      GIMSMP2   UR09988  DELETED FROM SYSLIB LINKLIB                           ALIAS(ES) SMPTEST1
                             DELETED FROM SYSLIB PVTLIB                           ALIAS(ES) SMPTEST1
                                                                                  MOD ENTRIES UPDATED
LMOD      GOSLM0D1  UZ12340  RENAMED TO GOSLMOD0 IN SYSLIB LINKLIB                MOD ENTRIES UPDATED
                             RENAMED TO GOSLMOD0 IN SYSLIB SMPLTS
                             SIDE DECK RENAMED TO GOSLMOD0 IN LIBRARY SGOSSD
LMOD      GOSLMOD2  UZ12345  RENAMED TO GOSLMOD3 IN SYSLIB SGOSLOAD                MOD ENTRIES UPDATED
                             RENAMED TO GOSLMOD3 IN SYSLIB SMPLTS
                             SIDE DECK NOT RENAMED TO GOSLMOD3 IN LIBRARY SGOSSD  NOT FOUND
LMOD      GOSLMOD4  UZ12349  NOT RENAMED TO GOSLMOD5 IN SYSLIB SGOSLOAD
                             SIDE DECK NOT RENAMED TO GOSLMOD5 IN LIBRARY SGOSSD  NEWNAME ALREADY EXISTS
LMOD      GOSLMOD7  UZ98765  DELETED FROM SYSLIB SGOSLOAD                         MOD ENTRIES UPDATED
                             DELETED FROM SYSLIB SMPLTS
                             SIDE DECK DELETED FROM LIBRARY SGOSSD
LMOD      GOSLMOD9  UZ98777  ALIAS DELETED FROM SYSLIB SGOSLOAD - ALTNAME
                             SIDE DECK ALIAS DELETED FROM SGOSSD - ALTNAME

LMOD      GXXLMOD   UZ00442  MOVED FROM SYSLIB HFSPTH1 TO HFSPATH2                 ALIAS(ES) '../Friendly_alternate_n
                                                                                    ame1_which_is_64_characters_long
                                                                                    _exactly.', '../The_other_altnam
                                                                                    e_is_SHORTER!!!!'
LMOD      HFSLMOD   UZ00440  ALIAS DELETED FROM SYSLIB HFSPTH1 - '../I_am_a_64_chara
                               cter_name,_wonderfully_friendly!__Think_so????'
                             ALIAS DELETED FROM SYSLIB HFSPATH2 - '../I_too_am_a_lon
                               g_name.__But_not_64_characters.'
LMOD      HFSLMOD   UZ00441  ALIAS NOT DELETED FROM SYSLIB HFSPTH1 - '../lo'       ALIAS NOT IN SYSLIB
                             ALIAS NOT DELETED FROM SYSLIB HFSPATH2 - ../OK       ALIAS NOT IN SYSLIB
LMOD      LMODA     UR08856  NOT MOVED FROM SYSLIB HFS001 TO HFS002               HAS ASSOCIATED SYMBOLIC LINKS
LMOD      LMODC     UR08822  NOT MOVED FROM SYSLIB LINKLIB TO LPALIB             ALREADY MOVED
LMOD      LMODD     UR08544  NOT DELETED FROM SYSLIB LINKLIB                      NOT INSTALLED
                             NOT DELETED FROM SYSLIB PVTLIB                       NOT INSTALLED
LMOD      LMODH     UR06455  ALIAS DELETED FROM SYSLIB LINKLIB - LOTTO
                             ALIAS DELETED FROM SYSLIB PVTLIB - LOTTO
LMOD      LMODM     UR06455  ALIAS NOT DELETED FROM SYSLIB LINKLIB - LOTTO2       ALIAS NOT IN SYSLIB
                             ALIAS NOT DELETED FROM SYSLIB PVTLIB - LOTTO2        ALIAS NOT IN SYSLIB
LMOD      LRMFD1    PREX014  NOT DELETED FROM SYSLIB HFSPTH1                      MISSING ALIAS(ES) '../alternate_na
                                                                                    me1_which_is_friendly_but_long',
                                                                                    '../The_other_altname_is_SHORTER
                                                                                    !!!!'
```

Figure 78. MOVE/RENAME/DELETE report: sample report

RECEIVE exception SYSMOD data report

This report is produced at the completion of RECEIVE processing and shows the ++HOLD and ++RELEASE statements that were processed. This report is arranged by SYSMOD ID, and under each SYSMOD by category (ERROR first, SYSTEM second, and USER third), and under each category by reason ID. The ++HOLD text is displayed exactly as it is entered on the modification control statements.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                RECEIVE ++HOLD/++RELEASE SUMMARY REPORT

   NOTE:  SMD NF   - SYSMOD NOT RELEASED - NOT FOUND IN GLOBAL ZONE
          RSN NF   - SYSMOD NOT RELEASED - NOT HELD FOR THIS REASONID
          INT HLD  - SYSMOD NOT RELEASED - CANNOT RELEASE INTERNAL SYS HOLD

SYSMOD   TYPE     STATUS   REASON   FMID     ++HOLD MCS STATEMENTS

aaaaaaa  bbbb     ccccccc  ddddddd  eeeeeee  fffffffffffffffff
                                             fffffffffffffffff
aaaaaaa  bbbb     ccccccc  ddddddd  eeeeeee  fffffffffffffffff
aaaaaaa  bbbb     ccccccc  ddddddd  eeeeeee  fffffffffffffffff
                                             fffffffffffffffff
aaaaaaa  bbbb     ccccccc  ddddddd  eeeeeee  fffffffffffffffff
```

Figure 79. RECEIVE exception SYSMOD data report: standard format

These are the fields in the report:

SYSMOD
identifies the SYSMOD that is being held or release.

TYPE
is the type of ++HOLD or ++RELEASE:

ERR
Error hold

FIXC
Fix category hold

SYS
System hold

USER
User hold

STATUS
is the status of the ++HOLD or ++RELEASE statement.

EXCLUDED
The SYSMOD was not processed, because it was specified on the EXCLUDE operand of the RECEIVE command.

HELD
The ++HOLD statement was processed.

N/A
The SYSMOD specified in the ++HOLD statement was not applicable to your system, because the FMID specified was not in the GLOBALZONE FMID list.

RELEASED
The reason ID specified on the ++RELEASE statement was removed from the SYSMOD.

SMD NF
SMP/E could not find a SYSMOD entry for the SYSMOD specified on the ++RELEASE statement.

RSN NF
SMP/E could not find the reason ID specified on the ++RELEASE statement.

INT HLD
The reason ID specified was for an internal SYSTEM HOLD, and therefore could not be released. You can resolve this type of HOLD only by using BYPASS(HOLDSYS) on ACCEPT or APPLY.

REASON
identifies the SYSMOD specified in the REASON operand of the ++HOLD or ++RELEASE statement.

FMID
is the FMID specified on the ++HOLD statement. For ++RELEASE statements this field is blank.

++HOLD MCS STATEMENTS
lists the complete text of the ++HOLD statements. For ++RELEASE statements this field is blank.

Examples

The following sample reports are provided:

- “Example 1: Report for internal HOLDDATA” on page 504
- “Example 2: Report for ++HOLD data from SMPHOLD” on page 505

Example 1: Report for internal HOLDDATA

Figure 80 on page 505 is an example of a RECEIVE Exception SYSMOD Summary report.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                RECEIVE ++HOLD/++RELEASE SUMMARY REPORT

   NOTE:  SMD NF   - SYSMOD NOT RELEASED - NOT FOUND IN GLOBAL ZONE
          RSN NF   - SYSMOD NOT RELEASED - NOT HELD FOR THIS REASONID
          INT HLD  - SYSMOD NOT RELEASED - CANNOT RELEASE INTERNAL SYS HOLD

SYSMOD  TYPE  STATUS  REASON   FMID     ++HOLD MCS STATEMENTS

JXY5502 SYS   HELD    FULLGEN  JXY5502  ++HOLD(JXY5502) SYS FMID(JXY5502)
                                          REASON(FULLGEN)
                                          COMMENT(A FULL SYSGEN IS
                                          REQUIRED).
UZ00004 SYS   HELD    UCLIN    JXY5502  ++HOLD(UZ00004) SYS FMID(JXY5502)
                                          REASON(UCLIN)
                                          COMMENT(THIS PTF REQUIRES UCLIN).
```

Figure 80. RECEIVE exception SYSMOD data report: sample report for internal HOLDDATA

Example 2: Report for ++HOLD data from SMPHOLD

Figure 81 on page 505 is an example of the report produced as a result of receiving the following ++HOLD statements from SMPHOLD:

```
++HOLD(UZ00001) SYS FMID(JXY5502)
                REASON(UCLIN) COMMENT
                (UCLIN.
                 DEL MOD(MODA) LMOD(IEANOC01).
                 ENDUCL.).

++HOLD(UZ00002) ERR FMID(JXZ2102)
                REASON(AZ00004) DATE(07001).

++HOLD(AZ99991) USER FMID(JXY5502)
                REASON(EC) DATE(07001)
                COMMENT(NEEDS EC 123456).

++HOLD(MY00001) ERR FMID(HUSR001)
                REASON(MYAPAR1).
```

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                RECEIVE ++HOLD/++RELEASE SUMMARY REPORT

   NOTE:  SMD NF   - SYSMOD NOT RELEASED - NOT FOUND IN GLOBAL ZONE
          RSN NF   - SYSMOD NOT RELEASED - NOT HELD FOR THIS REASONID
          INT HLD  - SYSMOD NOT RELEASED - CANNOT RELEASE INTERNAL SYS HOLD

SYSMOD  TYPE  STATUS  REASON    FMID    ++HOLD MCS STATEMENTS

AZ99991 USER  HELD    EC        JXY5502  ++HOLD(AZ99991) USER FMID(JXY5502)
                                          REASON(EC) DATE(07001)
                                          COMMENT(NEEDS EC 123456).

UZ00001 SYS   HELD    UCLIN     JXY5502  ++HOLD(UZ00001) SYS FMID(JXY5502)
                                          REASON(UCLIN) COMMENT
                                          (UCLIN.
                                           DEL MOD(MODA) LMOD(IEANOC01).
                                           ENDUCL.).

UZ00002 ERR   HELD    AZ00004   JXZ2102  ++HOLD(UZ00002) ERR FMID(JXZ2102)
                                          REASON(AZ00004) DATE(07001).

MY00001 ERR   N/A     MYAPAR1   HUSR001
HBB7730 FIXC  HELD    AA15968   HBB7730  ++HOLD(HBB7730) FMID(HBB7730) REASON(AA15968) FIXCAT DATE(07001)
                                         RESOLVER(UA26443) CLASS(PSP)
                                         CATEGORY(IBM.Device.2094.z/OS
                                                  IBM.Device.zIIP).
HBB7730 FIXC  RELEASED AA15974 HBB7730
```

Figure 81. RECEIVE exception SYSMOD data report: sample report for external HOLDDATA

RECEIVE summary report

This report is produced after completion of HOLDDATA processing at the end of RECEIVE processing to summarize the processing that occurred for SYSMODs and other data in SMPPTFIN, such as source IDs

that might be assigned. If at least one FIXCAT HOLD with one or more assigned source IDs need to be reported, the RECEIVE summary report might be written when only HOLDDATA is processed .

If `SOURCEID` was specified on the RECEIVE command, `FOR SOURCEID=`*source-id* appears on the report heading.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                 RECEIVE SUMMARY REPORT

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SYSMOD   STATUS        TYPE      SOURCEID  FEATURE   STATUS FIELD COMMENTS

aaaaaaa  bbbbbbbbbbbb  cccccccc  dddddddd  eeeeeeee  ffffffffffffffffffffffff
aaaaaaa  bbbbbbbbbbbb  cccccccc  dddddddd  eeeeeeee  ffffffffffffffffffffffff
aaaaaaa  bbbbbbbbbbbb  cccccccc  dddddddd  eeeeeeee  ffffffffffffffffffffffff
aaaaaaa  bbbbbbbbbbbb  cccccccc  dddddddd  eeeeeeee  ffffffffffffffffffffffff
aaaaaaa  bbbbbbbbbbbb  cccccccc  dddddddd  eeeeeeee  ffffffffffffffffffffffff
aaaaaaa  bbbbbbbbbbbb  cccccccc  dddddddd  eeeeeeee  ffffffffffffffffffffffff
```

Figure 82. RECEIVE summary report: standard format

These are the fields in the report:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
identifies whether BYPASS(APPLYCHECK) or BYPASS(ACCEPTCHECK) operands were specified on this RECEIVE command. The value is blank or one of the following values:

- BYPASS(APPLYCHECK) SPECIFIED
- BYPASS(ACCEPTCHECK) SPECIFIED
- BYPASS(APPLYCHECK ACCEPTCHECK) SPECIFIED

If neither BYPASS(APPLYCHECK) nor BYPASS(ACCEPTCHECK) are specified, then this field appears as a blank line.

SYSMOD
identifies the SYSMOD that was processed.

STATUS
describes the outcome of the processing SMP/E did for the SYSMOD.

ASSIGNED
The source ID was assigned to the SYSMOD by the ++ASSIGN statement associated with this SYSMOD or the SOURCEID operand of the RECEIVE command.

ASSOCIATED
A feature was associated to the SYSMOD by a ++FEATURE statement.

NOT ASSIGNED
The SOURCEID operand specified by the ++ASSIGN statement was not assigned to the SYSMOD because the SYSMOD was not already received or the UPGRADE level for the global zone does not support the SOURCEID specified.

NOT RECEIVED
The SYSMOD was not received because of one of the following situations:

- An error occurred during SYSMOD processing.
- APPLYCHECK processing was in effect and the SYSMOD had previously been applied.
- ACCEPTCHECK processing was in effect and the SYSMOD had previously been accepted.

NOT SUPPORTED BY CURRENT UPGRADE LEVEL
Assigning the indicated SOURCEID to a SYSMOD would have made a change to the zone that cannot be processed completely by prior levels of SMP/E. The UPGRADE command must be run to allow SMP/E to make such changes.

RECEIVED
The SYSMOD was successfully received.

RE-RECEIVED
A reworked version of the SYSMOD was successfully received again.

TYPE
is the SYSMOD type: APAR, FUNCTION, PTF, or USERMOD.

SOURCEID
is the source ID specified on the ++ASSIGN statement associated with this SYSMOD, the source ID specified on the SOURCEID operand of the RECEIVE command, or a source ID assigned from a FIXCAT HOLD.

FEATURE
is the feature name specified on the ++FEATURE statement associated with this SYSMOD.

STATUS FIELD COMMENTS
is additional information about the SYSMOD.

ALREADY APPLIED
A Receive Zone Group was defined in the active OPTIONS entry or the ZONEGROUP operand was specified on the RECEIVE command, and the SYSMOD was applied in at least one of the zones in the Receive Zone Group.

ALREADY ACCEPTED
A Receive Zone Group was defined in the active OPTIONS entry, or the ZONEGROUP operand was specified on the RECEIVE command, and the SYSMOD was accepted in at least one of the zones in the Receive Zone Group.

ALREADY APPLIED AND ACCEPTED
A Receive Zone Group was defined in the active OPTIONS entry, or the ZONEGROUP operand was specified on the RECEIVE command, and the SYSMOD was applied and accepted in at least one target and distribution zone in the Receive Zone Group.

NO SYSMODS PROCESSED
No SYSMODs were received.

SYSMODS WITH SPECIFIED FMID(S) NOT FOUND IN SMPPTFIN
Although `RECEIVE FORFMID(`*xxxxxxx*`)` was specified, no SYSMODs were received, because there were no SYSMODs in the SMPPTFIN data set with an FMID matching the FMIDs specified on the FORFMID operand.

ALREADY RECEIVED
The SYSMOD was already in the CSI and PTS data sets and, therefore, was not received.

CONSTRUCTION ERROR
The SYSMOD was not constructed correctly. See SMPOUT for messages describing this error.

I/O ERROR DURING PROCESSING
A severe error occurred on one of the SMP/E data sets during SYSMOD processing. See SMPOUT for messages describing this error.

NO APPLICABLE ++VER
The SYSMOD did not contain a ++VER statement with SREL and FMID values that matched any SREL and FMID values in the global zone definition.

REWORK LEVEL IS NOT GREATER THAN THE VERSION ALREADY RECEIVED
The SYSMOD was not received again, because the REWORK level on its header MCS was not higher than the REWORK level for the previously received version of that SYSMOD.

SELECTED SYSMOD NOT FOUND ON SMPPTFIN
The SYSMOD was specified on the SELECT operand but was not in the input data set.

SELECTED SYSMOD NOT RECEIVED
The SYSMOD was specified on the SELECT operand but was not received because of an error.

SMPTLIB DATA SETS NOT PROCESSED
An error occurred while SMP/E was processing the FROMDS or RELFILE data sets associated with this SYSMOD. See SMPOUT for messages describing this error.

SMPTLIB DATA SET PROCESSING ERROR
An error occurred while SMP/E was processing the FROMDS or RELFILE data sets associated with this SYSMOD. See SMPOUT for messages describing this error.

SMPTLIB LOADED
SMP/E has successfully loaded the RELFILE or FROMDS data sets for this SYSMOD.

STOPPED BY EXIT ROUTINE
The RECEIVE installation exit routine passed SMP/E a return code indicating that the SYSMOD should not be received.

SYNTAX ERROR
SMP/E found a syntax error in at least one of the SYSMOD's message control statements. See SMPOUT for messages that describe this error.

SYSMOD EXCLUDED
The SYSMOD was specified on the EXCLUDE operand of the RECEIVE command.

SYSMOD NOT IN RECEIVE STATUS
The SYSMOD was specified on an ++ASSIGN statement but was not in the SMPPTS data set; therefore, the source ID could not be assigned.

Examples

The following sample reports are provided:

- “Example 1: Successful RECEIVE” on page 508
- “Example 2: Unsuccessful RECEIVE” on page 509

Example 1: Successful RECEIVE

This example shows a RECEIVE command from SMPPTFIN and the resulting RECEIVE Summary report formatted output:

```
SET      BDY(GLOBAL).       /* Set to global zone.      */
RECEIVE  SOURCEID(ABC0706). /* Receive data.            */
```

```
++ASSIGN SOURCEID(PUT0704) TO(UZ00011,UZ00012).
++ASSIGN SOURCEID(XAU3380) TO(UZ00011,UZ00014).
++PTF(UZ00011).
++VER(Z038) FMID(JXY5501).
++MOD(A) DISTLIB(DN554).
 A
++PTF(UZ00012).
++VER(Z038) FMID(F123456).
++MOD(C) DISTLIB(DN554).
 C
++ASSIGN SOURCEID(PUT0705) TO(UZ00013,UZ00014).
++PTF(UZ00013).
++VER(Z038) FMID(JXY5501).
++IF FMID(JXY5502) THEN REQ(UZ00014).
++MOD(D) DISTLIB(DN554).
 D
++PTF(UZ00014).
++VER(Z038) FMID(JXY5501) PRE(UZ00011).
++MOD(A) DISTLIB(AOS12).
 A
++MOD(B) DISTLIB(AOS12).
 B
++ASSIGN SOURCEID(PUT0704) TO(UZ70249).
++ASSIGN SOURCEID(PUT0705) TO(UZ60179).
/*
```

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                RECEIVE SUMMARY REPORT FOR SOURCEID = SAMPLE1

SYSMOD   STATUS        TYPE      SOURCEID  FEATURE   STATUS FIELD COMMENTS


HBB6605  RECEIVED      FUNCTION
         ASSIGNED                OS390R7
         ASSOCIATED                        OS3250BA
HMP1B00  RECEIVED      FUNCTION
         ASSIGNED                OS390R5
         ASSOCIATED                        OS3250BA
PUP1300  RECEIVED      FUNCTION
         ASSOCIATED                        PUP130BA
JUP1301  ASSOCIATED                        PUP130BA

UZ00011  RECEIVED      PTF
         ASSIGNED                PUT0704
         ASSIGNED                XAU3380
UZ00012  RECEIVED      PTF
         ASSIGNED                PUT0704
UZ00013  RECEIVED      PTF
         ASSIGNED                PUT0705
UZ00014  RECEIVED      PTF
         ASSIGNED                PUT0705
         ASSIGNED                XAU3380
UZ60179  ASSIGNED                PUT0705
UZ70249  NOT ASSIGNED            PUT0704            SYSMOD NOT IN RECEIVE STATUS
```

Figure 83. RECEIVE summary report: sample report for successful RECEIVE processing

Example 2: Unsuccessful RECEIVE

Figure 84 on page 509 shows a formatted report output of an unsuccessful RECEIVE run. Assume the following command was entered:

```
SET      BDY(GLOBAL)         /* Set to global zone.      */.
RECEIVE  S(UZ00001,          /* Receive two SYSMODs.     */
           UZ00002)          /*                          */
         SOURCEID(PUT0701)   /* Assign this SOURCEID.    */.
```

Also assume the SMPPTFIN input contained the following code:

```
++FUNCTION(JXY5502).
++VER(Z038).
++HOLD(JXY5502)
      SYSTEM
      FMID(JXY5502)
      REASON(FULLGEN)
      COMMENT(A FULL SYSGEN MUST BE PERFORMED TO
              INSTALL THIS FUNCTION)
      .
++MOD(MOD0003) DISTLIB(DLIB) TXLIB(TXLIB).
++PTF(UZ00001).
++VER(Z038) FMID(JXY5502).
++MOD(MOD0003) DISTLIB(DLIB) TXLIB(TXLIB).
```

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                RECEIVE SUMMARY REPORT FOR SOURCEID=SAMPLE2

SYSMOD   STATUS        TYPE      SOURCEID  FEATURE   STATUS FIELD COMMENTS

UZ00001  RECEIVED      PTF
UZ00002  NOT RECEIVED                              SELECTED SYSMOD NOT FOUND ON SMPPTFIN
```

Figure 84. RECEIVE summary report: sample report with failing SYSMOD

Example 3: RECEIVE summary report with source IDs assigned from FIXCAT HOLDs

Figure 85 on page 510 shows a formatted report output with source IDs assigned from FIXCAT HOLDs.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                RECEIVE SUMMARY REPORT FOR SOURCEID=SAMPLE2

SYSMOD  STATUS     TYPE     SOURCEID       FEATURE    STATUS   FIELD   COMMENTS

UA27033 ASSIGNED            IBM.Device.2094.z/OS
UA27113 ASSIGNED            IBM.Device.2094.z/OS
        ASSIGNED            IBM.Device.zIIP
UA27861 ASSIGNED            IBM.Device.2094.z/OS
UA27891 ASSIGNED            IBM.Function.DataSharing/MVS/ESA
```

Figure 85. RECEIVE summary report: source IDs assigned from FIXCAT HOLDs

Receive product summary report

This report is produced at the completion of RECEIVE processing to summarize the processing that occurred for ++FEATURE and ++PRODUCT MCS. If no ++FEATURE or ++PRODUCT MCS are processed, this report is not generated.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                        RECEIVE PRODUCT SUMMARY REPORT

PRODID   VRM
  FEATURE          STATUS        STATUS COMMENTS        DESCRIPTION

aaaaaaaa vv.rr.mm  ccccccccccc  ddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeee…
  bbbbbbbb         ccccccccccc  ddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeee…
  bbbbbbbb         ccccccccccc  ddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeee…
  bbbbbbbb         ccccccccccc  ddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeee…

aaaaaaaa vv.rr.mm  ccccccccccc  ddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeee…
  bbbbbbbb         ccccccccccc  ddddddddddddddddddddd  eeeeeeeeeeeeeeeeeeeeeeeeeeeeee…
```

Figure 86. Receive product summary report: standard format

These are the fields in the report:

PRODID
identifies the product identifier of the PRODUCT that was processed.

VRM
identifies the version, release, and modification level of the PRODUCT that was processed.

FEATURE
identifies the FEATURE that was processed.

STATUS
describes the outcome of the processing SMP/E did for the ++FEATURE or ++PRODUCT MCS.

NOT RECEIVED
The ++FEATURE or ++PRODUCT MCS was not received because of one of the following situations occurred:

- an error occurred during ++FEATURE or ++PRODUCT MCS processing.
- the ++FEATURE or ++PRODUCT MCS was already received.
- the ++PRODUCT MCS did not contain an SREL value that matched any SREL values in the global zone definition.
- the ++FEATURE MCS did not identify a PRODUCT that was either being received or had already been received.

RECEIVED
The ++FEATURE or ++PRODUCT MCS was successfully received.

RE-RECEIVED
A reworked version of the ++FEATURE or ++PRODUCT MCS was successfully received again.

STATUS FIELD COMMENTS
is additional information about the ++FEATURE or ++PRODUCT MCS.

ALREADY RECEIVED
The FEATURE or PRODUCT was already in the CSI but the REWORK operand was not specified on the incoming MCS. Therefore, the ++FEATURE or ++PRODUCT MCS was not received.

CONSTRUCTION ERROR
The ++FEATURE or ++PRODUCT MCS was not constructed correctly. See SMPOUT for messages describing this error.

I/O ERROR DURING PROCESSING
A severe error occurred on one of the SMP/E data sets during FEATURE or PRODUCT processing. See SMPOUT for messages describing this error.

NO APPLICABLE SREL
The ++PRODUCT MCS did not contain an SREL value that matched any SREL values in the global zone definition.

NO APPLICABLE FMID
The FEATURE MCS did not contain an FMID value that matched any FMID values in the global zone definition.

NO APPLICABLE PRODUCT
The ++FEATURE MCS did not specify a PRODUCT value matching any PRODUCT entries in the global zone.

REWORK LEVEL IS NOT GREATER THAN THE VERSION ALREADY RECEIVED
The ++FEATURE or ++PRODUCT MCS was not received again, because the REWORK level on the MCS was not higher than the REWORK level for the previously received version of the ++FEATURE or ++PRODUCT MCS.

STOPPED BY EXIT ROUTINE
The RECEIVE installation exit routine passed SMP/E a return code indicating that the FEATURE or PRODUCT should not be received.

SYNTAX ERROR
SMP/E found a syntax error in the ++FEATURE or ++PRODUCT MCS. See SMPOUT for messages that describe this error.

DESCRIPTION
the DESCRIPTION value for the PRODUCT or FEATURE that was processed.

Example

This example shows an SMPPTFIN data set containing ++FEATURE and ++PRODUCT MCS, and the resulting Receive Product Summary Report. For this example, assume that the "RECEIVE SYSMODS." command was entered and that SREL P150 does not exist in the global zone.

```
++PRODUCT(5647-A01,2.5.0) DESCRIPTION(OS/390)
  SREL(Z038).
++FEATURE(OS3250BA) DESCRIPTION(OS/390 Base)
  PRODUCT(5647-A01,2.5.0)
  FMID(HBB6605,HMP1B00).
++FEATURE(OS3250DD) DESCRIPTION(OpenEdition DCE User Privacy DES)
  PRODUCT(5647-A01,2.5.0)
  FMID(JMB3125).
++FEATURE(OS3250LD) DESCRIPTION(Language Environment Decryption)
  PRODUCT(5647-A01,2.5.0)
  FMID(JMWL755).
++FEATURE(OS3250PN) DESCRIPTION(PSF NetSpool)
  PRODUCT(5647-A01,2.5.0)
  FMID(HPRF226).

++PRODUCT(4697-949,1.3.0) DESCRIPTION(PUP/E)
  SREL(P150).
++FEATURE(PUP130BA) DESCRIPTION(PUP/E Base)
  PRODUCT(4697-949,1.3.0)
  FMID(PUP1300,JUP1301).

++PRODUCT(5645-007,3.1.0) DESCRIPTION(NETVIEW)
  SREL(Z038).
++FEATURE(NV310BAS) DESCRIPTION(NetView Entitled English (ENU))
  PRODUCT(5645-007,3.1.0)
  FMID(HPZ8100,HPZ8130,JPZ8101).

++PRODUCT(5668-911,2.3.0) DESCRIPTION(PL/I)
  SREL(Z038).
++FEATURE(PLI230LI) DESCRIPTION(OS PL/I Library)
  PRODUCT(5668-911,2.3.0)
  FMID(HDL1202,HHL2302,JDL1212,JDL1214).

++PRODUCT(5695-013,1.3.0) DESCRIPTION(REXX)
  SREL(Z038).
++FEATURE(RX130CMP) DESCRIPTION(REXX Compiler/370)
  PRODUCT(5695-013,1.3.0)
  FMID(HWK0130,JWK0131).
++FEATURE(RX130LIB) DESCRIPTION(REXX Library/370)
  PRODUCT(5695-013,1.3.0)
  FMID(HWJ9130,HWJ9133,JWJ9131).
```

Figure 87. Sample SMPPTFIN containing ++FEATURE and ++PRODUCT MCS

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                       RECEIVE PRODUCT SUMMARY REPORT


PRODID   VRM
  FEATURE          STATUS        STATUS COMMENTS        DESCRIPTION

5647-A01 02.06.00  RECEIVED                             OS/390 Base
  DOS3B260         RECEIVED                             OS/390 Base Opt Feat Installed
  DO3NJ260         NOT RECEIVED  NO APPLICABLE FMID     OS/390 Base JPN Opt Feat Installed
  IONSD260         RECEIVED                             SDSF
  IOS3B260         RECEIVED                             OS/390 Base
  IO3BZ260         NOT RECEIVED  NO APPLICABLE FMID     SDSF JPN
  IO3NJ260         NOT RECEIVED  NO APPLICABLE FMID     OS/390 Base JPN

5648-A25 02.01.00  RECEIVED                             IBM COBOL for OS/390
  ACOB210          RECEIVED                             COBOL Full

5655-DB2 05.01.00  NOT RECEIVED  NO APPLICABLE SREL     IBM Database 2 Server for OS/390
  DB2B510          NOT RECEIVED  NO APPLICABLE PRODUCT  DB2 for OS/390
  DB2N510          NOT RECEIVED  NO APPLICABLE PRODUCT  DB2 NET.DATA
  DB2P510          NOT RECEIVED  NO APPLICABLE PRODUCT  DB2PM

5655-147 01.02.00  NOT RECEIVED  REWORK LEVEL IS NOT    CICS Transaction Server
                                 GREATER THAN THE
                                 VERSION ALREADY
                                 RECEIVED
  CTSB120          RECEIVED                             CICS TS ENU

5695-081 01.03.00  RECEIVED                             CICSPLEX SM
  CIPX130          RECEIVED                             CICSPLEX Capacity ENU
  CITX130          RECEIVED                             CICSPLEX TIERS ENU
```

Figure 88. Receive product summary report: sample report

REJECT summary report

This report is produced at the completion of REJECT processing to summarize the processing that occurred for SYSMODs and other data.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                command SUMMARY REPORT FOR mode MODE continuation


 ZONES CHECKED  NOTE: * - zoneset name   # - zone name in a zoneset

    DLIB    - xddddddddd xddddddddd xddddddddd xddddddddd
              xddddddddd xddddddddd xddddddddd xddddddddd

    TARGET  - xttttttttt xttttttttt xttttttttt xttttttttt
              xttttttttt xttttttttt xttttttttt xttttttttt


NOT REJECTED REPORT

    FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY

        fffffff fffffff fffffff fffffff fffffff fffffff
        fffffff fffffff fffffff fffffff fffffff fffffff

    SYSMODS NOT REJECTED

        SYSMOD  COMMENTS

        sssssss ccccccccccccccccccccccccccccccccc
        sssssss ccccccccccccccccccccccccccccccccc


SUCCESSFULLY REJECTED REPORT

    FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY

        fffffff fffffff fffffff fffffff fffffff fffffff
        fffffff fffffff fffffff fffffff fffffff fffffff

    SYSMODS REJECTED

        sssssss sssssss sssssss sssssss sssssss sssssss
        sssssss sssssss sssssss sssssss sssssss sssssss

    FEATURE ENTRIES REJECTED

        NAME      DESCRIPTION

        ffffffff  ddddddddddddddddddddddddddddddddddddd…
        ffffffff  ddddddddddddddddddddddddddddddddddddd…

    PRODUCT ENTRIES REJECTED

        PRODID   VRM       DESCRIPTION

        pppppppp  vv.rr.mm  ddddddddddddddddddddddddddd…
        pppppppp  vv.rr.mm  ddddddddddddddddddddddddddd…
```

Figure 89. REJECT summary report: standard format

These are the fields at the beginning of the report:

command
REJECT or REJECT CHECK.

mode
is the mode of REJECT processing that was done: MASS, NOFMID, PURGE, or SELECT.

continuation
is additional information appearing on continuation pages of the report to identify the part of the report that is being continued. It appears only when the report spans more than one page. This additional information may be one of the following statements:

- NOT REJECTED REPORT
- FEATURE ENTRIES REJECTED
- PRODUCT ENTRIES REJECTED
- SUCCESSFULLY REJECTED REPORT
- SYSMODS NOT REJECTED

DLIB
lists the distribution zones that were checked during REJECT processing.

For mass or select mode, DLIB shows every distribution zone that was defined by a zone index in the GLOBALZONE entry, unless the distribution zone was specified on the EXCLUDEZONE operand.

For PURGE mode, DLIB shows the distribution zones and ZONESETs that were specified, as well as the distribution zones contained in any specified ZONESETs. Each ZONESET name is preceded with an asterisk (*), and each distribution zone in the ZONESET is preceded with a pound sign (#).

If no distribution zones were checked, NONE appears instead of a zone name.

TARGET
lists the target zones that were checked during REJECT processing.

For mass or select mode, TARGET shows every target zone that was defined by a zone index in the GLOBALZONE entry, unless the target zone was specified on the EXCLUDEZONE operand.

For PURGE mode, TARGET shows the target zones and ZONESETs that were specified, as well as the target zones contained in any specified ZONESETs. Each ZONESET name is preceded with an asterisk (*), and each target zone in the ZONESET is preceded with a pound sign (#).

If there are no entries for this heading, NONE appears instead of a zone name.

These are the fields in the NOT REJECTED section of the report:

FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
lists each FMID specified on the DELETEFMID operand that did not exist in the GLOBALZONE entry. FMIDs are listed only for NOFMID mode processing.

If there are no entries for this heading, NONE appears instead of an FMID.

SYSMOD
lists each SYSMOD that was a candidate to be rejected but then became ineligible. SYSMODs are only listed for select or PURGE mode processing.

If some SYSMODs were successfully rejected and none failed, NONE appears instead of a SYSMOD ID.

COMMENTS
explains why the SYSMOD was not rejected:

REWORK LEVEL IS GREATER THAN THE VERSION IN ZONE *aaaaaaa*
The SYSMOD was not rejected because the REWORK level on its header MCS was greater than the REWORK level for the version currently installed in the specified target or distribution zone.

SYSMOD ACCEPTED IN ZONE *dlibzone*
The SYSMOD was accepted in the specified distribution zone, and BYPASS(ACCEPTCHECK) was not specified with the SELECT operand.

SYSMOD APPLIED IN ZONE *tgtzone*
The SYSMOD was applied in the specified target zone, and BYPASS(APPLYCHECK) was not specified with the SELECT operand.

SYSMOD NOT ACCEPTED IN ZONE *dlibzone*
The SYSMOD was accepted in one or more of the distribution zones indicated on the PURGE operand. However, it was not accepted in the specified zone, where it was applicable.

SYSMOD NOT APPLIED IN ZONE *tgtzone*
The SYSMOD was accepted in one or more of the distribution zones indicated on the PURGE operand. However, it was not applied in this target zone, which was specified on the TARGETZONE operand and in which the SYSMOD was applicable.

SYSMOD NOT FOUND IN THE GLOBAL ZONE
A SYSMOD specified on the SELECT operand was not in the global zone. It might not have been received, or it might have already been rejected.

THERE ARE NO SYSMODS TO REJECT
None of the SYSMODs in the global zone met the criteria specified on the REJECT command. No SYSMODs were rejected. No SYSMODs are listed in the NOT REJECTED section of the report, and NONE appears in the SUCCESSFULLY REJECTED section.

These are the fields in the SUCCESSFULLY REJECTED section of the report:

FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
lists each FMID that was deleted from the GLOBALZONE entry.

For NOFMID mode processing, it shows each FMID that was specified on the DELETEFMID operand and that was successfully deleted.

For mass-mode and select-mode processing, it shows FMIDs of function SYSMODs that were not applied or accepted anywhere and were successfully deleted.

If there are no entries for this heading, NONE appears rather than an FMID.

SYSMODS REJECTED
lists each SYSMOD that was rejected.

If there are no entries for this heading, (NONE) appears rather than a SYSMOD ID.

FEATURE ENTRIES REJECTED
lists the name and description of each FEATURE entry that was rejected during NOFMID mode processing.

If there are no entries for this heading (or for modes other than NOFMID), (NONE) appears rather than a FEATURE entry name and the description is left blank.

PRODUCT ENTRIES REJECTED
lists the product ID, version, release and modification level, and description of each PRODUCT entry that was rejected during NOFMID mode processing.

If there are no entries for this heading (or for modes other than NOFMID), (NONE) appears rather than a product ID and the VRM and description are left blank.

Examples

The following sample reports are provided:

- “Example 1: REJECT summary report for PURGE-mode processing” on page 515
- “Example 2: REJECT summary report for NOFMID-mode processing” on page 516
- “Example 3: REJECT summary for mass-mode processing” on page 517

Example 1: REJECT summary report for PURGE-mode processing

Suppose you defined a ZONESET named MVSSET containing both target and distribution zones. You want to reject all SYSMODs that have been installed in the zones defined in MVSSET; therefore, you entered these commands:

```
SET BDY(GLOBAL).
REJECT PURGE(MVSSET) TZONE(MVSSET).
```

Figure 90 on page 516 is an example of a REJECT Summary report that you might see after running these commands.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
                 REJECT SUMMARY REPORT FOR PURGE MODE


 ZONES CHECKED  NOTE: * - zoneset name   # - zone name in a zoneset
    DLIB    - *MVSSET   #DMVA
    TARGET  - *MVSSET   #TMV1      #TMV2


NOT REJECTED REPORT
    FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
        (NONE)
    SYSMODS NOT REJECTED
        SYSMOD  COMMENTS
        UZ01434 SYSMOD NOT APPLIED IN ZONE TMV2


SUCCESSFULLY REJECTED REPORT
    FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
        (NONE)
    SYSMODS REJECTED
        UZ01245
        UZ01345
        UZ01723
        UZ01803
    FEATURE ENTRIES REJECTED
        NAME      DESCRIPTION
        (NONE)
    PRODUCT ENTRIES REJECTED
        PRODID   VRM       DESCRIPTION
        (NONE)
```

Figure 90. REJECT summary report: sample report for PURGE-mode processing

Example 2: REJECT summary report for NOFMID-mode processing

Assume that you had planned to install function HMX1101, but then decided not to install it. You want to delete the FMID from the global zone and reject any service and HOLDDATA for that deleted FMID; therefore, you entered these commands:

```
SET BDY(GLOBAL).
REJECT DELETEFMID(HMX1101) NOFMID HOLDDATA.
```

Figure 91 on page 517 is an example of a REJECT Summary report that you might see after running these commands.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
                 REJECT SUMMARY REPORT FOR NOFMID MODE


 ZONES CHECKED  NOTE: * - zoneset name   # - zone name in a zoneset
    DLIB    -  (NONE)
    TARGET  -  (NONE)


NOT REJECTED REPORT
    FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
        (NONE)
    SYSMODS NOT REJECTED
        SYSMOD  COMMENTS
        (NONE)


SUCCESSFULLY REJECTED REPORT
    FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
        HMX1101
    SYSMODS REJECTED
        UZ01245
        UR02512
        UR02522
    FEATURE ENTRIES REJECTED
        NAME      DESCRIPTION
        (NONE)
    PRODUCT ENTRIES REJECTED
        PRODID   VRM       DESCRIPTION
        (NONE)
```

Figure 91. REJECT summary report: sample report for NOFMID-Mode processing

Example 3: REJECT summary for mass-mode processing

Assume that you want to reject all SYSMODs that have not been applied or accepted; so you entered these commands:

```
SET BDY(GLOBAL).
REJECT.
```

Figure 92 on page 518 is an example of a REJECT Summary report that you might see after running these commands.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
                 REJECT SUMMARY REPORT FOR MASS MODE


 ZONES CHECKED  NOTE: * - zoneset name   # - zone name in a zoneset
    DLIB    -  DMVA
    TARGET  -  TMV1       TMV2


NOT REJECTED REPORT
    FMIDS NOT DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
        (NONE)
    SYSMODS NOT REJECTED
        SYSMOD  COMMENTS
        (NONE)


SUCCESSFULLY REJECTED REPORT
    FMIDS DELETED FROM THE GLOBAL ZONE FMID SUBENTRY
        (NONE)
    SYSMODS REJECTED
        UZ01245 UZ02512
        UZ01629 UZ02522
        UZ01845
        UZ01852
        UZ01924
    FEATURE ENTRIES REJECTED
        NAME      DESCRIPTION
        (NONE)
    PRODUCT ENTRIES REJECTED
        PRODID   VRM       DESCRIPTION
        (NONE)
```

Figure 92. REJECT summary report: sample report for mass-mode processing

SOURCEID report

This report is produced for REPORT SOURCEID processing to summarize the source IDs found in the specified zones. A separate report is written for each zone specified on the REPORT SOURCEID command.

The format of the report depends on whether the SYSMODIDS operand was specified on the REPORT SOURCEID command.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
          SOURCEID REPORT FOR aaaaaa ZONE bbbbbbb
NOTE: * - INDICATES THE SYSMOD HAS MULTIPLE SOURCEIDS
SOURCEID    ___________________SYSMOD NAMES___________________
cccccccc
            ddddddd   ddddddd   ddddddd   ddddddd   ddddddd
            ddddddd   ddddddd   ddddddd   ddddddd
cccccccc
            ddddddd   ddddddd   ddddddd
```

Figure 93. SOURCEID report: standard format (SYSMODIDS operand specified)

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

          SOURCEID REPORT FOR aaaaaa ZONE bbbbbbb


_________________________SOURCEIDS_____________________________

cccccccc
cccccccc
cccccccc
cccccccc
cccccccc
cccccccc
cccccccc
cccccccc
cccccccc
```

Figure 94. SOURCEID report: standard format (SYSMODIDS operand not specified)

These are the fields in the report:

aaaaaa
is the type of the zone being reported on.

bbbbbbb
is the name of the zone being reported on.

SOURCEID(S)
is a source ID assigned to a SYSMOD in the indicated zone.

If none of the SYSMODs in the zone have been assigned a source ID, ***NONE appears in this field.

SYSMOD NAMES
lists the ID of each SYSMOD to which the indicated source ID is assigned.

The SYSMOD ID only appears when SYSMODIDS was specified on the REPORT SOURCEID command.

Examples

The following sample reports are provided:

- "Example 1: SYSMODIDS operand specified" on page 519
- "Example 2: SYSMODIDS operand not specified" on page 520

Example 1: SYSMODIDS operand specified

Assume that you entered these commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
       ZONES(TGT1)
       SYSMODIDS.
```

Figure 95 on page 519 is an example of the report SMP/E writes:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

          SOURCEID REPORT FOR TARGET ZONE TGT1

NOTE: * - INDICATES THE SYSMOD HAS MULTIPLE SOURCEIDS

SOURCEID    ________________SYSMOD NAMES_________________

PUT0703
            UZ00023*   UZ00024   UZ00035   UZ00037  UZ00039
            UZ00052    UZ00073   UZ00076   UZ00077

PUT0704
            UZ00015    UZ00023*  UZ00044
```

Figure 95. SOURCEID report: sample report (SYSMODIDS operand specified)

Example 2: SYSMODIDS operand not specified

Assume that you entered these commands:

```
SET BDY(GLOBAL).
REPORT SOURCEID
       ZONES(TGT2).
```

Figure 96 on page 520 is an example of the report SMP/E writes:

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

          SOURCEID REPORT FOR TARGET ZONE TGT2


_________________________SOURCEIDS_____________________________

PUT0706
PUT0707
PUT0708
PUT0701
PUT0702
PUT0703
PUT0704
```

Figure 96. SOURCEID report: sample report (SYSMODIDS operand not specified)

Summary of bypassed and unresolved HOLD reason report

This report is produced at the completion of APPLY and ACCEPT processing only if the "Unresolved HOLD Reason Report" or the "Bypassed HOLD Reason Report" is produced. This report can be useful in coding the proper BYPASS operands if you want to bypass the error conditions.

Note: This report does not indicate whether the reason ID caused termination or was bypassed. This information is contained in the "Unresolved HOLD Reason Report" and the "Bypassed HOLD Reason Report".

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SUMMARY OF BYPASSED AND UNRESOLVED HOLD REASON REPORT FOR xxxxxx yyyyy PROCESSING

NOTE: SEE THE HOLDDATA REPORT OF UNRESOLVED HOLD REASON IDS TO DETERMINE HOLDS CAUSING SYSMOD TERMINATIONS.
      SEE THE HOLDDATA REPORT OF BYPASSED HOLD REASON IDS TO DETERMINE HOLDS THAT WERE BYPASSED.

TYPE    REASON ID  REPORT      SYSMODS AFFECTED
------  --------   ----------  ------------------------------------------------------------------------------------

aaaaaa  bbbbbbb    ccccccccc  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd
                              ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd
                              ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd

aaaaaa  bbbbbbb    ccccccccc  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd
                              ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd

aaaaaa  bbbbbbb    ccccccccc  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd
                              ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd  ddddddd
```

Figure 97. Summary of Bypassed and Unresolved HOLD Reason Report: Standard Format

These are the fields in the report:

xxxxxx
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
is CHECK if CHECK was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

TYPE
is the type of ++HOLD which either caused a SYSMOD to terminate or was bypassed.

ERROR
indicates an error hold.

FIXCAT
indicates a fix category hold.

SYSTEM
indicates a system hold.

USER
indicates a user hold.

REASON ID
is a reason ID that affected the processing of the following list of SYSMODs.

REPORT
The report where the type, reason, or sysmod can be found. The report is one of the following types:

BYPASSED
Data found in BYPASSED HOLDDATA Report

UNRESOLVED
Data found in the UNRESOLVED HOLDDATA Report

SYSMODS AFFECTED
is a list of the SYSMODs affected by the unresolved or bypassed HOLD condition.

Example: Summary of bypassed and unresolved HOLD reason report

Figure 98 on page 521 is an illustration of a "Summary of bypassed and unresolved HOLD reason report".

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SUMMARY OF BYPASSED AND UNRESOLVED HOLD REASON REPORT FOR APPLY CHECK PROCESSING

NOTE: SEE THE HOLDDATA REPORT OF UNRESOLVED HOLD REASON IDS TO DETERMINE HOLDS CAUSING SYSMOD TERMINATIONS.
      SEE THE HOLDDATA REPORT OF BYPASSED HOLD REASON IDS TO DETERMINE HOLDS THAT WERE BYPASSED.

TYPE    REASON ID REPORT      SYSMODS AFFECTED
------  --------- ----------  ----------------------------------------------------------------------------------------
ERROR   AK18603   UNRESOLVED  HKDB310

        AK19334   UNRESOLVED  HKDB310

FIXCAT  AO14733   UNRESOLVED  HBB7720

SYSTEM  ACTION    UNRESOLVED  HKDB310

        DOC       BYPASSED    HKDB310

        IPL       BYPASSED    UK12345 UK23456 UK34567 UK45678 UK56789 UK67890 UK78901 UK89012 UK901234

USER    DELAY     UNRESOLVED  UK33778
```

Figure 98. Summary of bypassed and unresolved HOLD reason report: sample report

SYSMOD comparison report

This report is produced for REPORT SYSMODS processing to summarize the SYSMODs found in the input zone, but not found in the comparison zone. If no such SYSMODs are found in the input zone, the SYSMOD Comparison report states `THERE WERE NO SYSMODS TO REPORT.`

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SYSMOD COMPARISON REPORT FOR ztype ZONE zone1 AND ztype ZONE zone2
IN THE GLOBAL ZONE IN CSI globaldataset
MATCHING SREL(S) - aaaa, bbbb, …

FMID___  SYSMOD_  TYPE____  APPLICABLE  RECEIVED  SOURCEIDS

ccccccc  ddddddd  eeeeeeee  fffffffff  ggg       hhhhhhhh
                                                  hhhhhhhh
         ddddddd  eeeeeeee  fffffffff  ggg       hhhhhhhh

ccccccc  ddddddd  eeeeeeee  fffffffff  ggg       hhhhhhhh
         ddddddd  eeeeeeee  fffffffff  ggg       hhhhhhhh
         ddddddd  eeeeeeee  fffffffff  ggg       hhhhhhhh
```

Figure 99. SYSMOD comparison report: standard format

These are the fields in the report:

globaldataset
is the name of the global zone data set that contains *zone2*. This is included in the report when *zone1* and *zone2* are defined in different global zone data sets and the global data set name for *zone2* was specified on the COMPAREDTO operand of the REPORT SYSMODS command.

ztype
is the zone type being reported on. It may be either TARGET or DLIB.

zone1
is the name of the input zone.

zone2
is the name of the zone being compared against the input zone (called the *comparison zone*).

MATCHING SREL(S)
are the SRELs that match in the zones being compared. These are the SRELs defined in the zone definition entries.

FMID
is the FMID of the SYSMOD being reported on (shown in the SYSMOD field).

SYSMOD
is the ID of a SYSMOD installed in the input zone, but not found in the comparison zone.

Note: This field does not show superseded-only SYSMODs, SYSMODs with the DELBY subentry, or SYSMODs in error, because these types of SYSMODs are not individually installed or are not completely installed.

TYPE
is the SYSMOD type of the SYSMOD being reported on: APAR, FUNCTION, PTF, or USERMOD.

APPLICABLE
indicates whether the SYSMOD being reported on is applicable to the comparison zone.

YES
The SYSMOD is applicable to the comparison zone.

NO
The SYSMOD is not applicable to the comparison zone.

REINSTALL
The SYSMOD is a function that is installed in the comparison zone. However, the input zone contains a service update of that function that is applicable to the comparison zone. Therefore, you may want to reinstall the service-updated function in the comparison zone.

UNKNOWN
SMP/E cannot determine whether the SYSMOD is applicable to the comparison zone. This can happen if the input zone supports at least one SREL that is not supported by the comparison zone, and there is no entry for the SYSMOD or its FMID in the global zone.

To determine whether the SYSMOD is applicable to the comparison zone, you need to check the program directory or installation manual for the FMID to find out its SREL. If that SREL is supported by the comparison zone, the SYSMOD may be applicable. For more details, see the description of REPORT SYSMODS processing in "Processing" on page 333.

If the SYSMOD is applicable, receive it again, and change the SMPPUNCH output so that the SYSMOD is no longer commented out. If you could not determine whether the SYSMOD is applicable, you can still use this approach. Messages issued during APPLY CHECK or ACCEPT CHECK processing indicate whether the SYSMOD is applicable.

RECEIVED
indicates whether the SYSMOD being reported on has been received and is available in the global zone. Either YES or NO may appear in this field.

SOURCEIDS
is a list of the source IDs associated with the SYSMOD in the input zone. If there are no source IDs associated with the SYSMOD, this field is blank.

Note: If an applicable SYSMOD is not received, the source IDs may help you determine where to obtain the SYSMOD.

Example: SYSMOD comparison report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SYSMOD COMPARISON REPORT FOR TARGET ZONE TZONE1 AND TARGET ZONE TZONE2
IN THE GLOBAL ZONE IN CSI SYS1.PRODUCT.VSAM.CSI

MATCHING SREL(S) - Z038

FMID___  SYSMOD_  TYPE____  APPLICABLE  RECEIVED  SOURCEIDS

HAA1202  HAA1202  FUNCTION  YES         YES
         UZ00011  PTF       YES         NO        PUT0706
         UZ00012  PTF       YES         YES       PUT0707
         AZ00013  APAR      YES         NO        RETAIN
         TZ00001  USERMOD   YES         YES
HBB1202  UZ00002  PTF       YES         YES       PUT0707
                                                   PDO0001
JBB1222  UZ00021  PTF       NO          NO        PUT0706
         UZ00022  PTF       NO          YES       PUT0707
```

Figure 100. SYSMOD comparison report: samplereport

SYSMOD comparison HOLDDATA report

This report is produced for REPORT SYSMODS processing to show the SYSTEM and USER HOLDDATA that must be resolved before the SYSMODs appearing in the SYSMOD Comparison Report can be installed. If no such HOLDDATA is found in the input zone, the SYSMOD Comparison HOLDDATA report states THERE WAS NO HOLDDATA TO REPORT.

The SYSMOD Comparison HOLDDATA Report will be directed to SMPHRPT. If SMPHRPT is not allocated, the SYSMOD Comparison HOLDDATA Report will be directed to SMPRPT. If neither SMPHRPT nor SMPRPT are allocated, the SYSMOD Comparison HOLDDATA Report will be directed to SMPOUT.

Format and explanation of data

```
PAGE nnnn - NOW SET TO zzzzzz ZONE nnnnnnnn DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPHRPT OUTPUT

SYSMOD COMPARISON HOLDDATA REPORT FOR ztype ZONE zone1 AND ztype ZONE zone2
IN THE GLOBAL ZONE IN CSI csidataset

TYPE     REASON ID   FMID     SYSMOD   ++HOLD DATA
------   ---------   -------  -------
--------------------------------------------------------------------------------
aaaaaa   bbbbbbb     ccccccc  ddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

                                ddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                        ccccccc  ddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee


eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
         bbbbbbb     ccccccc  ddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                        ccccccc ddddddd
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

Figure 101. SYSMOD comparison HOLDDATA report: standard format

These are the fields in the report:

ztype
is the zone type being reported on. It may be either TARGET or DLIB.

zone1
is the name of the input zone.

zone2
is the name of the comparison zone.

csidataset
the global zone CSI data set name for the COMPAREDTO zone. This data set name is included in the report only if it differs from the global zone CSI data set name for the input zone.

TYPE
is the type of HOLD for the following reason ID.

SYSTEM
indicates a system hold

USER
indicates a user hold

REASONID
reason ID that identifies why the SYSMOD should be held

FMID
is the FMID of the held SYSMOD being reported (shown in the SYSMOD field)

SYSMOD
is the ID identified in the ++HOLD statement as the held SYSMOD

++HOLD DATA
this is one of the following selections:

- the ++HOLD MCS that was received for the HOLD reason ID. This data can span several lines and is provided for each reason ID, or
- an indication that the ++MCS has been suppressed for the reason IDs specified in the Suppress HOLDDATA (SUPPHOLD) subentry of the OPTIONS entry.

Example: SYSMOD Comparison HOLDDATA report

Figure 102 on page 525 is an example of the SYSMOD Comparison HOLDDATA report.

```
PAGE 2 - NOW SET TO GLOBAL ZONE        DATE 04/07/08    TIME 15:04:07    SMP/E 35.00   SMPHRPT   OUTPUT

SYSMOD COMPARISON HOLDDATA REPORT FOR TARGET ZONE OS390 AND TARGET ZONE OS390A
IN THE GLOBAL ZONE IN CSI SYS2.PRODUCT.VSAM.CSI

TYPE      REASON ID   FMID      SYSMOD    ++HOLD DATA
------    ---------   -------   -------
--------------------------------------------------------------------------------
SYSTEM    DOC         HRSL380   UA36586   ++ HOLD(UA36586) SYS FMID(HRSL380) REASON(ACTION) DATE(07241)
                                              COMMENT
                                               (With this update to the LDAP server it is
                                                recommended, but not required, that the GDBM database
                                                be updated for row level locking.

                                                Detailed instructions on creating and altering GDBM for this
                                                update can be found in the documentation update included with
                                                this PTF.).

          IPL         HBB7740   UA35261   ++ HOLD(UA35261) SYS FMID(HBB7740) REASON(IPL) DATE(07215)
                                              COMMENT(Reason for the IPL hold).

                                UA36063   ++ HOLD(UA36063) SYS FMID(HBB7740) REASON(IPL) DATE(08031)
                                              COMMENT(Reason for the IPL hold).

                                UA36447   ++ HOLD(UA36447) SYS FMID(HBB7740) REASON(IPL) DATE(08066)
                                              COMMENT(Reason for the IPL hold).

                    HDZ1190     UA36048   ++ HOLD(UA36048) SYS FMID(HDZ1190) REASON(IPL) DATE(08031)
                                              COMMENT(Reason for the IPL hold).

                    HJE7740     UA36911   ++ HOLD(UA36911) SYS FMID(HJE7740) REASON(IPL) DATE(08044)
                                              COMMENT(Reason for the IPL hold).

                                UA37177   ++ HOLD(UA37177) SYS FMID(HJE7740) REASON(IPL) DATE(08021)
                                              COMMENT(Reason for the IPL hold).

           ACTION   HBB2102     UZ00030   ++ HOLD(UZ00030) SYS FMID(HBB2102) REASON(ACTION) DATE(06215)
                                              COMMENT(Reason for the ACTION hold).

           RESTART  HBB2102     UZ30031   ++ HOLD(UZ00031) SYS FMID(HBB2102) REASON(RESTART) DATE(06215)
                                              COMMENT(Reason for the RESTART hold).

USER       DELAY    HBB2102     UZ00032   ++ HOLD(UZ00032) USER FMID(HBB2102) REASON(DELAY) COMMENT(
                                              Delay installing this PTF.).

                    HBB2102     UZ00032   ++ HOLD(UZ00032) USER FMID(HBB2102) REASON(DELAY) COMMENT(
                                              Delay installing this PTF.).
```

Figure 102. SYSMOD Comparison HOLDDATA summary report: sample report

In the SYSMOD Comparison HOLDDATA Report, each report entry includes the ++HOLD card image for the subject HOLD. The SUPPHOLD subentry of the OPTIONS entry can be used to suppress the card images for certain HOLD reason IDs.

If a HOLD reason ID exists in the SUPPHOLD subentry list, the card images for that HOLD will not be included in the SYSMOD Comparison HOLDDATA Report. There will be one report entry per FMID for reason IDs that are to be suppressed. Here again is the example from earlier, except this time the suppressed entries are consolidated.

```
PAGE 2 - NOW SET TO GLOBAL ZONE        DATE 04/07/08    TIME 15:04:07    SMP/E 35.00   SMPHRPT   OUTPUT

SYSMOD COMPARISON HOLDDATA REPORT FOR TARGET ZONE OS390 AND TARGET ZONE OS390A
IN THE GLOBAL ZONE IN CSI SYS2.PRODUCT.VSAM.CSI

TYPE      REASON ID   FMID      SYSMOD    ++HOLD DATA
------    ---------   -------   -------
--------------------------------------------------------------------------------
SYSTEM    DOC         HRSL380   UA36586   ++ HOLD(UA36586) SYS FMID(HRSL380) REASON(ACTION) DATE(07241)
                                              COMMENT
                                               (With this update to the LDAP server it is
                                                recommended, but not required, that the GDBM database
                                                be updated for row level locking.

                                                Detailed instructions on creating and altering GDBM for this
                                                update can be found in the documentation update included with
                                                this PTF.).

          IPL         HBB7740             * ONE OR MORE IPL HOLDs ARE SUPPRESSED FOR HBB7740


                      HDZ1190             * ONE OR MORE IPL HOLDs ARE SUPPRESSED FOR HDZ1190


                      HJE7740             * ONE OR MORE IPL HOLDs ARE SUPPRESSED FOR HJE7740


           ACTION   HBB2102     UZ00030   ++ HOLD(UZ00030) SYS FMID(HBB2102) REASON(ACTION) DATE(06215)
                                              COMMENT(Reason for the ACTION hold).

           RESTART  HBB2102     UZ30031   ++ HOLD(UZ00031) SYS FMID(HBB2102) REASON(RESTART) DATE(06215)
                                              COMMENT(Reason for the RESTART hold).

USER       DELAY    HBB2102               * ONE OR MORE DELAY HOLDs ARE SUPPRESSED FOR HBB212
```

Figure 103. SYSMOD Comparison HOLDDATA Report (suppressed HOLDDATA)

SYSMOD regression report

This report is produced at the completion of APPLY and ACCEPT processing to summarize which SYSMODs were regressed. Regression occurs when SMP/E installs an element from a SYSMOD that did not express a proper PRE or SUP relationship with the RMID and UMID values in the element entry. Regression can occur only when BYPASS(ID) is used to ignore such errors. It does not occur when a new function is being installed, because elements from a function are selected on the basis of functional superiority. SMP/E assumes that service (PTFs and USERMODs) installed on the functionally inferior elements provides ++IF conditional requisite data to ensure that the PTF or USERMOD is at the proper service level.

If no regressions are detected, this report is generally not produced. In certain cases, however, SMP/E detects a regression that is later resolved by other SYSMODs being processed by the same APPLY or ACCEPT command. In this case, the regression report is produced but contains just one message: `NO SYSMODS REGRESSED.`

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SYSMOD REGRESSION REPORT FOR xxxxxxxx yyyyy PROCESSING

NOTE:  '*' INDICATES THAT THE REGRESSED MODID IS SUPD BY ANOTHER SYSMOD BEING PROCESSED. THE REGRESSION MAY BE RESOLVED.
       '-' INDICATES THE CURRENT RMID IS A HIGHER LEVEL SYSMOD THAN THE REGRESSING SYSMOD

REGRESSING  REGRESSED  COMMON    ELEMENTS    CURRENT    OTHER POTENTIALLY
SYSMOD      SYSMOD     TYPE      NAME        RMID       REGRESSED SYSMODS


aaaaaaa     bbbbbbb    ccccccc   ddddddd     eeeeeee    fffffff  fffffff fffffff

            bbbbbbb    ccccccc   ddddddd     eeeeeee    fffffff  fffffff fffffff

aaaaaaa     bbbbbbb    ccccccc   ddddddd     eeeeeee    fffffff  fffffff fffffff
```

Figure 104. SYSMOD regression report: standard format

These are the fields in the report:

xxxxxxxx
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
is CHECK if CHECK was specified on APPLY or ACCEPT. Otherwise, this field is blank.

REGRESSING SYSMOD
identifies the SYSMOD that caused the listed elements to be regressed.

REGRESSED SYSMOD
is a list of SYSMODs that had previously changed the elements listed in the COMMON ELEMENTS fields. These changes may have been overlaid.

COMMON ELEMENTS TYPE and NAME
lists the elements changed by the regressing SYSMOD.

CURRENT RMID
is the RMID for the highest level SYSMOD replacing this element in this SMP/E run. If the element is not being replaced by the current SMP/E run, or is being deleted, this column may be blank.

OTHER POTENTIALLY REGRESSED SYSMODS
is a list of SYSMODs that were superseded by the regressed SYSMOD, but were not superseded by the regressing SYSMOD. This list may include the SYSMOD IDs of APARs that were fixed (superseded) by the regressed SYSMOD, but were not included in the regressing SYSMOD.

Example: APPLY SYSMOD regression report

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SYSMOD REGRESSION REPORT FOR APPLY CHECK PROCESSING

NOTE:  '*' INDICATES THAT THE REGRESSED MODID IS SUPD BY ANOTHER SYSMOD BEING PROCESSED. THE REGRESSION MAY BE RESOLVED.
       '-' INDICATES THE CURRENT RMID IS A HIGHER LEVEL SYSMOD THAN THE REGRESSING SYSMOD

REGRESSING  REGRESSED  COMMON    ELEMENTS    CURRENT    OTHER POTENTIALLY
SYSMOD      SYSMOD     TYPE      NAME        RMID       REGRESSED SYSMODS


UZ00099     UZ00001    MODULE    HMAB0123               AZ00050  AZ00051 AZ00052

            UZ00002    MACRO     HMAMAC01               AZ00055
                       MODULE    HMAB0456

            UZ00003    MODULE    HMAB0790               AZ00056  AZ00057
                       MODULE    HMAB0012
                       MODULE    HMAB0345

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

UZ01000     UZ00100    MODULE    MOD002      UZ01000    UZ00002  UZ00003 UZ00004 UZ00005
                       MODULE    MOD003      UZ01000    UZ00006  UZ00007 UZ00012 UZ00013
                       SOURCE    MOD002      UZ01000    UZ00014  UZ00015 UZ00016 UZ00017
                       SOURCE    MOD003      UZ01000
                       SOURCE    SRC002
                       SOURCE    SRC003
```

Figure 105. SYSMOD regression report: sample report for APPLY

SYSMOD status report

This report is produced at the completion of APPLY, ACCEPT, and RESTORE processing to summarize the processing that occurred for every eligible SYSMOD. The SYSMODs are listed in alphanumeric order.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT


 SYSMOD STATUS REPORT FOR xxxxxxxx PROCESSING     SYSMODS yyyyyyyy
- nnnn

 NOTE:  '-' INDICATES THE REQUISITE SYSMOD OR HOLD CONDITION IS NOT SATISFIED
        '*' INDICATES THE NON SATISFIED REQUISITE SYSMOD OR HOLD CONDITION IS BYPASSED
        '#' INDICATES THE SUPERSEDING SYSMOD WAS NOT PROCESSED

 SYSMOD   STATUS    TYPE      FMID      REQUISITE SYSMODS, SUPBY SYSMODS, HOLD REASON-IDS, AND CAUSER SYSMODS
 aaaaaaa  bbbbbbb   ccccccc   ddddddd   eeeeeee  fffffff…
 aaaaaaa  bbbbbbb   ccccccc   ddddddd   eeeeeee  fffffff…
 aaaaaaa  bbbbbbb   ccccccc   ddddddd   eeeeeee  fffffff…
 aaaaaaa  bbbbbbb   ccccccc   ddddddd   eeeeeee  fffffff…
 aaaaaaa  bbbbbbb   ccccccc   ddddddd   eeeeeee  fffffff…
```

Figure 106. SYSMOD status report: standard format

These are the fields in the report:

xxxxxxxx
is the SMP/E command being processed: APPLY, ACCEPT, or RESTORE.

yyyyyyyy
indicates the type of processing that was done: APPLIED, ACCEPTED, or RESTORED.

nnnn
is the number of SYSMODs with a status of APPLIED, ACCEPTED, or RESTORED; that number depends on the command that was processed.

SYSMOD
identifies the SYSMOD that was processed.

STATUS
describes what happened to the SYSMOD.

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

DELETED
The SYSMOD was explicitly or implicitly deleted.

ERROR
SYSMOD processing stopped after some target libraries or SMP/E libraries were updated, but before the SYSMOD was completely processed. A SYSMOD is completely processed when all its elements have been processed and all its requisites have been completely processed. See SMPOUT to determine the cause of the error.

Note: ERROR does not appear when the CHECK operand is specified on the command.

EXCLUDED
The SYSMOD was specified on the EXCLUDE operand.

HELD
The SYSMOD was held because one or more of HOLD reason IDs were not resolved.

INCMPLT
SYSMOD processing is incomplete because of some failure. No target libraries were updated.

NOGO
The SYSMOD was not processed before any updates. This can happen when a related SYSMOD has an error. See SMPOUT to determine the cause of the error.

NOGO(E)
SYSMOD processing stopped because a required SYSMOD was excluded.

NOGO(H)
SYSMOD processing stopped because a required SYSMOD was held.

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

Note: Not all superseded SYSMODs are listed in the report. For example, SYSMODs that were not selected for processing and appear only in another SYSMOD's ++VER SUP operand are not listed.

TYPE
is the SYSMOD type: APAR, FUNCTION, PTF, or USERMOD. For superseded SYSMODs that have not been received, this field is blank.

FMID
is the function SYSMOD that owns the SYSMOD. For superseded SYSMODs, this field is blank.

REQUISITE SYSMODS, SUPBY SYSMODS, HOLD REASON-IDs, AND CAUSER SYSMODS
lists SYSMODs or reason IDs associated with the SYSMODs being installed. If a SYSMOD was not installed, these SYSMODs or reason IDs are preceded by a character indicating why (-, *, or #). The list of SYSMODs or reason IDs is preceded by one of the following values, which indicate the type of SYSMOD or reason ID.

CAUSER
SYSMODs whose failure led to the failure of the SYSMOD in the SYSMOD field. All the SYSMODs in the CAUSER field are in the Causer SYSMOD Summary report, along with a summary of the related error and, when feasible, a list of possible causes for the error. This holds true even when the SYSMOD in the SYSMOD field and the causer SYSMOD are the same.

HOLDE
ERROR reason IDs for the SYSMOD.

HOLDF
FIXCAT reason IDs for the SYSMOD.

HOLDS
SYSTEM reason IDs for the SYSMOD.

HOLDU
USER reason IDs for the SYSMOD.

IFREQ
Conditional requisites for the SYSMOD, as defined by its associated ++IF statements or, if the SYSMOD is a function, defined by previously processed SYSMODs.

PRE
Prerequisites for the SYSMOD.

REQ
Requisites for the SYSMOD.

SUPBY
SYSMODs that supersede the SYSMOD.

XZIFREQ

- For APPLY and ACCEPT processing, these are:
  - Conditional requisites for the SYSMOD, as defined by its associated ++IF statements. Cross-zone requisite checking has found that the FMID named on the ++IF exists in another zone.
  - Conditional requisites (CIFREQ data) found in other zones for function SYSMODs being installed into the set-to zone.
- For RESTORE processing, these are the causer SYSMODs in another zone. Cross-zone requisite checking has found that another SYSMOD (a causer) has named the SYSMOD being restored as a requisite on a ++IF statement.

XZIFREQ is listed as a parenthesized pair of names —for example, (SYSMOD2 ZONE3)— identifying a SYSMOD and the participating cross-zone.

For HOLDE, HOLDS, HOLDU, IFREQ, PRE, REQ, and XZIFREQ:

- A dash (–) next to a listed SYSMOD means that SYSMOD has NOGO status and may not be 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 commands. This means that even if the SYSMOD is not available for processing, the SYSMOD that has specified it as a requisite can be processed.

For SUPBY:

- If a pound sign (#) appears next to a listed SYSMOD, that SYSMOD has not been successfully processed. As long as at least one superseding SYSMOD has been successfully processed, the superseded SYSMOD is considered to be installed.

Example: APPLY SYSMOD status report

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

 SYSMOD STATUS REPORT FOR APPLY PROCESSING     SYSMODS APPLIED - 349

 NOTE:  '-' INDICATES THE REQUISITE SYSMOD OR HOLD CONDITION IS NOT SATISFIED
        '*' INDICATES THE NON SATISFIED REQUISITE SYSMOD OR HOLD CONDITION IS BYPASSED
        '#' INDICATES THE SUPERSEDING SYSMOD WAS NOT PROCESSED

 SYSMOD   STATUS    TYPE      FMID      REQUISITE SYSMODS, SUPBY SYSMODS, HOLD REASON-IDS, AND CAUSER SYSMODS

 UY15425  NOGO      PTF       HAE1500   IFREQ   -UZ63106
                                        XZIFREQ  (UZ12345 ZONE1)    -(UZ98765 ZONE321)
                                        CAUSER   UY15425   UZ65387
⋮
 UZ62368  HELD      PTF       JTC2412   PRE      UZ59092   UZ60917
                                        HOLDE   -AZ71745  -AZ75533
                                        CAUSER   UZ62368
⋮
 UZ65356  NOGO(H)   PTF       JTC2412   PRE     -UZ63740
                                        CAUSER   UZ63419   UZ63420

 UZ65375  NOGO(H)   PTF       HHM1302   IFREQ    UZ61932   UZ61934   UZ62802   UZ65385   UZ65386  -UZ65387
                                                -UZ65388  -UZ65389
                                        XZIFREQ -(UZ33333 ZONE444)   (UZ33333 ZONE5)
                                        REQ      UZ65376   UZ65377   UZ65378   UZ65379   UZ65380   UZ65381
                                                 UZ65382   UZ65383   UZ65384
                                        CAUSER   UZ65387   UZ65388
⋮
```

Figure 107. SYSMOD status report: sample report for APPLY

UNLOAD summary report

This report is produced during UNLOAD processing to summarize which entries were found in the set-to zone and which entries were not.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNLOAD SUMMARY FOR aaaaaaaa

ENTRY-TYPE   ENTRY-NAME  STATUS

bbbbbbbb     cccccccc    ddddddddddddddddd
bbbbbbbb     cccccccc    ddddddddddddddddd
bbbbbbbb     cccccccc    ddddddddddddddddd
bbbbbbbb     cccccccc    ddddddddddddddddd
```

Figure 108. UNLOAD summary report: standard format

These are the fields in the report:

aaaaaaaa
is the name of the zone containing the entries being unloaded.

ENTRY-TYPE
is the type of entry SMP/E looked for. If no specific entries of a given type were selected, that entry type is shown only once. If several specific entries of a given type were selected, that entry type is shown for each of the selected entries.

ENTRY-NAME
is the name of an entry specified on the UNLOAD command. If no specific entries were selected, this is blank.

STATUS
indicates whether any entries were found.

FOUND
The specified entry or entry type was found.

NOT FOUND
The specified entry or entry type was not found.

EMPTY ZONE – NO ENTRIES FOUND
This may appear for the UNLOAD command. No entries were found in the specified zone. This is the only line in the report. The entry type and entry name fields are blank.

Example: UNLOAD summary report

Figure 109 on page 531 shows the UNLOAD Summary report that may accompany the UNLOAD output for the following command:

```
UNLOAD MOD MAC SRC CLIST(CLIST01,CLIST02,CLIST03).
```

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNLOAD SUMMARY FOR MVSTGT1

ENTRY-TYPE   ENTRY-NAME  STATUS

CLIST        CLIST01     FOUND
CLIST        CLIST02     NOT FOUND
CLIST        CLIST03     FOUND
MAC                      NOT FOUND
MOD                      FOUND
SRC                      FOUND
```

Figure 109. UNLOAD summary report: sample report

Unresolved HOLD reason report

This report is produced at the completion of APPLY and ACCEPT processing to identify SYSMODs that have been terminated because of one or more unresolved HOLD conditions. Only HOLD conditions causing termination are listed in this report. If a SYSMOD also had HOLD conditions bypassed, these appear in the "Bypassed HOLD Reason Report".

- The report is split into three sections. Each section has a unique format and starts on a new page in the SMPRPT output. The three sections of the report include:
  - ERROR HOLDS
  - FIXCAT HOLDS
  - SYSTEM and USER HOLDS
- Each section of the report uses a different technique to sort and group the information based on the value of the types of information.

Format and explanation of data for ERROR HOLDS section

The standard format of the ERROR HOLDS section of the "Unresolved HOLD Reason Report" is illustrated in Figure 110 on page 532. The data in this section of the report is sorted by FMID and then by HOLD CLASS within the ERRORS relating to a specific FMID.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNRESOLVED HOLD REASON REPORT FOR xxxxxx yyyyy PROCESSING

NOTE: THE SYSMODS LISTED IN THIS REPORT ALSO APPEAR IN THE CAUSER SYSMOD SUMMARY REPORT.

                         HOLD        MISSING     HELD         RESOLVING      RESOLVER
TYPE        FMID         CLASS       APAR        SYSMOD       SYSMOD         STATUS
----        -------      -------     -------     -------      ---------      --------
aaaaaa      bbbbbbb      ccccccc     ddddddd     eeeeeee      fffffff        ggggggg
                         ddddddd     eeeeeee     fffffff        ggggggg
                                                 fffffff        ggggggg
                         ccccccc     ddddddd     eeeeeee      fffffff        ggggggg
```

Figure 110. Unresolved HOLD reason report: ERROR HOLD section

These are the fields in the report:

xxxxxx
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
is CHECK, if CHECK was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

TYPE
is the type of ++HOLD. In this section of the report, the reason is always:

ERROR
indicates an error hold.

FMID
is the FMID of the held SYSMOD.

HOLD CLASS
is the hold class specified on the CLASS operand of the ++HOLD MCS. If the CLASS operand is not specified on the ++HOLD, this field is blank.

MISSING APAR
is the unresolved HOLD reason ID that caused the named SYSMOD to fail.

HELD SYSMOD
is the ID of a SYSMOD that failed due to an unresolved HOLD condition.

RESOLVING SYSMOD
is the list of SYSMODs that can resolve the HOLD reason ID. If a fixing SYSMOD is identified in the FIX field of the SMRTDATA on the ++HOLD MCS, that SYSMOD is included in this list. If a SYSMOD is in process for the APPLY or ACCEPT command and it supersedes the reason ID APAR, that SYSMOD is also in this list. If no SYSMODs are in the list, this field and the RESOLVER STATUS field will both be blank.

RESOLVER STATUS
describes what happened to the SYSMOD that resolves the HOLD.

ERROR
SYSMOD processing stopped after updates were made to target or distribution libraries or to the target or dlib zone, but before the SYSMOD was completely processed.

EXCLUDED
the SYSMOD was specified on the EXCLUDE operand.

HELD
the SYSMOD was held because one or more HOLD reason IDs were not resolved.

INCMPLT
SYSMOD processing is incomplete because of some failure. No library or zone updates were made.

MISSING
the SYSMOD was either not available (not in the global zone) or was not selected for APPLY or ACCEPT command processing.

NOGO
SYSMOD processing stopped before any library or zone updates were made for this SYSMOD. For example, a required SYSMOD was missing.

NOGO(E)
SYSMOD processing stopped because a required SYSMOD was excluded.

NOGO(H)
SYSMOD processing stopped because a required SYSMOD was held.

Format and explanation of data for FIXCAT HOLDS section

The standard format of the FIXCAT HOLDS section of the "Unresolved HOLD Reason Report" is illustrated in Figure 111 on page 533. This section of the report is sorted by fix category, and then by the SYSMOD that can resolve the HOLD. The sort order then arranges the report entries by FMID rather than reason ID.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNRESOLVED HOLD REASON REPORT FOR xxxxxx yyyyy PROCESSING

NOTE: THE SYSMODS LISTED IN THIS REPORT ALSO APPEAR IN THE CAUSER SYSMOD SUMMARY REPORT.

                                        HOLD      MISSING      HELD       RESOLVING      RESOLVER
TYPE        FIX CATEGORY    FMID        CLASS     APAR         SYSMOD     SYSMOD         STATUS
----        ------------    ----        -----     -------      ------     ---------      --------
aaaaaa      hhhhhhhhhhhhhhhhhhhhhhhhhh
                            bbbbbbb     ccccccc   ddddddd      eeeeeee    fffffff        ggggggg
                                                  ddddddd      eeeeeee    fffffff        ggggggg
                                                                          fffffff        ggggggg
aaaaaa      hhhhhhhhhhhhhhhhhhhhhhhhhh
                            bbbbbbb     ccccccc   ddddddd      eeeeeee    fffffff        ggggggg
```

Figure 111. Unresolved HOLD reason report: FIXCAT HOLD section

These are the fields in the report:

xxxxxx
is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
is CHECK, if CHECK was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

TYPE
is the type of ++HOLD. In this section of the report, the reason is always:

FIXCAT
indicates a fix category hold

FIX CATEGORY
is a fix category specified on the CATEGORY operand of the ++HOLD MCS. If multiple fix category values are specified on the ++HOLD, then the unresolved reason will be reported once for each fix category value that matches any of the fix categories of interest specified by the user.

RESOLVING SYSMOD
is the list of SYSMODs that can resolve the HOLD reason ID. If a fixing SYSMOD is identified in the RESOLVER operand on the ++HOLD MCS, then that SYSMOD is included in this list. If a SYSMOD is in process for the APPLY or ACCEPT command and it supersedes the reason ID APAR, then that SYSMOD is also in this list. If no SYSMODs are in the list, this field and the RESOLVER STATUS field are both blank.

The remaining parameters are the same as those defined in “Format and explanation of data for ERROR HOLDS section” on page 532.

Format and explanation of data for SYSTEM and USER HOLDS section

The standard format of the "Unresolved HOLD Reason Report" is illustrated in Figure 112 on page 534. This last section of the report maintains the format and information in the previous version of the report.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNRESOLVED HOLD REASON REPORT FOR xxxxxx yyyyy PROCESSING

NOTE: THE SYSMODS LISTED IN THIS REPORT ALSO APPEAR IN THE CAUSER SYSMOD SUMMARY REPORT.

TYPE         REASON ID   FMID       SYSMOD      ++HOLD DATA

aaaaaa       bbbbbbb     ccccccc    ddddddd     eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

                                    ddddddd     eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

                         ccccccc    ddddddd     eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

aaaaaa       bbbbbbb     ccccccc    ddddddd     eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                                                eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

Figure 112. Unresolved HOLD reason report: SYSTEM and USER HOLD section

These are the fields in the report:

xxxxxx
: is the SMP/E command being processed: APPLY or ACCEPT.

yyyyy
: is CHECK, if CHECK was specified on the APPLY or ACCEPT command. Otherwise, this field is blank.

TYPE
: is the type of ++HOLD for the following reason ID.

  SYSTEM
  : indicates a system hold.

  USER
  : indicates a user hold.

REASON ID
: is a list of one or more HOLD reason IDs that caused the named SYSMOD to fail.

FMID
: is the FMID of the SYSMOD in the next field of the report.

SYSMOD
: is the ID of a SYSMOD that failed due to an unresolved HOLD condition.

++HOLD DATA
: this is one of the following selections:

  - the ++HOLD MCS that was received for the HOLD reason ID. This data may span several lines and is provided for each reason ID, or
  - an indication that the ++HOLD MCS has been suppressed, for the reason IDs specified in the Suppress HOLDDATA (SUPPHOLD) subentry of the active OPTIONS entry

Examples

The following examples illustrate the three sections of the "Unresolved HOLD Reason Report":

- “Example 1: ERROR section of the unresolved HOLD reason report” on page 535
- “Example 2: FIXCAT section of the unresolved HOLD reason report” on page 535
- “Example 3: SYSTEM and USER section of the unresolved HOLD reason report” on page 535

Example 1: ERROR section of the unresolved HOLD reason report

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNRESOLVED HOLD REASON REPORT FOR APPLY CHECK PROCESSING

NOTE: THE SYSMODS LISTED IN THIS REPORT ALSO APPEAR IN THE CAUSER SYSMOD SUMMARY REPORT.

                    HOLD        MISSING     HELD          RESOLVING     RESOLVER
TYPE      FMID      CLASS       APAR        SYSMOD        SYSMOD        STATUS
------    -------   -------     -------     -------       ---------     --------
ERROR     HKDB310   HIPER       AK18603     HKDB310       UK14916       MISSING
                              AK19334     HKDB310       UK12397       MISSING
                                                        UK53888       NOGO
                    PE          AK25387     UK63290       UK64221       NOGO(H)
```

Figure 113. Unresolved HOLD reason report: ERROR HOLD example

Example 2: FIXCAT section of the unresolved HOLD reason report

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNRESOLVED HOLD REASON REPORT FOR APPLY CHECK PROCESSING

NOTE: THE SYSMODS LISTED IN THIS REPORT ALSO APPEAR IN THE CAUSER SYSMOD SUMMARY REPORT.

                                        HOLD        MISSING      HELD     RESOLVING    RESOLVER
TYPE        FIX CATEGORY  FMID          CLASS       APAR         SYSMOD   SYSMOD       STATUS
------      ------------- ------        -------     -------      -------  ---------    --------
FIXCAT      IBM.Coexistence.z/OS.V1R8
                          HBB7720       PSP         A014733      HBB7720  UO73920      MISSING
                                                                         U136622      NOGO
            IBM.Device.zIPP
                          HBB7720       PSP         AO23456      HBB7720  UO95873      HELD
                          HBB7730       PSP         AA15968      HBB7730  UA27113      MISSING
            IBM.Device.2094
                          HBB7730       PSP         AA15968      HBB7730  UA27113
MISSING
```

Figure 114. Unresolved HOLD reason report: FIXCAT HOLD example

Example 3: SYSTEM and USER section of the unresolved HOLD reason report

The example in Figure 115 on page 535 shows both a SYSTEM and a USER HOLD. In this report you will note that the comment text supplies a lot of information about the HOLD.

```
PAGE nnnn  - NOW SET TO TARGET ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

UNRESOLVED HOLD REASON REPORT FOR APPLY CHECK PROCESSING

NOTE: THE SYSMODS LISTED IN THIS REPORT ALSO APPEAR IN THE CAUSER SYSMOD SUMMARY REPORT.

TYPE      REASON ID   FMID    SYSMOD   ++HOLD DATA
------    ---------   ------- ------   ------------------------------------------------------------------------------
SYSTEM    ACTION      HKDB310 UK12345  ++ HOLD(UK12345) SYS FMID(HKDB310) REASON(ACTION) DATE(07215)
                                           COMMENT
                                           (OA16310
                                           ********************************************************************
                                           FUNCTION AFFECTED: GDPS/XRC
                                           Geographically Dispersed Parallel Sysplex
                                           ********************************************************************
                                           DESCRIPTION : Modification to GRS Resource Name List (RNL)
                                           ********************************************************************
                                           TIMING : After APPLY and prior to restarting XRC
                                           ********************************************************************
                                           If your installation exploits the GDPS capability for the
                                           Controlling system to recover the FlashCopy devices while
                                           the system data movers remain active, an additional statement
                                           must be added to the GRSRNLxx parmlib member:

                                           RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(SYSZXRC1)

                                                                                  This statement is in addition to, and does not replace, the
                                                                                  other statements required for the capability:

                                           RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(SYSXRCMS)
                                           RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(SYSXRCLS)

                                           Note, this update is required even if you do not to intend to
                                           exploit the XRC scalability enhancements. ).

USER      DELAY       HKDB310    UK33778      ++ HOLD(UK33778) USER FMID(HKDB310) REASON(DELAY) COMMENT(
                                                    Delay installing this PTF. ).
```

Figure 115. Unresolved HOLD reason report: sample of the SYSTEM and USER section of the report

ZONEEDIT summary report

This report is produced during ZONEEDIT processing to show the DDDEF and UTILITY entry names that were changed.

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

SUMMARY REPORT FOR ZONEEDIT

ENTRY NAME     FIELD NAME    FROM                                          TO

aaaaaaaa       bbbbbbbb      ccccccccc…                                    ddddddddd…
aaaaaaaa       bbbbbbbb      ccccccccc…                                    ddddddddd…
aaaaaaaa       bbbbbbbb      ccccccccc…                                    ddddddddd…
aaaaaaaa       bbbbbbbb      ccccccccc…                                    ddddddddd…
aaaaaaaa       bbbbbbbb      ccccccccc…                                    ddddddddd…
```

Figure 116. ZONEEDIT summary report: standard format

These are the fields in the report:

ENTRY NAME
is the name of the entry that was changed. The valid entry types are DDDEF, UTILITY, and XZENTRIES.

Note: XZENTRIES is not an actual entry type. It is used to change a zone name in all the cross-zone subentries in a specified zone.

FIELD NAME
is the subentry that was changed.

For a DDDEF entry, the subentry name can be any of the following types:

- DATASET
- PATH
- SYSOUT
- UNIT
- VOLUME
- WAIT

For a UTILITY entry, the subentry name can be any of the following types:

- NAME
- PRINT

For XZENTRIES, the subentry name can be any of the following types:

- XZLMOD in MOD entries
- XZMOD in LMOD entries
- TIEDTO in the TARGETZONE entry

FROM
is the old value of the subentry. It can be up to 44 characters long, except for the pathname value of the PATH subentry, which can be up to 255 characters long. If the pathname (including its enclosing apostrophes) is more than 44 characters long, the pathname spans multiple lines.

TO
is the new value of the subentry. It can be up to 44 characters long, except for the pathname value of the PATH subentry, which can be up to 255 characters long. If the pathname (including its enclosing apostrophes) is more than 44 characters long, the pathname spans multiple lines.

Examples

The following sample reports are provided:

- "Example 1: ZONEEDIT summary report for DDDEF entries" on page 537
- "Example 2: ZONEEDIT summary report for PATH subentries" on page 537
- "Example 3: ZONEEDIT summary report for XZENTRIES" on page 537

Example 1: ZONEEDIT summary report for DDDEF entries

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
SUMMARY REPORT FOR ZONEEDIT
ENTRY NAME    FIELD NAME    FROM                                          TO
AOS12         UNIT          3330                                          3350
AOS22         UNIT          3330                                          3350
LPALIB        UNIT          3330                                          3350
LINKLIB       UNIT          3330                                          3350
```

Figure 117. ZONEEDIT summary report: sample report for DDDEF entries

Example 2: ZONEEDIT summary report for PATH subentries

Figure 118 on page 537 contains examples of the long pathnames that can appear in PATH subentries.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
SUMMARY REPORT FOR ZONEEDIT
ENTRY NAME    FIELD NAME    FROM                                          TO

BPXLIB1       PATH          '/this/pathname/needs/only/1/report/record/'    '/and/this/name/needs/only/1/report/record/'

BPXLIB2       PATH          '/this/is/a/very/long/path/name/It requires/    '/so/too/is/this/a/very/long/path/name/It re
                            more/than/a/single/line/to/display./it/also/    quires/more/than/a/single/line/to/display./i
                            contains/a/''/single/quote/'                    t/also/contains/a/''/single/quote/'
```

Figure 118. ZONEEDIT summary report: sample report for PATH subentries

Example 3: ZONEEDIT summary report for XZENTRIES

Suppose you have used the LINK command to link-edit modules from zone CICS1 into load modules in zone SYS1. Later, because of new zone-naming conventions, you used the ZONERENAME command to rename zone CICS1 to CICSPRD. When the ZONERENAME processing was completed, you realized that zone SYS1 was the only zone connected to zone CICS1 by cross-zone subentries.

You now have to change the cross-zone subentries in zone SYS1 so that SYS1 is connected to the renamed zone, CICSPRD. The following ZONEEDIT commands make this change:

```
SET      BDY (SYS1)       /* Set to zone to edit.       */.
ZONEEDIT XZENTRIES        /* Edit cross-zone subentries.*/.
CHANGE   ZONEVALUE(CICS1  /* Change zone from old name  */
                 CICSPRD) /* to new name.               */.
ENDZONEEDIT               /* End of ZONEEDIT.           */.
```

After SMP/E processes these commands, the ZONEEDIT Summary report that is produced is similar to the following sample report.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT
NOW SET TO TARGET ZONE SYS1
SUMMARY REPORT FOR ZONEEDIT
ENTRY NAME     FIELD NAME    FROM                                          TO
SYS1           TIEDTO        CICS1                                         CICSPRD
LMOD1          XZMOD         CICS1                                         CICSPRD
MOD1           XZLMOD        CICS1                                         CICSPRD
```

Figure 119. ZONEEDIT summary report: sample report for XZENTRIES

ZONEMERGE report

This report is produced during ZONEMERGE processing to show the entries that were copied. If an error occurs and command processing stops, you can use this report to determine how far the merge operation has been completed. This report is arranged by entry type and, within entry type, alphanumerically by entry name. If no entries were merged, the ZONEMERGE report states `NO ENTRIES MERGED. NO APPLICABLE ENTRIES IN FROM ZONE.`

Format and explanation of data

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                SMP/E ZONEMERGE REPORT FROM ZONE xxxxxxx TO ZONE yyyyyyy

TYPE        NAME        ACTION        REASON

aaaaaaaa    bbbbbbbb    ccccccccc     ddddddddddddddddddddddd
aaaaaaaa    bbbbbbbb    ccccccccc     ddddddddddddddddddddddd
aaaaaaaa    bbbbbbbb    ccccccccc     ddddddddddddddddddddddd
aaaaaaaa    bbbbbbbb    ccccccccc     ddddddddddddddddddddddd
aaaaaaaa    bbbbbbbb    ccccccccc     ddddddddddddddddddddddd
aaaaaaaa    bbbbbbbb    ccccccccc     ddddddddddddddddddddddd
```

Figure 120. ZONEMERGE Report: Standard Format

These are the fields in the report:

xxxxxxx
is the zone containing the entries to be copied, also called the *FROM zone*.

yyyyyyy
is the zone to which you are copying entries, also called the *TO zone*.

TYPE
is the entry type.

NAME
is the name of the entry.

ACTION
describes what SMP/E did with that entry.

MERGED
The specified entry was in the FROM zone but not in the TO zone; so SMP/E added it to the TO zone.

REPLACED
The specified entry was in both the FROM zone and the TO zone. Because REPLACE was specified on ZONEMERGE, SMP/E replaced the entry in the TO zone with the entry in the FROM zone.

NOT MERGED
The specified entry was in both the FROM zone and the TO zone. Because NOREPLACE was specified on ZONEMERGE, SMP/E did not replace the entry in the TO zone.

SRELS MERGED
The specified entry was in the ZONE definition entry of the FROM zone that did not exist in the ZONE definition entry of the TO zone. These SRELs were added to the ZONE definition entry of the TO zone.

REASON
is the reason the entry was not merged if the ACTION field shows `NOT MERGED`. The only value for this field is `REPLACE NOT SPECIFIED`.

Examples

The following sample reports are provided:

- "Example 1: Merge to null zone" on page 539
- "Example 2: Merge to existing zone with REPLACE operand" on page 539
- "Example 3: Merge to existing zone with NOREPLACE operand" on page 540

Example 1: Merge to null zone

Assume that you are merging zone TGT1 into a null zone TGT2. Figure 121 on page 539 is an example of the ZONEMERGE report when the TO zone is null.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                SMP/E ZONEMERGE REPORT FROM ZONE TGT1    TO ZONE TGT2

TYPE          NAME        ACTION        REASON

DDDEF         AMACLIB     MERGED
DDDEF         ASAMPLIB    MERGED
ASSEM         ASSEM01     MERGED
ASSEM         ASSEM02     MERGED
ASSEM         ASSEM03     MERGED
LMOD          LMOD01      MERGED
LMOD          LMOD02      MERGED
LMOD          LMOD03      MERGED
MACRO         MAC01       MERGED
MACRO         MAC02       MERGED
MACRO         MAC03       MERGED
MODULE        MOD01       MERGED
MODULE        MOD02       MERGED
MODULE        MOD03       MERGED
SOURCE        SRC01       MERGED
SOURCE        SRC02       MERGED
SOURCE        SRC03       MERGED
DLIB          AMACLIB     MERGED
DLIB          ASRCLIB     MERGED
SYSMOD        AZ10001     MERGED
SYSMOD        AZ10002     MERGED
SYSMOD        UZ00001     MERGED
SYSMOD        UZ00002     MERGED
SYSMOD        UZ00003     MERGED
TARGETZONE    TGT2        SRELS MERGED
```

Figure 121. ZONEMERGE report: sample report for merging to a null zone

Example 2: Merge to existing zone with REPLACE operand

If zone TGT2 contained the following entries:

DDDEF ASAMPLIB
ASSEM ASSEM01
LMOD   LMOD02
LMOD   LMOD03
MACRO MAC02
MODULE MOD02
SOURCE SRC03

and you specified the following commands:

```
SET      BDY(TGT2)          /* Set to target zone.      */.
ZMRG     (TGT1)             /* Merge TGT1               */
```

```
         INTO(TGT2)          /* into TGT2,               */
         REPLACE             /* replacing like entries.  */.
```

then the ZONEMERGE report looks like Figure 122 on page 540.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                SMP/E ZONEMERGE REPORT FROM ZONE TGT1    TO ZONE TGT2

TYPE          NAME        ACTION        REASON

DDDEF         AMACLIB     MERGED
DDDEF         ASAMPLIB    REPLACED
ASSEM         ASSEM01     REPLACED
ASSEM         ASSEM02     MERGED
ASSEM         ASSEM03     MERGED
LMOD          LMOD01      MERGED
LMOD          LMOD02      REPLACED
LMOD          LMOD03      REPLACED
MACRO         MAC01       MERGED
MACRO         MAC02       REPLACED
MACRO         MAC03       MERGED
MODULE        MOD01       MERGED
MODULE        MOD02       REPLACED
MODULE        MOD03       MERGED
SOURCE        SRC01       MERGED
SOURCE        SRC02       MERGED
SOURCE        SRC03       REPLACED
DLIB          AMACLIB     MERGED
DLIB          ASRCLIB     MERGED
SYSMOD        AZ10001     MERGED
SYSMOD        AZ10002     MERGED
SYSMOD        UZ00001     MERGED
SYSMOD        UZ00002     MERGED
SYSMOD        UZ00003     MERGED
TARGETZONE    TGT2        SRELS MERGED
```

Figure 122. ZONEMERGE report: sample report for REPLACE processing

Example 3: Merge to existing zone with NOREPLACE operand

Assuming the same two zones were merged and you specified the following commands:

```
SET      BDY(TGT2)           /* Set to target zone.      */.
ZMRG     (TGT1)              /* Merge TGT1               */
         INTO(TGT2)          /* into TGT2.               */
         NOREPLACE           /* Don't merge like entries.*/.
```

the ZONEMERGE report looks like Figure 123 on page 540.

```
PAGE nnnn  - NOW SET TO zzzzzz ZONE nnnnnnn  DATE mm/dd/yy TIME hh:mm:ss SMP/E 36.nn SMPRPT OUTPUT

                SMP/E ZONEMERGE REPORT FROM ZONE TGT1    TO ZONE TGT2

TYPE          NAME        ACTION        REASON

DDDEF         AMACLIB     MERGED
DDDEF         ASAMPLIB    NOT MERGED    REPLACE NOT SPECIFIED
ASSEM         ASSEM01     NOT MERGED    REPLACE NOT SPECIFIED
ASSEM         ASSEM02     MERGED
ASSEM         ASSEM03     MERGED
LMOD          LMOD01      MERGED
LMOD          LMOD02      MERGED
LMOD          LMOD03      MERGED
MACRO         MAC01       MERGED
MACRO         MAC02       NOT MERGED    REPLACE NOT SPECIFIED
MACRO         MAC03       MERGED
MODULE        MOD01       MERGED
MODULE        MOD02       NOT MERGED    REPLACE NOT SPECIFIED
MODULE        MOD03       MERGED
SOURCE        SRC01       MERGED
SOURCE        SRC02       MERGED
SOURCE        SRC03       NOT MERGED    REPLACE NOT SPECIFIED
DLIB          AMACLIB     MERGED
DLIB          ASRCLIB     MERGED
SYSMOD        AZ10001     MERGED
SYSMOD        AZ10002     MERGED
SYSMOD        UZ00001     MERGED
SYSMOD        UZ00002     MERGED
SYSMOD        UZ00003     MERGED
TARGETZONE    TGT2        SRELS MERGED
```

Figure 123. ZONEMERGE report: sample report for NOREPLACE processing

Appendix A. Processing the SMP/E RC operand

SMP/E keeps records of the highest return code for each command processed during a single SMP/E run. Before SMP/E processes a command, it checks whether the return codes for previous commands are less than the maximum allowed. If so, SMP/E can process the command; otherwise, the command fails.

You can change which return codes SMP/E checks for by specifying the RC operand on the command you want to process. This operand must be the last one specified on the command. It indicates which commands and return codes SMP/E is to check before processing the current command. The return code must be a decimal number greater than or equal to 0 and less than 16. For example, if you want the ACCEPT command to run unless the return code from a previous APPLY command is higher than 8, and you want SMP/E to check the default values for JCLIN, UCLIN, and SET, you should code this RC operand on the ACCEPT command:

```
RC(APPLY=08,JCLIN=08,UCLIN=08,SET=00)
```

Table 30 on page 541 shows the default maximum return codes for each SMP/E command. Using these default values, for example, SMP/E processes an ACCEPT command if the return codes for previous commands meet these conditions:

- The highest return code for a SET command was 0. This condition is required because if the SET command fails, SMP/E does not know which zone to process for the subsequent commands.
- The highest return code for a JCLIN or UCLIN command was less than 8. This condition is required because these commands prepare data sets, such as the CSI, for processing by subsequent commands. If these commands fail, the other commands could either fail or run incorrectly.
- The highest return code for any other command was less than 12.

Table 30. Default maximum return codes for commands previously processed

Command to be processed	Maximum for all commands except JCLIN, UCLIN, and SET	Maximum for JCLIN or UCLIN	Maximum for SET
ACCEPT	12	08	00
APPLY	12	08	00
BUILDMCS	12	08	00
CLEANUP	12	08	00
DEBUG	16	16	00
GENERATE	12	08	00
GZONEMERGE	12	08	00
JCLIN	12	08	00
LINK	12	08	00
LIST	16	12	00
LOG	12	08	00
RECEIVE	12	08	00
REJECT	12	08	00
REPORT	16	16	00

Table 30. Default maximum return codes for commands previously processed (continued)

Command to be processed	Maximum for all commands except JCLIN, UCLIN, and SET	Maximum for JCLIN or UCLIN	Maximum for SET
RESETRC	16	08	00
RESTORE	12	08	00
SET	16	16	16
UCLIN	12	08	00
UNLOAD	12	08	00
UPGRADE	16	16	00
ZONECOPY	12	08	00
ZONEDELETE	12	08	00
ZONEEDIT	12	08	00
ZONEEXPORT	12	08	00
ZONEIMPORT	12	08	00
ZONEMERGE	12	08	00
ZONERENAME	12	08	00

Return code processing for the RC operand is similar to processing for the default maximum return codes. Before SMP/E processes a command with the RC operand, it checks whether the return codes for previous commands are **less than or equal to** the maximums specified on the RC operand. If so, SMP/E can process the command; otherwise, the command fails. There is a difference, however. SMP/E checks return codes only for the commands specified on the RC operand. Return codes for any commands not specified do not affect processing for the current command. Therefore, if you use the RC operand, you should specify every command whose return code you want SMP/E to check.

Appendix B. Sharing SMP/E data sets

Processing any SMP/E command requires a number of resources, such as shared and exclusive use of various data sets. These data sets include:

- The target libraries, distribution libraries, various temporary work data sets, and SYSOUT data sets
- All the CSI data sets and the SMPPTS data set

If you want to run more than one SMP/E job concurrently, either you must ensure data set integrity, or SMP/E must automatically provide that integrity.

You can manage the first category of data sets through:

- The DISP=OLD or DISP=SHR DD statement parameters. For more information, see the section "Sample SMP/E Cataloged Procedure" in *z/OS SMP/E User's Guide*.
- DDDEF entries. For more information, see the section on "Dynamic Allocation" in *z/OS SMP/E User's Guide*, or the section "DDDEF Entry (Distribution, Target, and Global Zone)" in *z/OS SMP/E Reference*.
- GIMDDALC control statements in SMPPARM member GIMDDALC. For more information, see the "Preparing to Use SMP/E" chapter in *z/OS SMP/E User's Guide*, or the "Defining Control Statements in SMPPARM members" chapter in *z/OS SMP/E Reference*.

SMP/E itself controls how the CSI and SMPPTS data sets are shared for concurrent background jobs by:

- Using different types of access for different types of processing
- Dividing command processing into phases so each phase can use the correct type of access
- Using the system enqueue facility to obtain and release the data sets
- Providing special support for sharing the global zone and the SMPPTS

Types of access

Every SMP/E command needs access to the global zone. Some commands also require access to one or more other zones. In addition, commands may need different types of access. For example, LIST only reads data and can, therefore, share it with other LIST jobs. On the other hand, APPLY actually updates the target zone, and, therefore, needs exclusive use of that data set to ensure data set integrity.

To meet the needs of all the commands, SMP/E supports three types of access:

- **Read access with no control of the zone**

  This access is required primarily during initialization for each command. For example, SMP/E might check the ZONEINDEX entries in the global zone to obtain the data set name for the target zone for the APPLY command. With this type of access, however, the data in the zone can be changed while SMP/E is checking it.

  For this type of access no enqueue is issued. It is, therefore, called *read without enqueue*.

- **Read access with shared control of the zone**

  This access is required during certain phases of processing to ensure that the data being checked is not being changed by another command. For example, when selecting SYSMODs to be applied, SMP/E must ensure that SYSMODs are not being received or rejected at the same time.

  To gain this type of access, SMP/E issues a shared enqueue on the data set where the zone resides. This type of access is called *read with shared enqueue*.

- **Update access with exclusive control of the zone**

  This access is required when the zone may be updated during command processing (as the target zone is during APPLY).

  To gain this type of access, SMP/E issues an exclusive enqueue on the data set where the zone resides.

This type of access is called *update with exclusive enqueue*.

Command processing phases

To avoid using the most restrictive type of access for the duration of a command, processing for each command is divided into phases. This way, SMP/E can obtain resources using the proper access type at the start of each phase, and free them at the end of each phase. This minimizes the amount of time a resource is tied up for exclusive use.

These are the important things to remember about command processing phases:

- Each command has at least two phases: initialization and termination. Most commands have one or more additional phases. For each command, the “Zone and Data Set Sharing Considerations” section of the chapter in this book devoted to that command describes the various phases of that command, which resources are required, and the type of access needed for that resource.
- During various phases of processing a command, SMP/E can use all three types of access for a given zone. For instance, during APPLY processing, the global zone is opened to obtain the name of the target zone CSI data set (read without enqueue), is then checked for which SYSMODs are eligible (read with shared enqueue), and is finally updated (update with exclusive enqueue).
- The most important use of command phases is to manage the global zone. This is important because each command must at least access the global zone to obtain further processing information, and most of the commands also update the global zone at some time during processing. Without command phases, almost all background SMP/E jobs would be serialized on the CSI data set containing the global zone.

Using the system enqueue facility

SMP/E uses the system enqueue facility for most of its processing and provides its own support to control shared use of the global zone and SMPPTS.

At the start of each phase, SMP/E tries to obtain each required data set. It does this by issuing ENQ with the SYSTEMS parameter. This enables you to control access to the data sets across multiple systems. The name used for the ENQ is `GIMSMP.`*dsname*, where:

GIMSMP
indicates that SMP/E issued the ENQ.

dsname
is the name of the CSI data set containing the required zone.

You should use this information if you are using Global Resource Serialization (GRS) to ensure that the data set is not updated by multiple systems simultaneously.

If the required data sets are being used by another SMP/E job, the enqueue request fails, and SMP/E issues a dequeue request to free all data sets for which a successful enqueue was issued. It then retries the whole series of enqueue requests every 10 seconds until the resource is obtained. How long SMP/E continues this depends on the PROCESS value in the EXEC statement PARM field. You can specify either PROCESS=WAIT or PROCESS=END. If you do not specify PROCESS, the default is PROCESS=WAIT.

- **PROCESS=WAIT.** SMP/E waits for the required data set until it is obtained. Every 30 minutes, it sends a message to the system operator indicating that the job is still waiting for the data set.
- **PROCESS=END.** SMP/E waits 10 minutes for the data set. If the data set is still not available after that time, command processing ends with a return code of 12.

Once all data sets are available, that phase of the command can continue. At the completion of each phase, SMP/E issues a dequeue request for all data sets it requested and that are not required by the next phase of the command. SMP/E then starts the next command phase by requesting the additional data sets required for that phase.

Note: Because SMP/E manages zone sharing at the CSI data set level, it cannot process concurrent changes to different zones that are on the same CSI. The first job to run obtains exclusive use of the entire

CSI data set. If you want to be able to process such concurrent changes, you should define the zones in different data sets. This is important to consider in determining how many CSI data sets you need and how to spread zones among them.

Sharing the global zone and SMPPTS data set

Because of the way SMP/E controls access to the various CSI data sets, only one user can update a data set at a time. However, the global zone and SMPPTS data sets are affected whenever SYSMODs are installed; either the zone name must be placed in the global zone SYSMOD entries, or the SMPPTS must be deleted. To avoid tying up these resources with requests for update access, SMP/E takes these actions:

- It records information in the zone being updated to reflect pending updates to the global zone and SMPPTS. Then, during one of the last phases in APPLY, ACCEPT, or RESTORE processing, SMP/E tries to make those pending changes to the global zone or SMPPTS. If either the global zone or SMPPTS is not available during this phase, SMP/E leaves the pending updates in the target zone or distribution zone.
- Whenever SMP/E opens a zone, the first thing it does is check for pending updates. If there are any, it tries to make the changes in the global zone and SMPPTS. If the global zone and SMPPTS are still not available, command processing continues, and the updates are left for the next access to that zone.
- Once the updates are made, SMP/E deletes the pending information from the target zone or distribution zone.

This processing allows multiple jobs to run concurrently against a common global zone and SMPPTS data set while protecting those data sets from concurrent updates and ensuring that all pending updates are made.

Appendix C. Building load modules

SMP/E often needs to build load modules during APPLY, LINK LMODS, LINK MODULE, and RESTORE command processing. After determining which load modules must be built, SMP/E verifies that all the modules needed to build the load modules are available. If a module is not found, SMP/E checks whether the target zone contains a MOD entry for that module. If it finds one, it checks whether the entry points to an existing LMOD entry and looks for these situations:

- If there is an LMOD entry and it contains a COPY indicator, SMP/E can include the module from the target library indicated in the LMOD entry.
- If the LMOD entry does not contain a COPY indicator, SMP/E checks whether the load module is a single-CSECT load module. If so, SMP/E can include the load module from the target library indicated in the LMOD entry.
- If the LMOD entry does not contain a COPY indicator and is not a single-CSECT load module, SMP/E assumes that the module has been assembled and looks for an ASSEM or SRC entry with the same name as the module. If SMP/E finds an ASSEM entry in the target zone, it assembles the entry and uses the output when it link-edits the load module. If SMP/E finds an SRC entry in the target zone, the SMPSTS data set, or the distribution library, it assembles the source and uses the output when it link-edits the load module.

If SMP/E could not find a usable copy of the module, SMP/E checks whether there is a DLIB entry corresponding to the distribution library for the module. If so, SMP/E checks whether the module is in the target library pointed to by the DLIB entry. If this is true, SMP/E uses that copy of the module when it link-edits the load module.

If a module is missing from the target zone or the target libraries and is not a new module, SMP/E checks for it in the distribution zone. If the FMID, RMID, and UMID in the distribution zone MOD entry match those in the target zone MOD entry, SMP/E uses the distribution library version of the module to build the load module.

If SMP/E does not find a required module by the previously described searches, it then tries to find a copy of the module within the SYSMOD that last replaced the module in the target system. If the target zone MOD entry for the module does not indicate that the module was assembled or updated since its last replacement (RMIDASM is not set and there are no UMIDs), SMP/E locates the SYSMOD identified by the RMID subentry in the SMPPTS data set. If the RMID SYSMOD is found in the SMPPTS data set, the ++MOD statement describing the required module is located within that SYSMOD. The ++MOD statement describes how the module is packaged and where the module is located:

- If the module is found inline within the SYSMOD, SMP/E copies the object module from the SMPPTS data set to the SMPWRK3 data set. Later, during the link-edit operation, the module is included from the SMPWRK3 data set.
- If the module is packaged in a RELFILE data set, SMP/E allocates the associated SMPTLIB data set that contains the module. Later, during the link-edit operation, the module is included from the SMPTLIB data set. If the SMPTLIB data set cannot be allocated, SMP/E issues messages to describe the allocation error and identify the modules that could not be found.
- If the module is packaged in either an LKLIB or TXLIB data set, SMP/E allocates the specified data set that contains the module. Later, during the link-edit operation, the module is included from the LKLIB or TXLIB data set. If the specified data set could not be allocated, SMP/E issues error messages to describe the allocation error and identify the modules that could not be found.

If after this search, the module is still not found, SMP/E takes one of these actions:

- If the module had been previously installed in the target system, but not in the distribution system, and no usable copy of the module could be found, then SMP/E issues error messages identifying each incomplete load module that was to include the identified module.
- If the module had been previously installed in the target system and distribution system, but the service level of the module in the distribution system does not match the service level of the module in the

target system, and no usable copy of the module could be found, then SMP/E issues error messages identifying each incomplete load module that was to include the identified module.

- If the module has not been previously installed in the target system or distribution system, then SMP/E issues a warning message for each incomplete load module that was to include the identified module. Processing continues, but the module is not included in the load module during the link-edit operation. SMP/E does not fail processing in this case because you can correct the problem by installing the product that supplies the required module. Once the product that supplies the needed module is applied, the module can be included in the incomplete load module.

Building load modules with CALLLIBS subentries

When building a load module with CALLLIBS, SMP/E may first need to link-edit the "base" version of the load module into the SMPLTS data set before link-editing the load module into its true target libraries:

- If the UPGLEVEL subentry exists for the set-to zone, a load module should exist in the SMPLTS data set only if it has both CALLLIBS subentries and an XZMOD subentry, indicating that at least one cross-zone module was linked into the load module. Therefore:
  - For a load module with CALLLIBS, but with no XZMOD subentries, SMP/E rebuilds the load module from scratch and links the rebuilt load module into its true system libraries by passing the CALL parameter with a SYSLIB data set allocated using the load module's CALLLIBS subentries. Typically, SMP/E does not link-edit the load module into the SMPLTS, although it may need to create a temporary member in SMPLTS to resolve certain warning conditions identified by the binder. This temporary member (if created) is deleted from the SMPLTS after a successful link-edit into the target library.
  - For a load module with both CALLLIBS and XZMOD subentries, SMP/E first looks for the "base" version of the load module in the SMPLTS data set. If the load module does not exist in the SMPLTS, SMP/E rebuilds the "base" version of load module from scratch and saves it in the SMPLTS data set. SMP/E next links the load module into its true system libraries by including the "base" version from the SMPLTS and passing the CALL parameter with a SYSLIB data set allocated using the load module's CALLLIBS subentries.
- If no UPGLEVEL subentry exists for the set-to zone, SMP/E first looks for the "base" version of the load module in SMPLTS. If the load module does not exist in the SMPLTS data set, SMP/E rebuilds the load module from scratch and saves it in the SMPLTS data set. SMP/E next links the load module into its true system libraries by including the base version from the SMPLTS and passing the CALL parameter with a SYSLIB data set allocated using the load module's CALLLIBS subentries.

Appendix D. Accessibility

Accessible publications for this product are offered through IBM Documentation (www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com®/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, its affiliates, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

Product Number: 5650-ZOS

SA23-2275-50