z/OS Basic Skills Information Center

z/OS system installation and management

z/OS Basic Skills Information Center

z/OS system installation and management

Note

Before using this information and the product it supports, read the information in "Notices" on page 67.

This edition applies to z/OS (product number 5694-A01).

Contents

z/OS system management roles and tasks

The job of the z/OS® system programmer is very complex and requires skills in many aspects of the system. The role of the z/OS system programmer can vary from one installation to another, but usually includes installing, customizing, and maintaining the operating system. In most large z/OS installations, system programmers become specialists for only specific tasks.

Generally speaking, the system programmer is responsible for managing the mainframe hardware configuration, and installing, customizing, and maintaining the mainframe operating system. System programmers ensure that their installation's system and its services are available and operating to meet service level agreements. Installations with 24-hour, 7-day operations need to plan for minimal disruption of their operation activities.

The role of system programmer usually includes some degree of involvement in all of the following aspects of system operation shown in Figure 1.

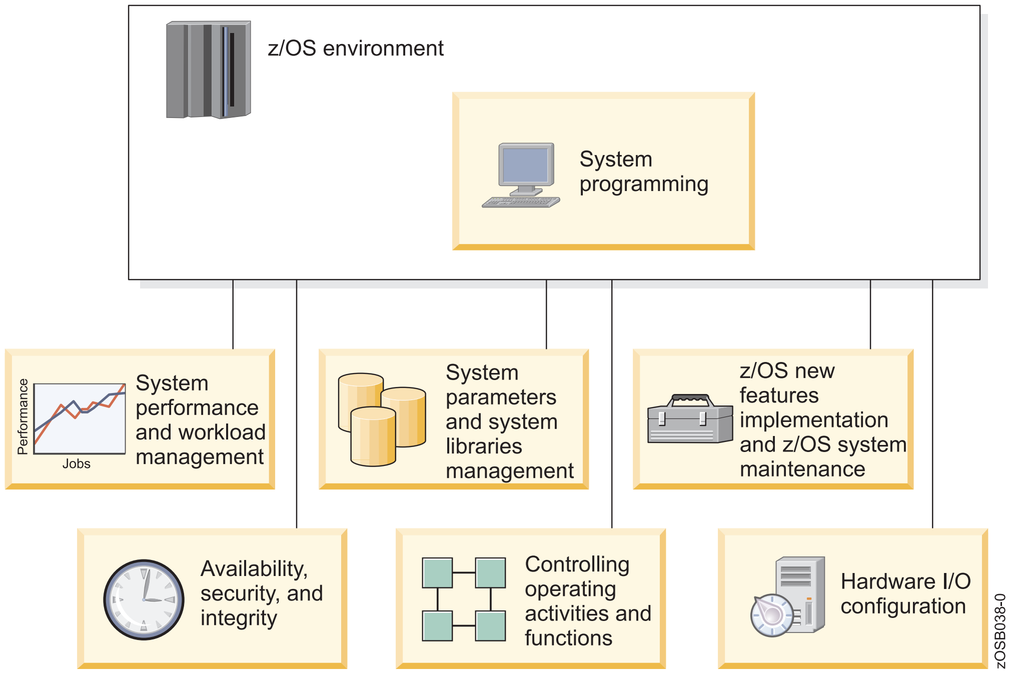

z/OS environment
System programming
Performance
Jobs
System performance and workload management
System parameters and system libraries management
z/OS new features implementation and z/OS system maintenance
Availability, security, and integrity
Controlling operating activities and functions
Hardware I/O configuration
zOSB038-0

Figure 1. Some areas in which the system programmer is involved

"Separation of duties" enables specialization

In a large z/OS installation, there is usually a separation of duties both among members of the system programming staff, and between the system programming department and other departments in the IT organization. A typical z/OS installation includes the following roles and more:

- z/OS system programmer
- CICS® system programmer
- Database system programmer
- Database administrator
- Network system programmer

- Automation specialist
- Security manager
- Hardware management
- Production control analyst
- System operator
- Network operator
- Security administrator
- Service manager

In part, the separation is an audit requirement– ensuring that one person does not have too much power on a system.

When a new application is to be added to a system, for example, a number of tasks need to be performed before the application can be used by end users. A production control analyst is needed to add batch applications into the batch scheduling package, add the new procedures to a procedure library, and set up the operational procedures. The system programmer is needed to perform tasks concerned with the system itself, such as setting up security privileges and adding programs to system libraries. The programmer is also involved with setting up any automation for the new application.

On a test system, however, a single person might have to perform all the roles, including being the operator, and this is often the best way to learn how everything works.

Part 1. Installation of z/OS and other software products

To install means to perform the tasks necessary to make the system operational, starting with a decision to either install for the first time or upgrade, and ending when the system is ready for production.

An installation plan is a record of the actions you need to take to install z/OS.

Chapter 1. z/OS base elements and optional features

The z/OS operating system consists of base elements and optional features.

- The base elements (or simply elements) deliver essential operating system functions. Base elements include:

  The Base Control Program (BCP)
  : The BCP provides essential operating system services. The BCP includes the I/O configuration program (IOCP), the workload manager (WLM), system management facilities (SMF), the z/OS UNIX® System Services (z/OS UNIX) kernel, the program management binder, and other components.

  Common Information Model (CIM)
  : CIM is a standard data model for describing and accessing systems management data in heterogeneous environments. It allows system administrators to write applications that measure system resources in a network with different operating systems and hardware.

  Communications Server
  : Communications Server (also known as CS z/OS) supports secure TCP/IP, SNA, and UNIX networking throughout an enterprise. It gives you the ability to connect subsystems and applications to each other, and to connect network devices (such as terminals and printers) to the system.

  Cryptographic Services
  : Cryptographic Services provides the following base cryptographic functions: data secrecy, data integrity, personal identification, digital signatures, and the management of cryptographic keys. Keys as long as 56 bits are supported by this base element.

  DFSMSdfp™
  : DFSMSdfp provides storage, data, program, and device management functions.

  Distributed File Service
  : Distributed File Service provides:
    - The DCE file serving (DFS(TM)) component of the Open Group Open Software Foundation (OSF) DCE. The file serving support (the DFS™ client and server) is at the OSF 1.2.2 level.
    - The zSeries® File System (zFS). The zFS is a UNIX file system that can be used in addition to the hierarchical file system (HFS). zFS file systems contain files and directories that can be accessed with the z/OS hierarchical file system file APIs. zFS file systems can be mounted into the z/OS UNIX hierarchy along with other local (or remote) file system types (such as HFS, TFS, AUTOMNT, and NFS). The zFS does not replace the HFS; it is complementary to the HFS.

  Hardware Configuration Definition (HCD)
  : HCD defines both the operating system configuration and the processor hardware configuration for a system.

  IBM® HTTP Server
  : IBM HTTP Server is the Web server for z/OS. It provides scalable, high performance Web serving for critical e-business applications. It supports

Secure Sockets Layer (SSL) secure connections, dynamic caching using the Fast Response Cache Accelerator, multiple IP addresses, proxy authentication, and double-byte character set characters.

Integrated Security Services
Integrated Security Services provides base security functions for z/OS. Its components include:
- DCE Security Server, which uses the limited DES algorithm for encryption.
- Enterprise Identity Mapping (EIM), which allows you to map a user's identity on one system to the user's identity on another system.
- Network Authentication Service, which uses the DES algorithm for encryption.

Interactive System Productivity Facility (ISPF)
ISPF provides facilities for all aspects of host-based software development. ISPF has four major components: Dialog Manager, Program Development Facility, Software Configuration and Library Manager, and the Client/Server component.

Job entry subsystem (JES)
z/OS installations may use one of two job entry subsystems; a job entry subsystem accepts the submission of work for the BCP.
- JES2 exercises independent control over its job processing functions.
- JES3 exercises centralized control.

JES2 is a base element of z/OS; JES3 is an optional feature.

Language Environment®
Language Environment provides the run-time environment for programs generated with C, C++, COBOL, Fortran, and PL/I.

Network File System (NFS)
NFS acts as a file server to workstations, personal computers, or other authorized systems in a TCP/IP network.

System Modification Program Extended (SMP/E)
SMP/E is a tool for installing and maintaining software, and for managing the inventory of software that has been installed.

Time Sharing Option/Extensions (TSO/E)
TSO/E allows users to create an interactive session with the z/OS system. TSO provides a single-user logon capability and a basic command prompt interface to z/OS.

z/OS UNIX System Services (z/OS UNIX)
z/OS UNIX provides the standard command interface familiar to interactive UNIX users.

- The optional features (or simply features) are orderable with z/OS and provide additional operating system functions. Optional features include:

DFSMSdss™
DFSMSdss copies and moves data for backup and recovery, and to reduce free-space fragmentation.

DFSMShsm™
DFSMShsm provides automated DASD storage management, including space management for low and inactive data, and availability management for accidental data loss caused by local and site disasters. DFSMShsm also lets you make effective use of tape media.

DFSMS™ Transactional VSAM Services (DFSMStvs)
DFSMStvs enables batch jobs and CICS online transactions to update shared VSAM data sets concurrently.

DFSORT™
DFSORT provides fast and easy sorting, merging, copying, reporting, and analysis of your business information, as well as versatile data handling at the record, field, and bit level.

Infoprint Server
Infoprint Server allows you to print files on z/OS printers from any workstation that has TCP/IP access.

Resource Measurement Facility (RMF™)
RMF gathers data about z/OS resource usage and provides reports at any system in a sysplex.

System Display and Search Facility (SDSF)
SDSF provides you with information to monitor, manage, and control your z/OS system.

Chapter 2. Methods of installing z/OS

Several IBM packages are available for installing z/OS. Some are entitled with the product (as part of your z/OS license, at no additional charge), while others are available for an additional fee.

The most commonly used methods of installation include:

ServerPac
: ServerPac is an entitled software delivery package consisting of products and service for which IBM has performed the SMP/E installation steps and some of the post-SMP/E installation steps. To install the package on your system and complete the installation of the software it includes, you use the CustomPac Installation Dialog.

Custom-Built Product Delivery Option (CBPDO)
: CBPDO is an entitled software delivery package consisting of uninstalled products and unintegrated service. There is no dialog program to help you install, as there is with ServerPac. You must use SMP/E to install the individual z/OS elements and features, and their service, before you can IPL.

SystemPac®
: SystemPac is a software package, available for an additional fee and offered worldwide, that helps you install z/OS, subsystems (DB2®, IMS™, CICS, NCP, and WebSphere® Application Server), and selected vendor products. SystemPac is tailored to your specifications; it is manufactured according to parameters and input/output definition file (IODF) definitions that you supply during order entry.

Chapter 3. Typical organization of IBM and other software in storage

Your installation's operating system environment consists of several different types of code and data– some supplied by IBM, some by other vendors, or some by your company's programmers– all of which combine to make your z/OS system unique. Once these software products and applications have been organized and stored, system programmers spend much of their time planning when to install updates or new software products, how to test them, and when to move them into production systems without adversely affecting workloads.

Figure 2 illustrates the different types of software code and data that exist in a system, and how they are usually organized in system storage.

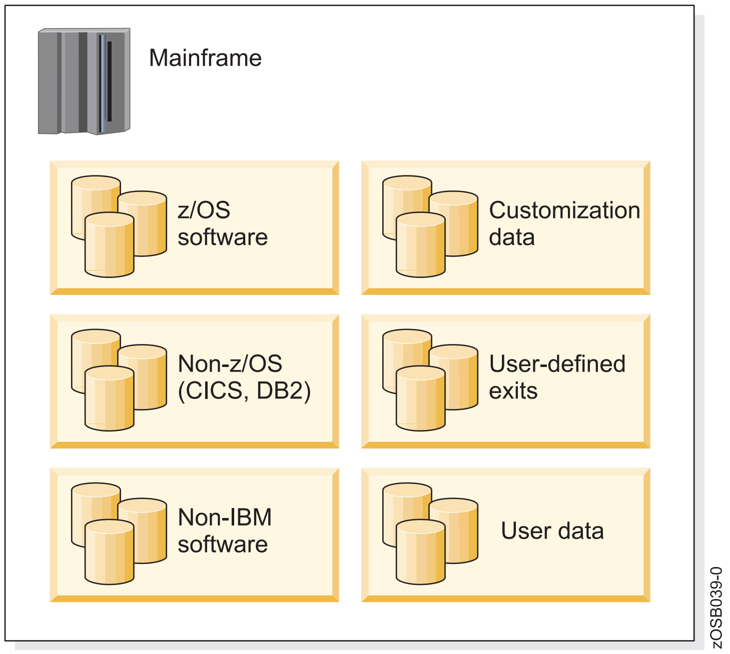


Figure 2. Typical organization of software

The z/OS software– as supplied by IBM– is usually installed on a series of disk volumes known as the system residence volumes (SYSRES). Much of the flexibility of z/OS is built on these SYSRES sets. They make it possible to apply maintenance to a new set that is cloned from the production set while the current set is running production work. A short outage can then be taken to IPL from the new set--and the maintenance has been implemented! Also, the change can be backed out by IPLing from the old set.

Fixes to z/OS are managed with a product called System Modification Program/Extended (SMP/E). Indirect cataloging using system symbols is used so that a particular library is cataloged as being on, for example, SYSRES volume 2, and the name of that volume is resolved by the system at IPL time from the system symbols.

Other IBM software (such as CICS and DB2), and non-IBM or third-party vendor software products are usually installed on another group of volumes, rather than on the SYSRES volumes. The SYSRES sets are usually managed as one entity by SMP/E, so their content is usually limited to z/OS software. The non-z/OS software is installed on as many volumes as are required, and thus can be managed separately.

Another group of volumes is reserved for customization data, which refers to data such as the z/OS system libraries (SYS1.PARMLIB and SYS1.PROCLIB, for example); the master catalog; the I/O definition file (IODF); page data sets; job entry subsystem (JES) spools; the /etc directory; and other items that are essential to the running of the z/OS system. It is also where SMP/E data is stored to manage the software.

These data sets are not always located on separate DASD volumes from IBM-supplied z/OS software; some installations place the PARMLIB and PROCLIB on the first SYSRES pack, others place them on the master catalog pack or elsewhere. This is a matter of choice and is dependent on how the SYSRES volumes are managed. Each installation will have a preferred method.

On many systems, some of the IBM-supplied defaults are not appropriate, so they need to be modified. User exits and user modifications (usermods) are made to IBM code so that it will behave as the installation requires. The modifications are usually managed using SMP/E.

Finally, another set of volumes contains production, test, and user data; this set is usually the largest pool of disk volumes. This set of volumes is not part of the system libraries, but is presented here for completeness. It is often split into pools and managed by System Managed Storage (SMS), which can target data to appropriately managed volumes. For example, production data can be placed on volumes that are backed up daily, whereas user data may only be captured weekly and may be migrated to tape after a short period of inactivity to free up the disk volumes for further data.

Part 2. z/OS system customization

System customization (also known as tailoring) is the overall process by which an installation selects its operating system. System programmers thoroughly plan and complete the steps in this process, selecting system options through several different mechanisms.

System customization is accomplished through the following mechanisms:

MVS™ hardware configuration definition (HCD)
System programmers use the HCD dialog to perform a variety of tasks, including defining the operating system and hardware configurations, activating configuration data (that is, applying configuration changes to the system), and querying or printing configuration data.

Initialization-time selections
When initializing the operating system, system programmers tailor the system environment through several sources, including operator actions, customization data in system libraries (SYS1.PARMLIB and other parmlib data sets), and job control language (JCL) for the master scheduler subsystem.

Implicit system parameters
Various system requirements affect the way the system performs. These system requirements may be considered as ″implicit″ parameters. They involve DD statements, data sets, hardware choices, and so forth. Some examples are:

- SYSABEND, SYSMDUMP, and SYSUDUMP DD statements.
- The System Management Facilities (SMF) data sets.
- Addition of new modules to the LPALST concatenation.
- Choice of the device on which the PLPA paging data sets will reside.
- Definition of page data sets through the DEFINE PAGESPACE command.

Chapter 4. z/OS system libraries

The z/OS system libraries contain customization data that is essential to the running of the z/OS system. Some of these data sets are related to IPL processing, while others are related to the search order of invoked programs or to system security, among other functions.

SYS1.PARMLIB

SYS1.PARMLIB is a required partitioned data set that contains control parameters for the whole system. PARMLIB is an important data set in a z/OS operating system, and can be thought of as performing a function similar to /etc on a UNIX system. The purpose of PARMLIB is to provide many control parameters in a pre-specified form in a single data set, thus minimizing the need for the operator to enter parameters during the initialization (IPL) process.

SYS1.PARMLIB must reside on a direct access volume, which can be the system residence volume. Parameters are specified in both IBM-supplied and installation-created members. All parameters and members of the SYS1.PARMLIB data set are described in *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

SYS1.LINKLIB

SYS1.LINKLIB contains many of the executable code, also known as modules, for z/OS components and utilities. By default, SYS1.LINKLIB is the first data set in the linklist, which is a collection of libraries containing system and user code.

One of the modules in SYS1.LINKLIB is MSTJCL00, which contains the initial job control language (JCL) statements that start the master scheduler subsystem. The master scheduler subsystem establishes communication between the operating system and the primary job entry subsystem, which is either JES2 or JES3.

System libraries for the link pack area (LPA)

The LPA is part of an address space's common area storage, and is divided into pageable, fixed, and modified sections:

- Libraries specified in SYS1.LPALIB, the LPALSTxx, or PROGxx parmlib members are loaded into pageable LPA (PLPA). These libraries contain modules for read-only system programs, along with any read-only reenterable user programs selected by an installation that can be shared among users of the system.
- IEAFIXxx members specify the modules loaded into fixed LPA (FLPA). This area should be used only for modules that significantly increase performance when they are fixed rather than pageable. The best candidates for the FLPA are modules that are infrequently used, but are needed for fast response. Modules placed in FLPA are always in central storage.
- IEALPAxx members specify the modules loaded into modified LPA (MLPA). The MLPA is used to contain reenterable routines from APF-authorized libraries that are to be part of the pageable extension to the link pack area during the current IPL. Note that the MLPA exists only for the duration of an IPL.

Link pack area (LPA) modules are loaded in common storage, and shared by all address spaces in the system. Because these modules are reentrant and are not self-modifying, each can be used by a number of tasks in any number of address spaces at the same time.

Modules placed anywhere in the LPA are always in virtual storage. To prevent their pages from being stolen, LPA modules must be referenced very often. When a page in LPA (other than in FLPA) is not continually referenced by multiple address spaces, it tends to be stolen.

SYS1.PROCLIB

SYS1.PROCLIB is a required partitioned data set that contains the IBM-supplied JCL procedures used to perform certain system functions. The JCL can be for system tasks or for processing program tasks invoked by the operator or the programmer.

One JCL procedure in SYS1.PROCLIB is the JES2 cataloged procedure, which defines job-related procedure libraries.

SYS1.NUCLEUS

SYS1.NUCLEUS contains the basic supervisor modules of the system.

Chapter 5. System symbols in PARMLIB

System symbols are elements that allow different z/OS systems to share PARMLIB definitions while retaining unique values in those definitions. System symbols act like variables in a program; they can take on different values, based on the input to the program.

When you specify a system symbol in a shared PARMLIB definition, the system symbol acts as a “placeholder”. Each system that shares the definition replaces the system symbol with a unique value during initialization.

Each system symbol has a name, which begins with an ampersand (&) and optionally ends with a period (.), and has substitution text, which is the character string that the system substitutes for a symbol each time it appears.

There are two types of system symbols:

Dynamic
: The substitution text can change at any point in an IPL.

Static
: The substitution text is defined at system initialization and remains fixed for the life of an IPL.

Some symbols are reserved for system use. You can display the symbols in your system by entering the D SYMBOLS command. Figure 3 shows the result of entering this command.

```
HQX7708 -----------------  SDSF PRIMARY OPTION MENU  --
COMMAND INPUT ===> -D SYMBOLS
  IEA007I STATIC SYSTEM SYMBOL VALUES
          &SYSALVL.  = "2"
          &SYSCLONE. = "70"
          &SYSNAME.  = "SC70"
          &SYSPLEX.  = "SANDBOX"
          &SYSR1.    = "Z17RC1"
          &ALLCLST1. = "CANCEL"
          &CMDLIST1. = "70,00"
          &COMMDSN1. = "COMMON"
          &DB2.      = "V8"
          &DCEPROC1. = "."
          &DFHSMCMD. = "00"
          &DFHSMHST. = "6"
          &DFHSMPRI. = "NO"
          &DFSPROC1. = "."
          &DLIB1.    = "Z17DL1"
          &DLIB2.    = "Z17DL2"
          &DLIB3.    = "Z17DL3"
          &DLIB4.    = "Z17DL4"
          &IEFSSNXX. = "R7"
          &IFAPRDXX. = "4A"
```

Figure 3. Partial output of the D SYMBOLS command (some lines removed)

The IEASYMxx PARMLIB member provides a single place to specify system parameters for each system in a multisystem environment. IEASYMxx contains statements that define static system symbols and that specify IEASYSxx PARMLIB members that contain system parameters (the SYSPARM statement). Figure 4 on page 16 page 16 shows an IEASYMxx PARMLIB member.

```
SYSDEF     SYSCLONE(&SYSNAME(3:2))
           SYMDEF(&SYSR2='&SYSR1(1:5).2')
           SYMDEF(&SYSR3='&SYSR1(1:5).3')
           SYMDEF(&DLIB1='&SYSR1(1:3).DL1')
           SYMDEF(&DLIB2='&SYSR1(1:3).DL2')
           SYMDEF(&DLIB3='&SYSR1(1:3).DL3')
           SYMDEF(&DLIB4='&SYSR1(1:3).DL4')
           SYMDEF(&ALLCLST1='CANCEL')
           SYMDEF(&CMDLIST1='&SYSCLONE.,00')
           SYMDEF(&COMMDSN1='COMMON')
           SYMDEF(&DFHSMCMD='00')
           SYMDEF(&IFAPRDXX='00')
           SYMDEF(&DCEPROC1='.')
           SYMDEF(&DFSPROC1='.')
SYSDEF     HWNAME(SCZP901)
           LPARNAME(A13)
           SYSNAME(SC70)
           SYSPARM(R3,70)
           SYMDEF(&IFAPRDXX='4A')
           SYMDEF(&DFHSMHST='6')
           SYMDEF(&DFHSMPRI='NO')
           SYMDEF(&DB2='V8')
```

Figure 4. Partial IEASYMxx PARMLIB member (some lines removed)

In the example, the variable &SYSNAME will have the value specified by the SYSNAME keyword; SC70 in this case. Because each system in a sysplex has a unique name, you can use &SYSNAME in the specification of system-unique resources, where permitted. As an example, you could specify the name of an SMF data set as SYS1.&SYSNAME..MAN1, with substitution resulting in the name SYS1.SC70.MAN1 when running on SC70.

You can use variables to construct the values of other variables. In Figure 4, &SYSCLONE takes on the value of &SYSNAME beginning at position 3 for a length of 2. Here, &SYSCLONE will have a value of 70. Similarly, &SYSR2 is constructed from the first 5 positions of &SYSR1 with a suffix of 2. Where is &SYSR1 defined? &SYSR1 is system-defined with the VOLSER of the IPL volume. If you refer back to Figure 3 on page 15, you will see the values of &SYSR1 and &SYSR2.

Figure 4 also shows the definition of a global variable defined to all systems--&IFAPRDXX with a value of 00--and its redefinition for SC70 to a value of 4A.

System symbols are used in cases where multiple z/OS systems share a single PARMLIB. The use of symbols allows individual members to be used with symbolic substitution, as opposed to having each system require a unique member. The LOADxx member specifies the IEASYMxx member that the system is to use.

Chapter 6. Search order for programs

Modules (programs), whether stored as load modules or program objects, must be loaded into both virtual storage and central storage before they can be run. When one module calls another module, either directly by asking for it to be run or indirectly by requesting a system service that uses it, it does not begin to run instantly. How long it takes before a requested module begins to run depends on where in its search order the system finds a usable copy and on how long it takes the system to make the copy it finds available.

When a program is requested through a system service (like LINK, LOAD, XCTL, or ATTACH) using default options, the system searches for it in the following sequence:

1. Job pack area (JPA)

   A program in JPA has already been loaded in the requesting address space. If the copy in JPA can be used, it will be used. Otherwise, the system either searches for a new copy or defers the request until the copy in JPA becomes available. (For example, the system defers a request until a previous caller is finished before reusing a serially-reusable module that is already in JPA.)
2. TASKLIB

   A program can allocate one or more data sets to a TASKLIB concatenation. Modules loaded by unauthorized tasks that are found in TASKLIB must be brought into private area virtual storage before they can run. Modules that have previously been loaded in common area virtual storage (LPA modules or those loaded by an authorized program into CSA) must be loaded into common area virtual storage before they can run.
3. STEPLIB or JOBLIB

   These are specific DD names that can be used to allocate data sets to be searched ahead of the default system search order for programs. Data sets can be allocated to both the STEPLIB and JOBLIB concatenations in JCL or by a program using dynamic allocation. However, only one or the other will be searched for modules. If both STEPLIB and JOBLIB are allocated for a particular jobstep, the system searches STEPLIB and ignores JOBLIB.

   Any data sets concatenated to STEPLIB or JOBLIB will be searched after any TASKLIB but before LPA. Modules found in STEPLIB or JOBLIB must be brought into private area virtual storage before they can run. Modules that have previously been loaded in common area virtual storage (LPA modules or those loaded by an authorized program into CSA) must be loaded into common area virtual storage before they can run.
4. LPA, which is searched in this order:
   a. Dynamic LPA modules, as specified in PROGxx members
   b. Fixed LPA (FLPA) modules, as specified in IEAFIXxx members
   c. Modified LPA (MLPA) modules, as specified in IEALPAxx members
   d. Pageable LPA (PLPA) modules, loaded from libraries specified in LPALSTxx or PROGxx

   LPA modules are loaded in common storage, shared by all address spaces in the system. Because these modules are reentrant and are not self-modifying, each can be used by any number of tasks in any number of address spaces at the same time. Modules found in LPA do not need to be brought into virtual storage, because they are already in virtual storage.

5. Libraries in the linklist, as specified in PROGxx and LNKLSTxx

   By default, the linklist begins with SYS1.LINKLIB, SYS1.MIGLIB, and SYS1.CSSLIB. However, you can change this order using SYSLIB in PROGxx and add other libraries to the linklist concatenation. The system must bring modules found in the linklist into private area virtual storage before the programs can run.

The default search order can be changed by specifying certain options on the macros used to call programs. The parameters that affect the search order the system will use are EP, EPLOC, DE, DCB, and TASKLIB. Some IBM subsystems (notably CICS and IMS) and applications (such as ISPF) use these facilities to establish other search orders for programs.

Chapter 7. Input/output (I/O) device configuration

The I/O configurations to the operating system (software) and the channel subsystem (hardware) must be defined. The Hardware Configuration Definition (HCD) component of z/OS consolidates the hardware and software I/O configuration processes under a single interactive end-user interface.

The output of HCD is an I/O definition file (IODF), which contains I/O configuration data. An IODF is used to define multiple hardware and software configurations to the z/OS operating system.

When a new IODF is activated, HCD defines the I/O configuration to the channel subsystem or the operating system, or both. With the HCD activate function or the z/OS ACTIVATE operator command, changes can be made in the current configuration without having to initial program load (IPL) the software or power-on reset (POR) the hardware. Making changes while the system is running is known as *dynamic configuration* or *dynamic reconfiguration*.

Chapter 8. Console configuration

Operating z/OS involves managing hardware such as processors and peripheral devices (including the consoles where your operators do their work); and software such as the z/OS operating control system, the job entry subsystem, subsystems (such as NetView®) that can control automated operations, and all the applications that run on z/OS.

The operation of a z/OS system involves the following:

- Message and command processing that forms the basis of operator interaction with z/OS and the basis of z/OS automation
- Console operations, or how operators interact with z/OS to monitor or control the hardware and software

Planning z/OS operations for a system must take into account how operators use consoles to do their work and how to manage messages and commands. The system programmer needs to ensure that operators receive the necessary messages at their consoles to perform their tasks, and select the proper messages for suppression, automation, or other kinds of message processing.

In terms of z/OS operations, how the installation establishes console recovery or whether an operator must re-IPL a system to change processing options are important planning considerations.

Because messages are also the basis for automated operations, the system programmer needs to understand message processing to plan z/OS automation.

As more installations make use of multisystem environments, the need to coordinate the operating activities of those systems becomes crucial. Even for single z/OS systems, an installation needs to think about controlling communication between functional areas.

In both single and multisystem environments, the commands that operators can enter from consoles can be a security concern that requires careful coordination. As a planner, the system programmer needs to make sure that the right people are doing the right tasks when they interact with z/OS.

A *console configuration* consists of the various consoles that operators use to communicate with z/OS. Your installation first defines the I/O devices it can use as consoles through the Hardware Configuration Definition (HCD), an interactive interface on the host that allows the system programmer to define the hardware configuration for both the channel subsystem and operating system.

Hardware Configuration Manager (HCM) is the graphical user interface to HCD. HCM interacts with HCD in a client/server relationship (that is, HCM runs on a workstation and HCD runs on the host). The host systems require an internal model of their connections to devices, but it can be more convenient and efficient for the system programmer to maintain (and supplement) that model in a visual form. HCM maintains the configuration data as a diagram in a file on the workstation in sync with the IODF on the host. While it is possible to use HCD directly for hardware configuration tasks, many customers prefer to use HCM exclusively, due to its graphical interface.

Besides HCD, Once the devices have been defined, z/OS is told which devices to use as consoles by specifying the appropriate device numbers in the CONSOLxx PARMLIB member.

Generally, operators on a z/OS system receive messages and enter commands on MCS and SMCS consoles. They can use other consoles (such as NetView consoles) to interact with z/OS, but here we describe the MCS, SMCS, and EMCS consoles as they are commonly used at z/OS sites:

- *Multiple Console Support (MCS) consoles* are devices that are locally attached to a z/OS system and provide the basic communication between operators and z/OS. MCS consoles are attached to control devices that do **not** support systems network architecture or SNA protocols.
- *SNA Multiple Console Support (SMCS) consoles* are devices that do not have to be locally attached to a z/OS system and provide the basic communication between operators and z/OS. SMCS consoles use z/OS Communications Server to provide communication between operators and z/OS, instead of direct I/O to the console device.
- *Extended Multiple Console Support (EMCS) consoles* are devices (other than MCS or SMCS consoles) from which operators or programs can enter commands and receive messages. Defining EMCS consoles as part of the console configuration allows the system programmer to extend the number of consoles beyond the MCS console limit, which is 99 for each z/OS system in a sysplex.

The system programmer defines these consoles in a configuration according to their functions. Important messages that require action can be directed to the operator, who can act by entering commands on the console. Another console can act as a monitor to display messages to an operator working in a functional area like a tape pool library, or to display messages about printers at your installation.

Figure 5 on page 23 shows a console configuration for a z/OS system that also includes the system console, an SMCS console, NetView, and TSO/E.

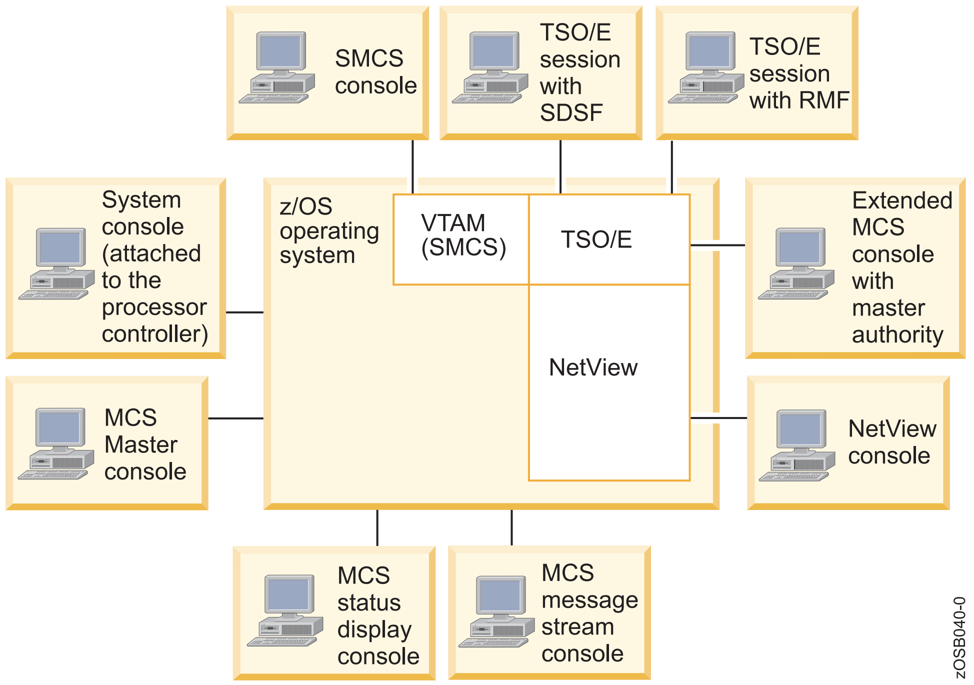

SMCS console
TSO/E session with SDSF
TSO/E session with RMF
System console (attached to the processor controller)
z/OS operating system
VTAM (SMCS)
TSO/E
NetView
Extended MCS console with master authority
MCS Master console
NetView console
MCS status display console
MCS message stream console
zOSB040-0

Figure 5. Sample console configuration for a z/OS system

The system console function is provided as part of the Hardware Management Console (HMC). An operator can use the system console to start up z/OS and other system software, and during recovery situations when other consoles are unavailable.

In addition to MCS and SMCS consoles, the z/OS system shown in Figure 5 has a NetView console defined to it. NetView works with system messages and command lists to help automate z/OS operator tasks. Many system operations can be controlled from a NetView console.

Users can monitor many z/OS system functions from TSO/E terminals. Using the System Display and Search Facility (SDSF) and the Resource Measurement Facility (RMF?), TSO/E users can monitor z/OS and respond to workload balancing and performance problems. An authorized TSO/E user can also initiate an extended MCS console session to interact with z/OS.

The MCS consoles shown in Figure 5 are:

- An MCS console from which an operator can view messages and enter z/OS commands

  This console is in full capability mode because it can receive messages and accept commands. An operator can control the operations for the z/OS system from an MCS or SMCS console.
- An MCS status display console

  An operator can view system status information from DEVSERV, DISPLAY, TRACK, or CONFIG commands. However, because this is a status display console, an operator cannot enter commands from the console. An operator on a full capability console can enter these commands and route the output to a status display console for viewing.
- An MCS message-stream console

A message-stream console can display system messages. An operator can view messages routed to this console. However, because this is a message-stream console, an operator cannot enter commands from here. Routing codes and message level information for the console are defined so that the system can direct relevant messages to the console screen for display. Thus, an operator who is responsible for a functional area like a tape pool library, for example, can view MOUNT messages.

In many installations, this proliferation of screens has been replaced by operator workstations that combine many of these screens onto one windowed display. Generally, the hardware console is separate, but most other terminals are combined. The systems are managed by alerts for exception conditions from the automation product.

The IBM Open Systems Adapter-Express Integrated Console Controller (OSA-ICC) is the modern way of connecting consoles. OSA-ICC uses TCP/IP connections over Ethernet LAN to attach to personal computers as consoles through a TN3270 connection (telnet).

Part 3. Starting z/OS: The initialization (IPL) process

How often do you start or reboot the operating system for your personal computer or notebook? Probably at least once a day, maybe more often, if you want to install upgrades or security patches. Depending on the work you are doing, you might reboot almost automatically, without doing more than a few save operations. Starting z/OS is, in a very general sense, a similar process, but is done far less frequently and only with much careful planning beforehand.

z/OS initialization, or an initial program load (IPL), is the act of loading a copy of the operating system from disk into the processor's real storage and executing it. This process essentially consists of:

- System and storage initialization, including the creation of system component address spaces
- Master scheduler initialization and subsystem initialization

z/OS systems are designed to run continuously with many months between reloads, allowing important production workloads to be continuously available. Change is the usual reason for a reload, and the level of change on a system dictates the reload schedule. For example:

- A test system may be IPLed daily or even more often.
- A high-availability banking system may only be reloaded once a year, or even less frequently, to refresh the software levels.
- Outside influences may often be the cause of IPLs, such as the need to test and maintain the power systems in the machine room.
- Sometimes badly behaved software uses up system resources that can only be replenished by an IPL, but this sort of behavior is normally the subject of investigation and correction.

Many of the changes that required an IPL in the past can now be done dynamically. Examples of these tasks are:

- Adding a library to the linklist for a subsystem such as CICS
- Adding modules to LPA

Shutting down z/OS happens as rarely as an IPL. To shut down the system, each task must be closed in turn, in the correct order. Today's z/OS installations use an automation package to control and execute this process. Shutting down the system usually requires a single command, which results in the removal of most tasks except for the automation task itself. The automation task is closed manually, followed by any commands needed to remove the system from a sysplex or serialization ring.

Chapter 9. System IPL: Sequence and key controls

The initialization process begins when the system programmer or operator selects the LOAD function at the Hardware Management Console (HMC).

To successfully IPL z/OS, the system programmer needs to supply the following information:
- The device address of the IPL volume
- The LOADxx member that contains pointers to system parameters
- The IODF data set that contains the configuration information
- The device address of the IODF volume

z/OS locates all usable central storage that is online and available, and begins creating the various system areas.

Not all disks attached to a CPU have loadable code on them. A disk that does is generally referred to as an "IPLable" disk, and more specifically as the SYSRES volume.

IPLable disks contain a bootstrap module at cylinder 0 track 0. At IPL, this bootstrap is loaded into storage at real address zero and control is passed to it. The bootstrap then reads the IPL control program IEAIPL00 (also known as IPL text) and passes control to it. This in turn starts the more complex task of loading the operating system and executing it.

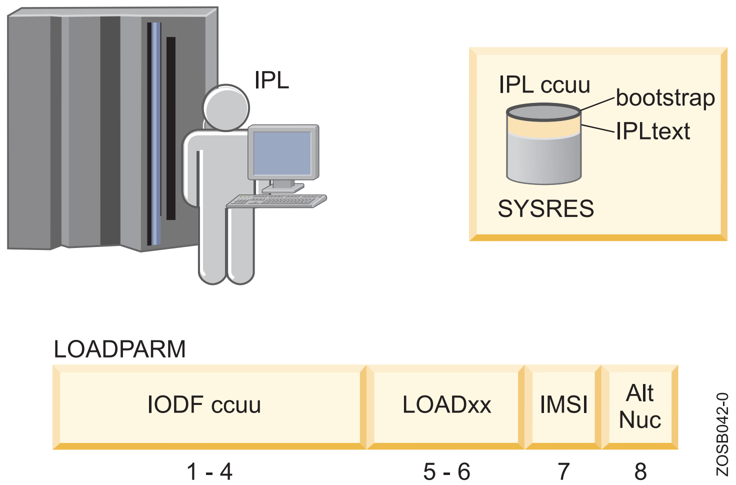


Figure 6. IPLing the machine

After the bootstrap is loaded and control is passed to IEAIPL00, IEAIPL00 prepares an environment suitable for starting the programs and modules that make up the operating system, as follows:
1. It clears central storage to zeros before defining storage areas for the master scheduler.
2. It locates the SYS1.NUCLEUS data set on the SYSRES volume and loads a series of programs from it known as IPL Resource Initialization Modules (IRIMs).

3. These IRIMs begin creating the normal operating system environment of control blocks and subsystems.

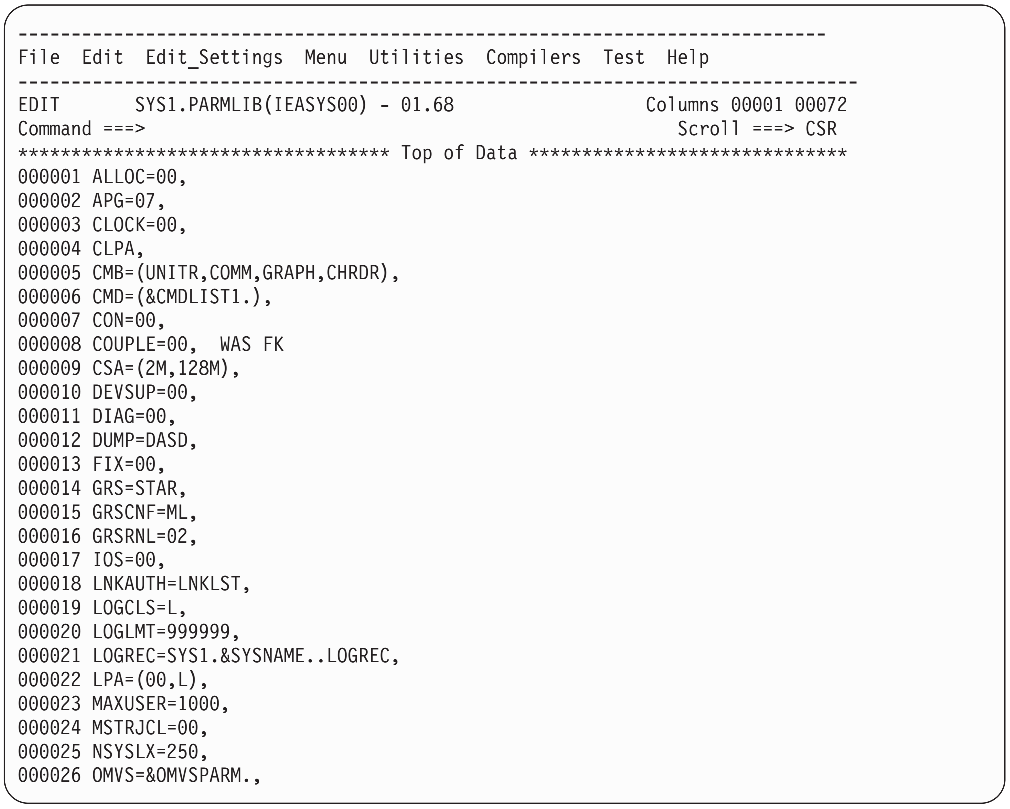
Some of the more significant tasks performed by the IRIMs are as follows:

- Read the LOADPARM information entered on the hardware console at the time the IPL command was executed.
- Search the volume specified in the LOADPARM member for the IODF data set. IRIM will first attempt to locate LOADxx in SYS0.IPLPARM. If this is unsuccessful, it will look for SYS1.IPLPARM, and so on, up to and including SYS9.IPLPARM. If at this point it still has not been located, the search continues in SYS1.PARMLIB. (If LOADxx cannot be located, the system loads a wait state.)
- If a LOADxx member is found, open and read information including the nucleus suffix (unless overridden in LOADPARM), the master catalog name, and the suffix of the IEASYSxx member to be used.
- Load the operating system's nucleus.
- Initialize virtual storage in the master scheduler address space for the System Queue Area (SQA), the Extended SQA (ESQA), the Local SQA (LSQA), and the Prefixed Save Area (PSA). At the end of the IPL sequence, the PSA will replace IEAIPL00 at real storage location zero, where it will then stay.
- Initialize real storage management, including the segment table for the master scheduler, segment table entries for common storage areas, and the page frame table.

The last of the IRIMs then loads the first part of the Nucleus Initialization Program (NIP), which invokes the Resource Initialization Modules (RIMs), one of the earliest of which starts up communications with the NIP console defined in the IODF.

During the NIP stage, the system might prompt the system programmer or operator to provide system parameters that control the operation of z/OS. The system also issues informational messages about the stages of the initialization process. IEASYSnn, a member of PARMLIB, contains parameters and pointers that control the direction that the IPL takes. The system programmer or operator may alter these parameters as necessary.

IEASYSnn, a member of PARMLIB, contains parameters and pointers that control the direction that the IPL takes. Figure 7 on page 29 illustrates partial content of an IEASYSxx member.



```
---------------------------------------------------------------------------
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       SYS1.PARMLIB(IEASYS00) - 01.68                 Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
*********************************** Top of Data ******************************
000001 ALLOC=00,
000002 APG=07,
000003 CLOCK=00,
000004 CLPA,
000005 CMB=(UNITR,COMM,GRAPH,CHRDR),
000006 CMD=(&CMDLIST1.),
000007 CON=00,
000008 COUPLE=00,  WAS FK
000009 CSA=(2M,128M),
000010 DEVSUP=00,
000011 DIAG=00,
000012 DUMP=DASD,
000013 FIX=00,
000014 GRS=STAR,
000015 GRSCNF=ML,
000016 GRSRNL=02,
000017 IOS=00,
000018 LNKAUTH=LNKLST,
000019 LOGCLS=L,
000020 LOGLMT=999999,
000021 LOGREC=SYS1.&SYSNAME..LOGREC,
000022 LPA=(00,L),
000023 MAXUSER=1000,
000024 MSTRJCL=00,
000025 NSYSLX=250,
000026 OMVS=&OMVSPARM.,
```

Figure 7. Partial listing of IEASYS00 member

The system continues the initialization process, interpreting and acting on the system parameters that were specified. NIP carries out the following major initialization functions:

- Expands the SQA and the extended SQA by the amounts specified on the SQA system parameter.
- Creates the pageable link pack area (PLPA) and the extended PLPA for a cold start IPL; resets tables to match an existing PLPA and extended PLPA for a quick start or a warm start IPL.
- Loads modules into the fixed link pack area (FLPA) or the extended FLPA. Note that NIP carries out this function only if the FIX system parameter is specified.
- Loads modules into the modified link pack area (MLPA) and the extended MLPA. Note that NIP carries out this function only if the MLPA system parameter is specified.
- Allocates virtual storage for the common service area (CSA) and the extended CSA. The amount of storage allocated depends on the values specified on the CSA system parameter at IPL.
- Page-protects the NUCMAP, PLPA and extended PLPA, MLPA and extended MLPA, FLPA and extended FLPA, and portions of the nucleus. An installation can override page protection of the MLPA and FLPA by specifying NOPROT on the MLPA and FIX system parameters.

To see information on how your system was IPLed, you can issue the D IPLINFO command, as Figure 8 on page 30 shows.

```
D IPLINFO
IEE254I  11.11.35  IPLINFO DISPLAY 906
 SYSTEM IPLED AT 10.53.04 ON 08/15/2007
 RELEASE z/OS 01.07.00    LICENSE = z/OS
 USED LOADS8 IN SYS0.IPLPARM ON C730
 ARCHLVL = 2  MTLSHARE = N
 IEASYM LIST = XX
 IEASYS LIST = (R3,65) (OP)
 IODF DEVICE C730
 IPL DEVICE 8603 VOLUME Z17RC1
```

Figure 8. Output of the D IPLINFO command

Chapter 10. System IPL: Address space creation and subsystem initialization

In addition to initializing system areas, z/OS establishes system component address spaces. It establishes an address space for the master scheduler and other system address spaces for various subsystems and system components. Some of the component address spaces are: *MASTER*, ALLOCAS, APPC, CATALOG, and so on.

The master scheduler address space is the first system component address space to be created (ASID=1). Then, the master scheduler may start the primary job entry subsystem (JES2 or JES3). On many production systems, JES is not started immediately; instead, an automation package starts all tasks in a controlled sequence.

Then other defined subsystems are started. All subsystems are defined in the PARMLIB library, in member IEFSSNxx. These subsystems are *secondary subsystems*.

Figure 9 shows some of the important system component address spaces; for started tasks VTAM®, CICS, TSO; and for a TSO user and a batch initiator. Each address space has 2 GB of virtual storage by default, whether the system is running in 31-bit or 64-bit mode.

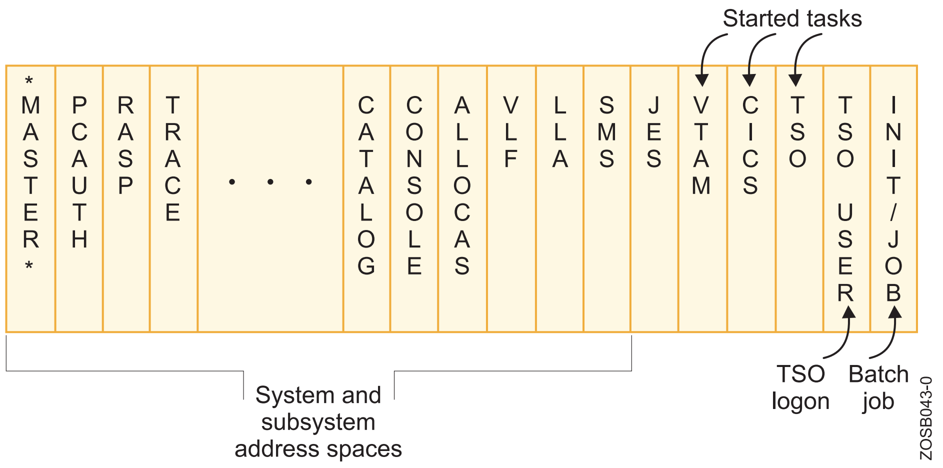


Figure 9. Virtual storage layout for multiple address spaces

The private areas are available only to that address space, but common areas are available to all.

After the system is initialized and the job entry subsystem is active, the installation can submit jobs for processing by using the START, LOGON, or MOUNT command. When a job is activated through START (for batch jobs), LOGON (for time-sharing jobs), or MOUNT, a new address space is allocated.

The master scheduler subsystem

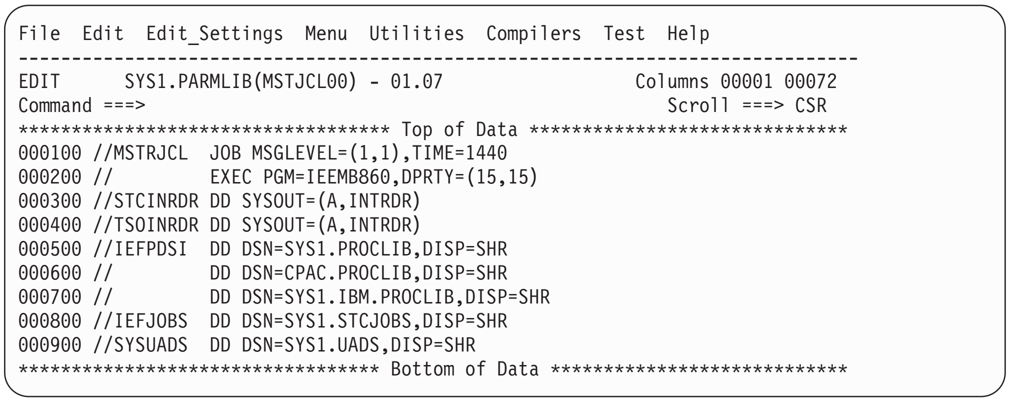
The master scheduler subsystem establishes communication between the operating system and the primary job entry subsystem, which can be JES2 or JES3.

To do its work, the master scheduler subsystem uses an MSTJCLxx member, commonly called *master JCL*, which contains data definition (DD) statements for all system input and output data sets that are needed to do the communication between the operating system and JES.

An initial MSTJCL00 load module can be found in the SYS1.LINKLIB library. As shipped, MSTJCL00 contains an IEFPDSI DD statement that defines the data set that contains procedure source JCL for started tasks. Normally this data set is SYS1.PROCLIB; it may be a concatenation. For useful work to be performed, SYS1.PROCLIB must at least contain the procedure for the primary JES.

If modifications are required, the recommended procedure is to create an MSTJCLxx member in the PARMLIB data set. The suffix is specified by the MSTRJCL parameter in the IEASYSxx member of PARMLIB.

Figure 10 shows a sample MSTJCLxx member.



```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       SYS1.PARMLIB(MSTJCL00) - 01.07                 Columns 00001 00072
Command ===>                                                  Scroll ===> CSR
*********************************** Top of Data ******************************
000100 //MSTRJCL  JOB MSGLEVEL=(1,1),TIME=1440
000200 //         EXEC PGM=IEEMB860,DPRTY=(15,15)
000300 //STCINRDR DD SYSOUT=(A,INTRDR)
000400 //TSOINRDR DD SYSOUT=(A,INTRDR)
000500 //IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR
000600 //         DD DSN=CPAC.PROCLIB,DISP=SHR
000700 //         DD DSN=SYS1.IBM.PROCLIB,DISP=SHR
000800 //IEFJOBS  DD DSN=SYS1.STCJOBS,DISP=SHR
000900 //SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
********************************* Bottom of Data ****************************
```

Figure 10. Sample master JCL

When the master scheduler has to process the start of a started task, the system determines whether the START command refers to a procedure or to a job. If the IEFJOBS DD exists in the MSTJCLxx member, the system searches the IEFJOBS DD concatenation for the member requested in the START command.

If there is no member by that name in the IEFJOBS concatenation, or if the IEFJOBS concatenation does not exist, the system searches the IEFPDSI DD for the member requested in the START command. If a member is found, the system examines the first record for a valid JOB statement and, if one exists, uses the member as the JCL source for the started task. If the member does not have a valid JOB statement in its first record, the system assumes that the source JCL is a procedure and creates JCL to invoke the procedure. After the JCL source has been created (or found), the system processes the JCL.

As shipped, MSTJCL00 contains an IEFPDSI DD statement that defines the data set that contains procedure source JCL for started tasks. Normally this data set is SYS1.PROCLIB; it may be a concatenation. For useful work to be performed, SYS1.PROCLIB must at least contain the procedure for the primary JES.

The job entry subsystem (JES)

For useful work to be performed on z/OS, SYS1.PROCLIB must contain a job procedure for the primary job entry subsystem.

To initialize JES, the master scheduler subsystem looks in SYS1.PROCLIB for the JES cataloged procedure. This procedure defines the job-related procedure libraries, including those that the JES subsystem uses to locate the JES initialization statements. A sample job procedure library is shown in Figure 11.

```
//PROC00 DD  DSN=SYS1.PROCLIB,DISP=SHR
//       DD  DSN=SYS3.PROD.PROCLIB,DISP=SHR
//PROC01 DD  DSN=SYS1.PROC2,DISP=SHR
...
//PROC99 DD  DSN=SYS1.LASTPROC,DISP=SHR
...
```

Figure 11. Job procedure libraries in the JES2 procedure

Many installations have very long lists of procedure libraries in the JES procedure. Care should be taken as to the number of users who can delete these libraries because JES will not start if one is missing. Normally a library that is in use cannot be deleted, but JES does not hold these libraries although it uses them all the time.

You can override the default specification by specifying this statement: `/*JOBPARM PROCLIB=`

After the name of the procedure library, you code the name of the DD statement in the JES2 procedure that points to the library to be used. For example, in Figure 11, assume that you run a job in class A and that class has a default PROCLIB specification on PROC00. If you want to use a procedure that resides in SYS1.LASTPROC, you'll need to include this statement in the JCL:`/*JOBPARM PROCLIB=PROC99`

Another way to specify a procedure library is to use the JCLLIB JCL statement. This statement allows you to code and use procedures without using system procedure libraries. The system searches the libraries in the order in which you specify them on the JCLLIB statement, prior to searching any unspecified default system procedure libraries.

Figure 12 shows the use of the JCLLIB statement.

```
//MYJOB  JOB
//MYLIBS JCLLIB ORDER=(MY.PROCLIB.JCL,SECOND.PROCLIB.JCL)
//S1     EXEC PROC=MYPROC1
...
```

Figure 12. Sample JCLLIB statement

Assuming that the system default procedure library includes SYS1.PROCLIB only, the system searches the libraries for procedure MYPROC1 in the following order:
1. MY.PROCLIB.JCL
2. SECOND.PROCLIB.JCL
3. SYS1.PROCLIB

Initialization of additional subsystems

Subsystem initialization is the process of readying a subsystem for use in the system. IEFSSNxx members of SYS1.PARMLIB contain the definitions for the primary subsystems such as JES2 or JES3, and the secondary subsystems such as NetView and DB2.

During system initialization, the defined subsystems are initialized. You should define the primary subsystem (JES) first because other subsystems, such as DB2, require the services of the primary subsystem in their initialization routines. Problems can occur if subsystems that use the subsystem affinity service in their initialization routines are initialized before the primary subsystem. After the primary JES is initialized, the subsystems are initialized in the order in which the IEFSSNxx PARMLIB members are specified by the SSN parameter. For example, for SSN=(aa,bb), PARMLIB member IEFSSNaa would be processed before IEFSSNbb.

Chapter 11. System IPL: Description of IPL types

Depending on the circumstances under which you want to start or restart z/OS, you may use one of three types of initialization: Cold start, quick start, or warm start.

The primary difference between the three types is whether or not certain storage is treated.

Cold start
: An IPL that loads (or reloads) the pageable link pack area (PLPA) and clears the virtual input/output (VIO) data set pages.

  The first IPL after system installation is always a cold start because the PLPA is initially loaded. Subsequent IPLs are cold starts when the PLPA is reloaded, either to alter its contents or to restore its contents if they were lost. This is usually done when changes have been made to the LPA (for example, when a new SYSRES containing maintenance is being loaded).

  VIO is a method of using memory to store small temporary data sets for rapid access. However, unlike a RAM disk on a PC, these are actually backed up to disk and so can be used as a restart point. Obviously there should not be too much data stored in this way, so the size is restricted.

Quick start
: An IPL that does not reload the PLPA, but clears the VIO data set pages. (The system resets the page and segment tables to match the last-created PLPA.) This is usually done when there have been no changes to LPA, but VIO must be refreshed. This prevents the warm start of jobs that were using VIO data sets.

Warm start
: An IPL that does not reload the PLPA, and preserves journaled VIO data set pages. This will allow jobs that were running at the time of the IPL to restart with their journaled VIO data sets.

Often, the preferred approach is to do a cold start IPL (specifying CLPA). The other options can be used, but extreme care must be taken to avoid unexpected change or backout of change. A warm start could be used when you have long-running jobs which you want to restart after IPL, but an alternative approach is to break down those jobs into smaller pieces which pass real data sets rather than use VIO. Modern disk controllers with large cache memory have reduced the need for VIO data to be kept for long periods.

Also, do not confuse a cold start IPL (CLPA would normally be used rather than the term ″cold start″) with a JES cold start. Cold starting JES is something that is done extremely rarely, if ever, on a production system, and totally destroys the existing data in JES.

Part 4. z/OS system tuning

The task of ″tuning″ a system is an iterative and continuous process, and it is the discipline that most directly impacts all users of system resources in an enterprise. The z/OS Workload Management (WLM) component is an important part of this process and includes initial tuning of selecting appropriate parameters for various system components and subsystems.

After the system is operational and criteria have been established for the selection of jobs for execution through job classes and priorities, WLM controls the distribution of available resources according to the parameters specified by the installation.

WLM, however, can deal with only available resources. If these are inadequate to meet the needs of the installation, even optimal distribution may not be the answer; other areas of the system should be examined to determine the possibility of increasing available resources. When requirements for the system increase and it becomes necessary to shift priorities or acquire additional resources (such as a larger processor, more storage, or more terminals), the system programmer needs to modify WLM parameters to reflect changed conditions.

Part 5. z/OS software maintenance

As a z/OS system programmer, it will be your responsibility to ensure that all software products and their modifications are properly installed on the system. You will also have to ensure that all products are installed at the proper level so that the elements of the system can work together.

At first, that might not sound too difficult, but as the complexity of the software configuration increases, so, too, does the task of monitoring all the elements of the system. Data center management is typically held accountable for Service Level Agreements (SLAs), often through a specialist team of service managers. Change control mechanics and practices in a data center are implemented to ensure that SLAs are met.

System Modification Program Extended (SMP/E) is the primary means of installing and updating the software in a z/OS system. SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets. SMP/E can be run either using batch jobs or using dialogs under Interactive System Productivity Facility/Program Development Facility (ISPF/PDF). With SMP/E dialogs, you can interactively query the SMP/E database and create and submit jobs to process SMP/E commands.

Software to be installed by SMP/E must be packaged as system modifications or SYSMODs, which combine the updated element with control information. This information describes the elements and any relationships the software has with other products or service that may also be installed on the same system.

The SMP/E job control language (JCL) and commands will be used frequently by a large enterprise z/OS system programmer, however, SMP/E modification control statement (MCS) instructions will rarely be coded by the same system programmer. The product and SYSMOD packaging will include the necessary MCS statements.

A critical responsibility of the system programmer is to work with IBM defect support when a problem surfaces in z/OS or optional IBM products. Problem resolution requires the system programmer to receive and apply fixes to the enterprise system.

Chapter 12. z/OS conventions: Following a process of change control

One of the advantages of the mainframe is the very high availability that it offers; therefore, all change must therefore be carefully controlled and managed. A high proportion of any system programmer's time is involved in the planning and risk assessment of change. One of the most important aspects of change is how to reverse it and go back to the previous state.

The implementation of any change must be under the control of the Operations staff. When a change is introduced into a production environment that results in problems or instability, Operations staff are responsible for observing, reporting, and then managing the activities required to correct the problem or back out the change.

Although system programmers will normally originate and implement their own changes, sometimes changes are based on a request through the change management system. Any instructions for Operations or other groups would be in the change record, and the approval of each group is required.

Implementing business application changes would normally be handled by a production control analyst. Application changes will normally reside in test libraries, and an official request (with audit trail) would result in the programs in the test libraries being promoted to the production environment.

Procedures involved in the change must be circulated to all interested parties. When all parties consider the change description to be complete, then it is considered for implementation and either scheduled, deferred, or possibly rejected.

The factors that need to be considered when planning a change are:
- The benefits that will result from the change
- What will happen if the change is not done
- The resources required to implement the change
- The relative importance of the change request compared to others
- Any interdependency of change requests

Risk assessment

It is common practice for data center management to have a weekly change control meeting to discuss, approve, or reject changes. These changes might be for applications, a system, a network, hardware, or power.

An important part of any change is *risk assessment*, in which the change is considered and evaluated from the point of view of risk to the system. Low risk changes may be permitted during the day, while higher risk changes would be scheduled for an outage slot.

It is also common practice for a data center to have periods of low and high risk, which will influence decisions. For example, if the system runs credit authorizations, then the periods around major public holidays are usually extremely busy and may cause a change freeze. Also, annual sales are extremely busy periods in retailing and may cause changes to be rejected.

IT organizations achieve their goals through disciplined change management processes and policy enforcement. These goals include:
- High service availability
- Increased security
- Audit readiness
- Cost savings

Change control record system

A *change control record system* is typically in place to allow for the requesting, tracking, and approval of changes. This is usually the partner of a *problem management system*. For example, if a production system has a serious problem on a Monday morning, then one of the first actions will be to examine the changes that were implemented over the weekend to determine if these have any bearing on the problem.

These records also show that the system is under control, which is often necessary to prove to auditors, especially in the heavily regulated financial services sector. The Sarbanes-Oxley Act of 2002 in the United States, which addresses corporate governance, has established the need for an effective internal control system. Demonstrating strong change management and problem management in IT services is part of compliance with this measure. Additionally, the 8th Directive on Company Law in the European Union addresses similar areas to Sarbanes-Oxley.

For these reasons, and at a bare minimum, before any change is implemented, there should be a set of controlled documents defined, which are known as *change request forms*. These should include the following:
- Who is responsible for implementing the change, completing the successful test and responsible for backout if required. Also who will ″sign off″ the change as successful. This is usually a department, group or person that requires the change.
- What are the affected systems or services (for example e-mail, file service, domain, and so on). Include as much detail as possible. Ideally, complete instructions should be included so that the change could be performed by someone else in an emergency.
- The start date and time and estimated duration of the change. There are often three dates: requested, scheduled. and actual.
- The scope of change, the business units, buildings, departments or groups affected or required to assist with the change.
- The implementation plan and a plan for backing off the changes, if the need arises.
- The priority of the change; that is, high, medium, low, business as usual, emergency, dated (for example clock change).
- The risk, which is usually classified as high, medium, or low.
- The impact if the change is implemented; what will happen if it is not; what other systems may be affected; what will happen if something unexpected occurs.

Production control

Production control usually involves a specialized staff to manage batch scheduling, using a tool such as Tivoli® Workload Scheduler to build and manage a complex batch schedule. This work might involve daily and weekly backups running at particular points within a complex sequence of application suites. Databases and online services might also be taken down and brought back up as part of the schedule. While making such changes, production control often needs to accommodate public holidays and other special events such as (in the case of a retail sales business) a winter sale.

Production control is also responsible for taking a programmer's latest program and releasing it to production. This task typically involves moving the source code to a secure production library, recompiling the code to produce a production load module, and placing that module in a production load library. JCL is copied and updated to production standards and placed in appropriate procedure libraries, and application suites added to the job scheduler.

There might also be an interaction with the system programmer if a new library needs to be added to the linklist, or authorized.

Chapter 13. System installation and maintenance using SMP/E

System Modification Program Extended (SMP/E) is the z/OS tool for managing the installation of software products on a z/OS system and for tracking modifications to those products.

SMP/E controls these changes at the component level by:
- Selecting the proper levels of code to be installed from a large number of potential changes.
- Calling system utility programs to install the changes.
- Keeping records of the installed changes by providing a facility to enable you to inquire on the status of your software, and to reverse the change if necessary.

All code and its modifications are located in the SMP/E database called the consolidated software inventory (CSI), which is comprised of one or more VSAM data sets.

SMP/E can be run either using batch jobs or using dialogs under ISPF/PDF. With SMP/E dialogs, you can interactively query the SMP/E database and create and submit jobs to process SMP/E commands.

The SMP/E view of the system

A z/OS system might appear to be one big block of code that drives the CPU. Actually, z/OS is a complex system comprising many different smaller blocks of code, and that's how SMP/E sees z/OS.

Each of those smaller blocks of code perform a specific function within the system. Functions that can appear in a z/OS system (some of which are shown in Figure 13 on page 46) include:
- Base Control Program (BCP)
- Job entry subsystem (JES2 or JES3)
- Time Sharing Option/Extensions (TSO/E)
- Interactive System Productivity Facility (ISPF)
- System Display and Search Facility (SDSF)
- Data Facility Storage Management Subsystem (DFSMS)
- System Modification Program Extended (SMP/E)
- z/OS UNIX System Services (z/OS UNIX)
- Resource Measurement Facility (RMF)
- HTTP Server
- DB2 (Database 2™)
- Customer Information Control System (CICS)
- WebSphere MQ

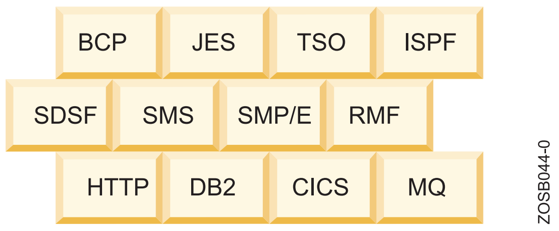


Figure 13. SMP/E view of the system

Each system function is composed of one or more load modules. In a z/OS environment, a load module represents the basic unit of machine-readable, executable code. Load modules are created by combining one or more object modules and processing them with a link-edit utility. The link-editing of modules is a process that resolves external references and addresses. The functions on your system, therefore, are one or more object modules that have been combined and link-edited.

To see where the object module comes from, look at the example in Figure 14.

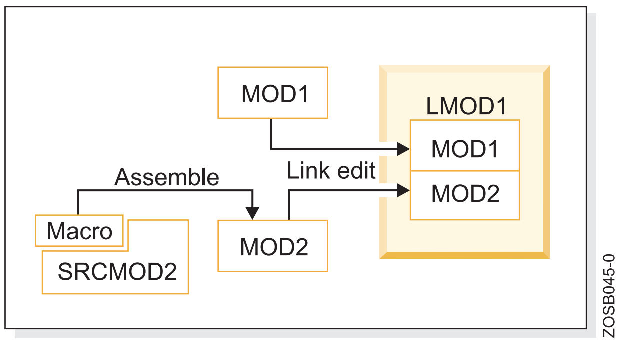


Figure 14. Load module creation

Most of the time, object modules are sent to you as part of a product. In this example, the object module MOD1 was sent as part of the product. Other times, you might need to assemble source code sent to you by product packagers to create the object module. You can modify the source code and then assemble it to produce an object module. In the example, SRCMOD2 is source code that you assemble to create object module MOD2. When assembled, you link-edit object module MOD2 with object module MOD1 to form the load module LMOD1.

In addition to object modules and source code, most products distribute many other parts, such as macros, help-panels, CLISTs and other z/OS library members. These modules, macros and other types of data and code are the basic building blocks of your system. All of these building blocks are called elements.

Elements are associated with, and depend upon, other products or services that may be installed on the same z/OS system. They describe the relationship the software has with other products or services that may be installed on the same z/OS system.

How the SMP/E environment is similar to a public library

To properly perform its processing, SMP/E must maintain a great deal of information about the structure, content, and modification status of the software it manages. Think of all the information SMP/E has to maintain as if it were all the information contained in the public library.

In a public library, you see bookshelves filled with books and a card catalog with drawers containing a card for each book in the library. These cards contain information, such as the title, author, publishing dates, type of book, and a pointer to the actual book on the shelf.

In the SMP/E environment, there are two distinct types of ″bookshelves.″ They are referred to as the distribution libraries and the target libraries. In much the same way the bookshelves in the public library hold the library books, the distribution and target libraries hold the elements of the system.

Distribution libraries contain all the elements, such as modules and macros, that are used as input for running your system. One very important use of the distribution libraries is for backup. Should a serious error occur with an element on the production system, the element can be replaced by a stable level found in the distribution libraries.

Target libraries contain the executable code needed to run your system (for example, the libraries from which you run your production system or your test system).

As you think of the analogy of the public library, you can see that there is one important piece of that picture that we have not yet considered. In the public library, there is a card catalog to help you find the book or piece of information you are looking for. SMP/E provides the same type of tracking mechanism in the form of the *consolidated software inventory* (CSI).

The CSI data sets contain all the information SMP/E needs to track the distribution and target libraries. As the card catalog contains a card for each book in the library, the CSI contains an entry for each element in its libraries. The CSI entries contain the element name, type, history, how the element was introduced into the system, and a pointer to the element in the distribution and target libraries. The CSI does not contain the element itself, but rather a description of the element it represents.

The cards in the public library card catalog are arranged alphabetically by the author's last name, and by the topic and title of the book. In the CSI, entries for the elements in the distribution and target libraries are grouped according to their installation status. That is, entries representing elements found in the distribution libraries are contained in the distribution zone. Entries representing elements found in the target libraries are contained in the target zone. Both of these zones serve the same purpose as the drawers of the public library card catalog.

In addition to the distribution and target zones, the SMP/E CSI also contains a global zone. Figure 15 on page 48 shows the relationship between SMPE zones and libraries.

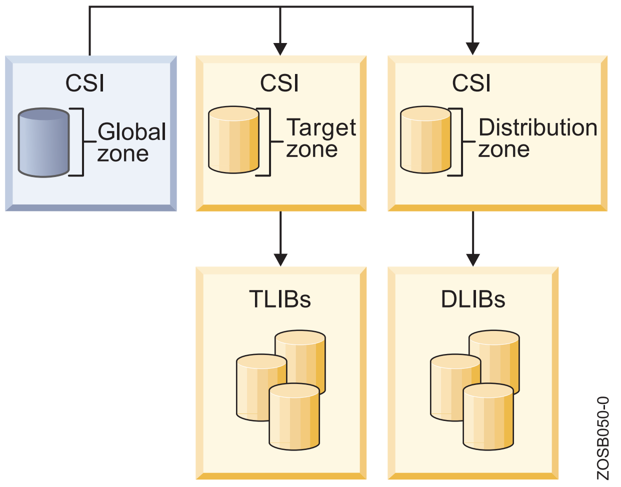


Figure 15. Relationship between SMP/E zones and libraries

The global zone contains:

- Entries needed to identify and describe each target and distribution zone to SMP/E
- Information about SMP/E processing options
- Status information for all system modifications (known as SYSMODs) that SMP/E has begun to process
- Exception data for SYSMODs requiring special handling or that are in error.

In SMP/E, the term *exception data* usually refers to HOLDDATA. HOLDDATA is often supplied for a product to indicate a specified SYSMOD should be held from installation. Reasons for holding a SYSMOD can be:

- A PTF is in error and should not be installed until the error is corrected (ERROR HOLD).
- Certain system actions may be required before SYSMOD installation (SYSTEM HOLD).
- The user may want to perform some actions before installing the SYSMOD (USER HOLD).

All the information located in the global zone, combined with the information found in the target and distribution zones, make up the data that SMP/E requires to install and track the system's software, which is often a great deal of data. You can display this information using the following SMP/E facilities:

- Query dialogs: The easiest and fastest way to obtain just the information you want
- LIST command: When you need an all-inclusive hardcopy listing of information about your system
- REPORT commands: To check and compare the zone contents and generate command output that can be used to update your system
- SMP/E CSI application programming interface: To write an application program to query the contents of your system's CSI data sets.

SMP/E data sets for inventory, zones, and libraries

To install z/OS elements in target and distribution libraries, SMP/E uses a database made up of several types of data sets.

SMPCSI (CSI) data sets

SMPCSI (CSI) data sets are VSAM data sets used to control the installation process and record the results of processing. A CSI can be divided into multiple partitions through the VSAM key structure. Each partition is referred to as a **zone**.

There are three types of zones:

- A single **global zone** is used to record information about SYSMODs that have been received into the SMPPTS data set. The global zone also contains information enabling SMP/E to access the other two types of zones, information about system utilities that SMP/E calls to install elements from SYSMODs, and information allowing you to tailor SMP/E processing.
- One or more **target zones** are used to record information about the status and structure of the operating system (or target) libraries. Each target zone also points to the related distribution zone, which can be used during APPLY, RESTORE, and LINK when SMP/E is processing a SYSMOD and needs to check the level of the elements in the distribution libraries.
- One or more **distribution zones** are used to record information about the status and structure of the distribution libraries (DLIBs). Each DLIB zone also points to the related target zone, which is used when SMP/E is accepting a SYSMOD and needs to check if the SYSMOD has already been applied.

There can be more than one zone in an SMPCSI data set (in fact, there can be up to 32766 zones per data set). For example, an SMPCSI data set can contain a global zone, several target zones, and several distribution zones. The zones can also be in separate SMPCSI data sets. One SMPCSI data set can contain just the global zone, a second SMPCSI data set the target zones, and a third SMPCSI data set the distribution zones.

SMPPTS (PTS) data set

An SMPPTS (PTS) data set is a data set for temporary storage of SYSMODs waiting to be installed. The PTS is used strictly as a storage data set for SYSMODs. The RECEIVE command stores SYSMODs directly on the PTS without any modifications of SMP/E information. The PTS is related to the global zone in that both data sets contain information about the received SYSMODs. Only one PTS can be used for a given global zone. Therefore, you can look at the global zone and the PTS as a pair of data sets that must be processed (for example, deleted, saved, or modified) concurrently.

SMPSCDS (SCDS) data set

The SMPSCDS (SCDS) data set contains backup copies of target zone entries modified during APPLY processing. Therefore, each SCDS is directly related to a specific target zone, and each target zone must have its own SCDS. SCDS data sets are used by SMP/E to store backup copies of target zone entries modified during APPLY processing. Therefore, each SCDS is directly related to a specific target zone, and each target zone must have its own SCDS.

SMP/E also uses the following data sets:

- The SMPMTS (MTS) data set is a library in which SMP/E stores copies of macros during installation when no other target macro library is identified. Therefore, the MTS is related to a specific target zone, and each target zone must have its own MTS data set.

- The SMPSTS (STS) data set is a library in which SMP/E stores copies of source during installation when no other target source library is identified. Therefore, the STS is related to a specific target zone, and each target zone must have its own STS data set.
- The SMPLTS (LTS) data set is a library that maintains the base version of a load module. The load module in this library specifies a SYSLIB allocation in order to implicitly include modules. Therefore, the LTS is related to a specific target zone, and each target zone must have its own LTS data set.
- Other utility and work data sets.

SMP/E uses information in the CSI data sets to select proper element levels for installation, to determine which libraries should contain which elements, and to identify which system utilities should be called for the installation.

System programmers can also use the CSI data sets to obtain the latest information on the structure, content, and status of the system. SMP/E provides this information in reports, listings, and dialogs to help you:
- Investigate function and service levels
- Understand intersections and relationships of SYSMODs (either installed or waiting to be installed)
- Build job streams for SMP/E processing.

Figure 16 shows sample job control language (JCL) for creating the CSI data sets.

```
//DEFINE     JOB  'accounting info',MSGLEVEL=(1,1)
//STEP01    EXEC PGM=IDCAMS
//CSIVOL    DD    UNIT=3380,VOL=SER=volid,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD   *
  DEFINE CLUSTER(                          -
                NAME(SMPE.SMPCSI.CSI)       -
                FREESPACE(10 5)             -
                KEYS(24 0)                  -
                RECORDSIZE(24 143)          -
                SHAREOPTIONS(2 3)           -
        VOLUMES(volid1)                     -
                )                           -
            DATA(                           -
                 NAME(SMPE.SMPCSI.CSI.DATA)  -
                 CONTROLINTERVALSIZE(4096)   -
                 CYLINDERS(250 20)           -
                )                           -
           INDEX(                           -
                 NAME(SMPE.SMPCSI.CSI.INDEX) -
                 CYLINDERS(5 3)              -
                )                           -
         CATALOG(user.catalog)
/*
```

Figure 16. JCL for defining a CSI VSAM data sets

What is a SYSMOD?

SMP/E can install a large variety of system updates, provided they are packaged as a system modification or SYSMOD. A SYSMOD is the actual package of elements and control information that SMP/E needs to install and track system modifications.

SYSMODs are composed of a combination of elements and control information. They are comprised of two parts, as follows:

- Modification control statements (MCSs), designated by ++ as the first two characters, that tell SMP/E:
- What elements are being updated or replaced
- How the SYSMOD relates to product software and other SYSMODs
- Other specific installation information
- Modification text, which is the object modules, macros, and other elements supplied by the SYSMOD

There are four different categories of SYSMODs, each supporting a task you might want to perform:

FUNCTION
This type of SYSMOD introduces a new product, a new version or release of a product, or updated functions for an existing product into the system.This type of SYSMOD introduces a new product, a new version or release of a product, or updated functions for an existing product into the system.

PTF A *program temporary fix* (PTF) is an IBM-supplied correction for a reported problem. They are meant to be installed in all environments. PTFs may be used as preventive service to avoid certain known problems that may have not yet appeared on your system, or they may be used as corrective service to fix problems you have already encountered. The installation of a PTF must always be preceded by that of a function SYSMOD, and often other PTFs as well.

APAR An *authorized program analysis report* (APAR) is a temporary fix designed to correct or bypass a problem for the first reporter of the problem. An APAR might not be applicable to your environment. The installation of an APAR must always be preceded by that of a function SYSMOD, and sometimes of a particular PTF. That is, an APAR is designed to be installed on a particular preventive-service level of an element.

USERMOD
This type of SYSMOD is created by you, either to change IBM code or to add independent functions to the system. The installation of a USERMOD must always be preceded by that of a function SYSMOD, sometimes certain PTFs, APAR fixes, or other USERMODs.

SMP/E keeps track of the functional and service levels of each element and uses this SYSMOD hierarchy to determine such things as which functional and service levels of an element should be installed and the correct order for installing updates for elements.

Function SYSMOD: Introducing an element in the system

One way you can modify your system is to introduce new elements into that system. To accomplish this with SMP/E, you can install a function SYSMOD.

The function SYSMOD introduces a new product, a new version or release of a product, or updated functions for an existing product into the system. All other types of SYSMODs are dependent upon the function SYSMOD, because they are all modifications of the elements originally introduced by the function SYSMOD.

When introducing a function SYSMOD, all of the element's components or code are placed in the system data sets, or libraries. Examples of these libraries are SYS1.LPALIB, SYS1.MIGLIB, and SYS1.SVCLIB.

Figure 17 shows the process of creating executable code in the production system libraries.

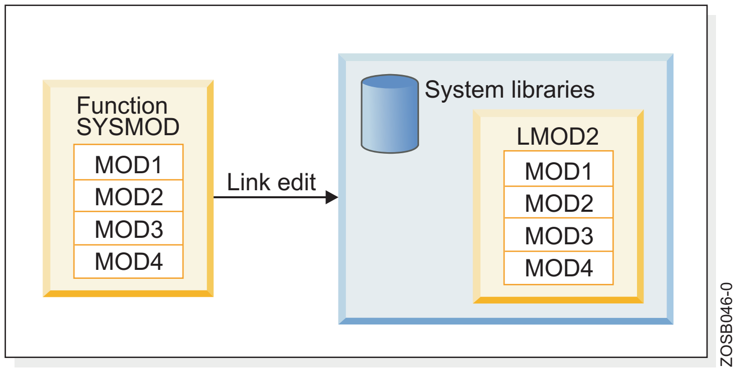


Figure 17. Introducing an element

In Figure 17, the installation of a function SYSMOD link-edits object modules MOD1, MOD2, MOD3, and MOD4 to create load module LMOD2. The executable code created in load module LMOD2 is installed in the system libraries through the installation of the function SYSMOD.

There are two types of function SYSMODs:
- A *base* function SYSMOD adds or replaces an entire system function. Examples of base functions are SMP/E and JES2.
- A *dependent* function SYSMOD provides an addition to an existing system function. It is called dependent because its installation depends upon a base function already being installed. Examples of dependent functions are the language features for SMP/E.

Both base function SYSMODs and dependent function SYSMODs are used to introduce new elements into the system. Figure 18 shows an example of a simple function SYSMOD that introduces four elements.

```
++FUNCTION(FUN0001)         /* SYSMOD type and identifier.   */.
++VER(Z038)                 /* For MVS SREL                  */.
++MOD(MOD1)  RELFILE(1)     /* Introduce this module         */
            DISTLIB(AOSFB)  /* in this distribution library. */.
++MOD(MOD2)  RELFILE(1)     /* Introduce this module         */
            DISTLIB(AOSFB)  /* in this distribution library. */.
++MOD(MOD3)  RELFILE(1)     /* Introduce this module         */
            DISTLIB(AOSFB)  /* in this distribution library. */.
++MOD(MOD4)  RELFILE(1)     /* Introduce this module         */
            DISTLIB(AOSFB)  /* in this distribution library. */.
```

Figure 18. Example of a simple function SYSMOD

PTF SYSMOD: Preventing or fixing problems with an element

When a problem with a software element is discovered, IBM supplies its customers with a tested fix for that problem. This fix comes in the form of a program temporary fix (PTF). Although you may not have experienced the problem the PTF is intended to prevent, it is wise to install the PTF on your system. The PTF SYSMOD is used to install the PTF, thereby preventing the occurrence of that problem on your system.

Usually, PTFs are designed to replace or update one or more complete elements of a system function. Look at Figure 19, which shows a previously installed load module, LMOD2. To replace the element MOD1, you would install a PTF SYSMOD that contains the module MOD1. That PTF SYSMOD replaces the element in error with the corrected element.

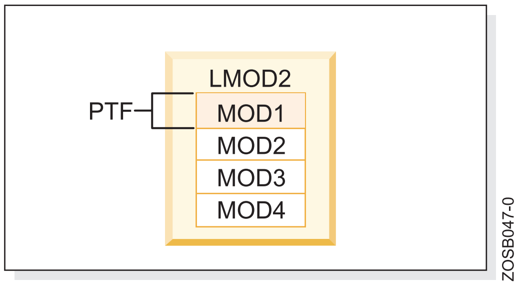


Figure 19. Preventing problems with an element

As part of the installation of the PTF SYSMOD, SMP/E relinks LMOD2 to include the new and corrected version of MOD1.

Figure 20 shows an example of a simple PTF SYSMOD.

```
++PTF(PTF0001)              /* SYSMOD type and identifier.   */.
++VER(Z038) FMID(FUN0001)   /* Apply to this product.        */.
++MOD(MOD1)                 /* Replace this module           */
            DISTLIB(AOSFB)  /* in this distribution library. */.
 ...
 ... object code for module
 ...
```

Figure 20. Example of a simple PTF SYSMOD

PTF SYSMODs are always dependent upon the installation of a function SYSMOD. In some cases, some PTF SYSMODs may also be dependent upon the installation of other PTF SYSMODs. These dependencies are called *prerequisites*.

PTF, APAR, and USERMOD SYSMODs all have the function SYSMOD as a prerequisite. In addition to their dependencies on the function SYSMOD:

- PTF SYSMODs might be dependent upon other PTF SYSMODs.
- APAR SYSMODs might be dependent upon PTF SYSMODs and other APAR SYSMODs.
- USERMOD SYSMODs might be dependent upon PTF SYSMODs, APAR SYSMODs, and other USERMOD SYSMODs.

Sometimes a PTF or even an APAR is dependent upon other PTF SYSMODs called *corequisites*. Consider the complexity of these dependencies– When you multiply that complexity by hundreds of load modules in dozens of libraries, the need for a tool like SMP/E becomes apparent.

APAR SYSMOD: Fixing problems with an element

You may sometimes find it is necessary to correct a serious problem that occurs on your system before a PTF is ready for distribution. In this situation, IBM supplies you with an authorized program analysis report (APAR). An APAR is a fix designed to quickly correct a specific area of an element or replace an element in error. You install an APAR SYSMOD to implement a fix, thereby updating the incorrect element.

In Figure 21, the shaded section shows an area of MOD2 containing an error.

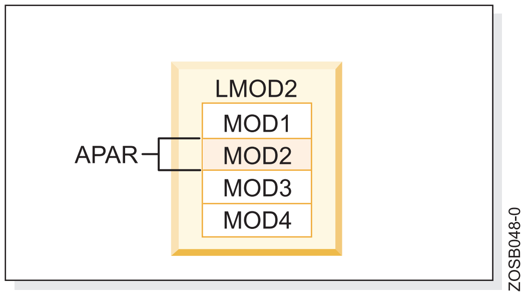


Figure 21. Fixing problems with an element

The processing of the APAR SYSMOD provides a modification for object module MOD2. During the installation of the APAR SYSMOD, MOD2 is updated (and corrected) in load module LMOD2.

Figure 22 shows an example of a simple APAR SYSMOD.

```
++APAR(APAR001)               /* SYSMOD type and identifier.   */.
++VER(Z038) FMID(FUN0001)     /* Apply to this product         */
            PRE(UZ00004)      /* at this service level.        */.
++ZAP(MOD2)                   /* Update this module            */
            DISTLIB(AOSFB)    /* in this distribution library. */.
...
... zap control statements
...
```

Figure 22. Example of a simple APAR SYSMOD

The APAR SYSMOD always has the installation of a function SYSMOD as a prerequisite, and can also be dependent upon the installation of other PTF or APAR SYSMODs.

PTF, APAR, and USERMOD SYSMODs all have the function SYSMOD as a prerequisite. In addition to their dependencies on the function SYSMOD:

- PTF SYSMODs might be dependent upon other PTF SYSMODs.
- APAR SYSMODs might be dependent upon PTF SYSMODs and other APAR SYSMODs.
- USERMOD SYSMODs might be dependent upon PTF SYSMODs, APAR SYSMODs, and other USERMOD SYSMODs.

Sometimes a PTF or even an APAR is dependent upon other PTF SYSMODs called *corequisites*. Consider the complexity of these dependencies– When you multiply that complexity by hundreds of load modules in dozens of libraries, the need for a tool like SMP/E becomes apparent.

USERMOD SYSMOD: Customizing an element

If you had a requirement for a product to perform differently from the way it was designed, you might want to customize that element of your system. IBM provides you with certain modules that allow you to tailor IBM code to meet your specific needs. After making the desired changes, you add these modules to your system by installing a USERMOD SYSMOD.

This SYSMOD can be used to replace or update an element, or to introduce a totally new user-written element into the system. In either case, the USERMOD SYSMOD is built by you either to change IBM code or to add your own code to the system.

In Figure 23, MOD3 has been updated through the installation of a USERMOD SYSMOD.

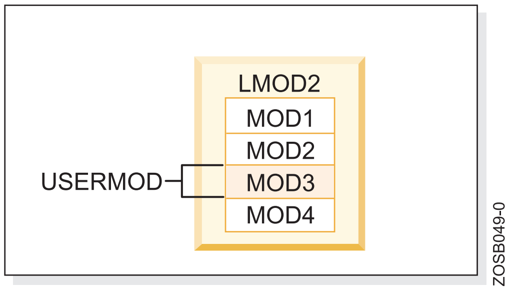


Figure 23. Customizing an element

Figure 24 shows an example of a simple USERMOD SYSMOD.

```
++USERMOD(USRMOD1)             /* SYSMOD type and identifier.   */.
++VER(Z038) FMID(FUN0001)      /* Apply to this product         */
            PRE(UZ00004)       /* at this service level.        */.
++SRCUPD(JESMOD3)              /* Update this source module     */
            DISTLIB(AOSFB)     /* in this distribution library. */.
...
... update control statements
...
```

Figure 24. Example of a simple USERMOD SYSMOD

PTF, APAR, and USERMOD SYSMODs all have the function SYSMOD as a prerequisite. In addition to their dependencies on the function SYSMOD:

- PTF SYSMODs might be dependent upon other PTF SYSMODs.
- APAR SYSMODs might be dependent upon PTF SYSMODs and other APAR SYSMODs.
- USERMOD SYSMODs might be dependent upon PTF SYSMODs, APAR SYSMODs, and other USERMOD SYSMODs.

Sometimes a PTF or even an APAR is dependent upon other PTF SYSMODs called *corequisites*. Consider the complexity of these dependencies– When you multiply that complexity by hundreds of load modules in dozens of libraries, the need for a tool like SMP/E becomes apparent.

Best practice: Keep track of system elements and modifications

The importance of keeping track of system elements and their modifications becomes readily apparent when we examine the z/OS maintenance process.

Often, a PTF contains multiple element replacements. In the example shown in Figure 25, PTF1 contains replacements for two modules, MOD1 and MOD2. Although load module LMOD2 contains four modules, only two of those modules are being replaced.

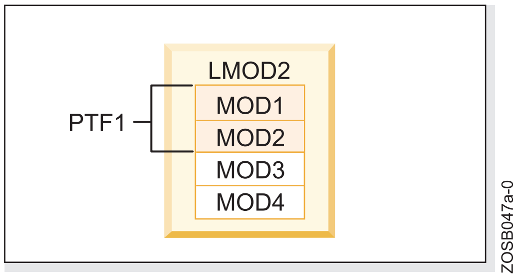


Figure 25. PTF replacement

But what happens if a second PTF replaces some of the code in a module that was replaced by PTF1, as shown in Figure 26?

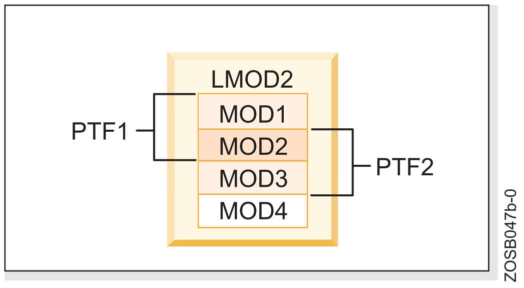


Figure 26. PTF prerequisite

In this example, PTF2 contains replacements for MOD2 and MOD3. For MOD1, MOD2, and MOD3 to interface successfully, PTF1 must be installed before PTF2. That's because MOD3 supplied in PTF2 may depend on the PTF1 version of MOD1 to be present. It is this dependency that constitutes a prerequisite. SYSMOD prerequisites are identified in the modification control statements (MCS) part of the SYSMOD package.

In addition to tracking prerequisites, there is another important reason to track system elements. The same module is often part of many different load modules, as shown in Figure 27 on page 57.

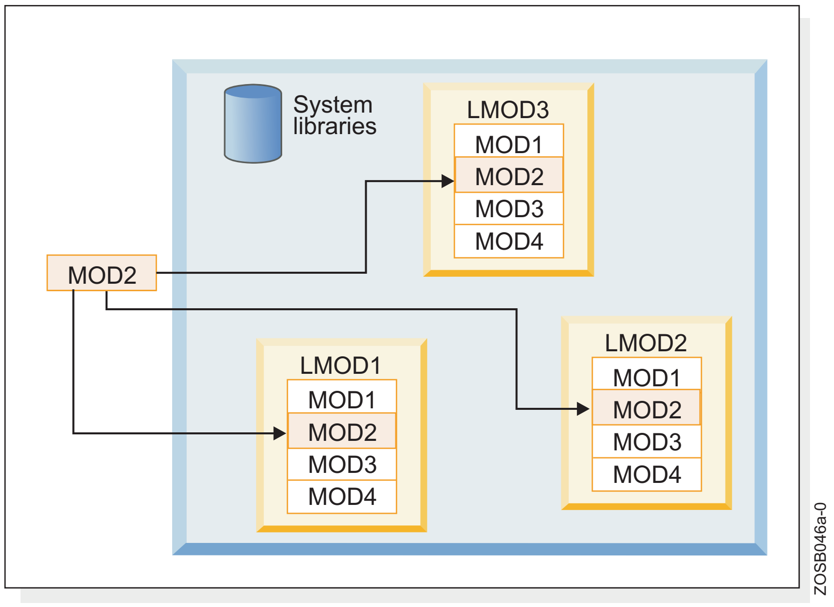


Figure 27. Load module constructions

In Figure 27, the same MOD2 module is present in LMOD1, LMOD2, and LMOD3. When a PTF is introduced that replaces the element MOD2, that module must be replaced in all the load modules in which it exists. Therefore, it is imperative that we keep track of all load modules and the modules they contain.

Tracking and controlling requisites

To track and control elements successfully, all elements and their modifications and updates must be clearly identified to SMP/E. SMP/E relies on modification identifiers to accomplish this. There are three modification identifiers associated with each element:

- Function Modification Identifiers (FMIDs) identify the function SYSMOD that introduces the element into the system.
- Replacement Modification Identifiers (RMIDs) identify the last SYSMOD (in most cases a PTF SYSMOD) to replace an element.
- Update Modification Identifiers (UMIDs) identify the SYSMOD that an update to an element since it was last replaced.

SMP/E uses these modification identifiers to track all SYSMODs installed on your system. This ensures that they are installed in the proper sequence.

The SMP/E process for installing z/OS elements or service

This summary of SMP/E processing illustrates how z/OS elements or service updates are installed.

SMP/E provides an excellent inventory and cross-reference of software dependencies, and a mechanism for installing new products or applying software maintenance. These capabilities ensure integrity of the overall system. The mechanics of using SMP/E to maintain a system can be quite simple, as shown in Figure 28 on page 58. How SMP/E works and the knowledge needed to package fixes and software producets for SMP/E is significantly complex. IBM provides instructions for system programmers to follow with all SMP/E packaged fixes, software upgrades, and products.

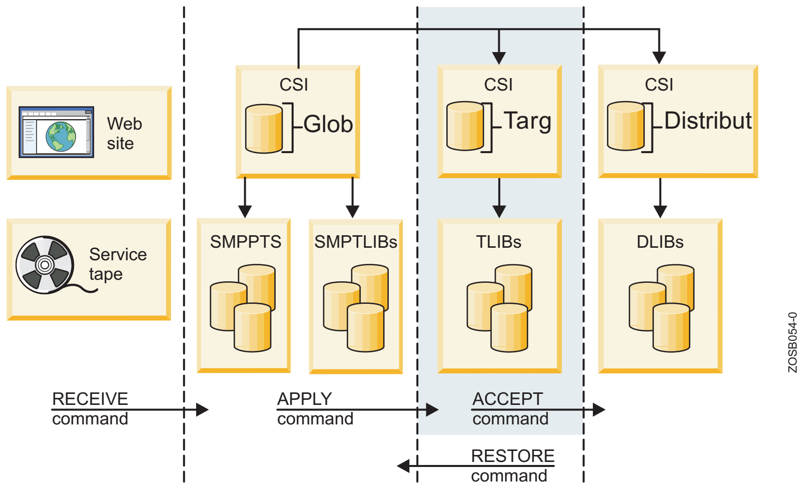


Figure 28. Overall SMP/E process flow

Figure 29 provides a sample of the job control language (JCL) that might be used to automate the process from receiving updates to applying them to a target system.

```
//SMPJOB    JOB  'accounting info',MSGLEVEL=(1,1)
//SMPSTEP   EXEC SMPPROC
//SMPPTFIN  DD   ...
/* points to file or data set containing SYSMODs to be received */
//SMPHOLD   DD   ...
/* points to file or data set containing HOLDDATA to be received */
//SMPTLIB   DD   UNIT=3380,VOL=SER=TLIB01
//SMPCNTL   DD   *
  SET       BDY(GLOBAL)         /* Set to global zone       */.
  RECEIVE   SYSMOD              /* Receive SYSMODs          */
            HOLDDATA            /* Receive HOLDDATA         */
            SOURCEID(MYPTFS)    /* Assign a source ID       */
                                /*                          */.
  LIST      MCS                 /* List the cover letters   */
            SOURCEID(MYPTFS)    /* for the SYSMODs          */
                                /*                          */.
  SET       BDY(TARGET1)        /* Set to target zone       */.
  APPLY     SOURCEID(MYPTFS)    /* Apply the SYSMODs        */
                                /*                          */.
  LIST      LOG                 /* List the target zone log */.
/*
```

Figure 29. SMP/E batch job example

SMP/E commands

Three commands– RECEIVE, APPLY, and ACCEPT– drive the SMP/E process for installing z/OS elements or service.

Additional SMP/E commands include:

- LIST command: When you need an all-inclusive hardcopy listing of information about your system
- REPORT commands: To check and compare the zone contents and generate command output that can be used to update your system

The SMP/E RECEIVE command

The RECEIVE command allows you to take a SYSMOD that is outside of SMP/E and stage it into the SMP/E library domain, which begins to construct the CSI entries that describe them. This staging allows them to be queried for input into later processes. More recently, the source can be electronic from a Web site, although usually it comes from a tape or even a third-party vendor media.

The RECEIVE process accomplishes several tasks that are shown in Figure 30.

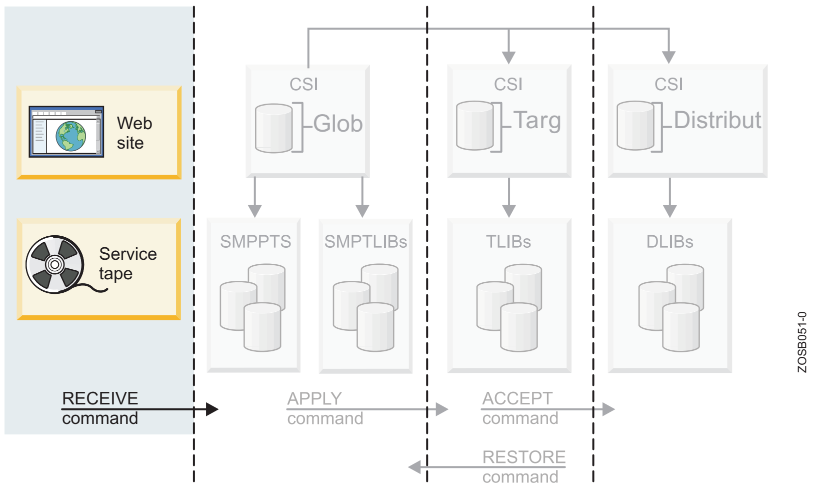


Figure 30. SMP/E RECEIVE processing

These tasks include:

- Constructing entries in the Global Zone for describing the SYSMOD.
- Ensuring the SYSMOD is valid, such as the syntax for modification control statements (MCS) associated to the products installed in the CSI.
- Installing the SYSMOD into the libraries. Example: the PTF temporary store library.
- Assessing the HOLDDATA to ensure errors are not introduced.

During the RECEIVE processing, the MCS for each SYSMOD is copied to an SMP/E temporary storage area called the SMPPTS data set, which contains the inline element replacement or update for that SYSMOD. There are also RELFILEs that package the elements in relative files that are separate from MCSs, which are mostly used by function SYSMODs. Relative files are stored in another temporary storage area called SMPTLIB data sets.

SMP/E updates the global zone with information about the SYSMODs that it has received.

In the course of maintaining the system, you need to install service and process the related HOLDDATA. For example, assume that IBM has supplied you with a service tape (such as a CBPDO or ESO tape) and you want to install it on the system. The first step is to receive the SYSMODs and HOLDDATA that are contained on the tape by entering these commands:

```
SET     BDY(GLOBAL).
RECEIVE.
```

Doing so causes SMP/E to receive all the SYSMODs and HOLDDATA on the tape.

Examples of RECEIVE commands

To receive only HOLDDATA that might require special handling or that is in error, you use this command:

```
SET      BDY(GLOBAL).
RECEIVE   HOLDDATA.
```

To receive only SYSMODs for installation into the global zone, you use this command:

```
SET      BDY(GLOBAL).
RECEIVE   SYSMODS.
```

To receive all SYSMODs, including HOLDDATA, for a specific product (for example, WebSphere Application Server), you use a command like the following:

```
SET      BDY(GLOBAL).
RECEIVE   FORFMID(H28W500).
```

The SMP/E APPLY command

The APPLY command specifies which of the received SYSMODs are to be selected for installation in the target libraries. SMP/E also ensures that all other required SYSMODs (prerequisites) have been installed or are being installed concurrently as well as in the proper sequence.

The source of the elements is the SMPTLIB data sets, the SMPPTS data set or indirect libraries depending on how is was packaged. This phase of the SMP/E process entails the following:

- Executing the appropriate utility to install the SYSMOD into the target library, depending on the type of input text supplied and target module being changed.
- Ensuring that the relationship of the new SYSMOD with other SYSMODs in the target zone is correct.
- The CSI is modified displaying the updated modules.

The APPLY command updates the system libraries and should be carefully used on a live production system. It is recommended that you initially use a copy of the production target libraries and zones.

The target zone reflects the content of the target libraries. Therefore, after the utility is completed and the zone updated, it will accurately reflect the status of those libraries.

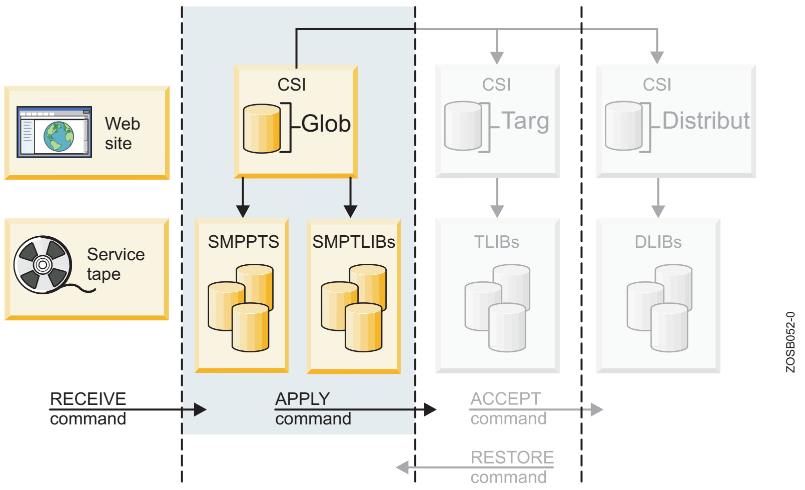


Figure 31. SMP/E APPLY processing

The APPLY processing, shown in Figure 31, is where the target zone is accurately updated:

- All SYSMOD entries in the Global Zone are updated to reflect that the SYSMOD has been applied to the target zone.
- The target zone accurately reflects each SYSMOD entry applied. Element entries (such as MOD and LMOD) are also created in the target zone.
- BACKUP entries are created in the SMPSCDS data set so the SYSMOD can be restored, if at all necessary.

Similar to the RECEIVE process, the APPLY command has many different operands for flexibility to select SYSMODs you would like to see for installation in the target libraries, and provides an assortment of output. The directives used instruct SMP/E what you want installed.

Examples of APPLY commands

To install only PTF SYSMODs, enter a command like the following:

```
SET    BDY(ZOSTGT1).
APPLY  PTFS.
```

To select PTF SYSMODs, you name them in the directives, for example:

```
SET    BDY(ZOSTGT1).
APPLY  SELECT(UZ00001, UZ00002).
```

Sometimes, you might want to install only corrective fixes (APARs) or user modifications (USERMODs) into the target library, for example:

```
SET    BDY(ZOSTGT1).
APPLY  APARS
      USERMODS.
```

At other times, you might want to update a selected product from a distribution tape:

```
SET    BDY(ZOSTGT1).
APPLY  PTFS
      FORFMID(H28W500).
```

Or:

```
SET   BDY (ZOSTGT1).
APPLY FORFMID(H28W500).
```

In these two examples, SMP/E applies all applicable PTFs for the FMID. Unless you specify otherwise, PTFs are the default SYSMOD type.

The APPLY command with the CHECK operand

There might be times when you want to see which SYSMODs are included before you actually install them. You can do this by including the CHECK operand with commands such as the following:

```
SET    BDY(MVSTGT1).
APPLY  PTFS
      APARS
      FORFMID(HOP1)
      GROUPEXTEND
      CHECK.
```

When these commands complete, you can check the SYSMOD status report to see which SYSMODs would have been installed if you had not specified the CHECK operand. If you are satisfied with the results of this trial run, you can enter the commands again, without the CHECK operand, to actually install the SYSMODs.

The SMP/E ACCEPT command

When a SYSMOD is installed into its target library, and you have tested it, you then accept the change through the ACCEPT command. This step takes the selected SYSMODs and installs them into the associated distribution libraries.

On the ACCEPT command, you specify operands to indicate which of the received SYSMODs are to be selected for installation. During this phase, SMP/E also ensures that the correct functional level of each element is selected.

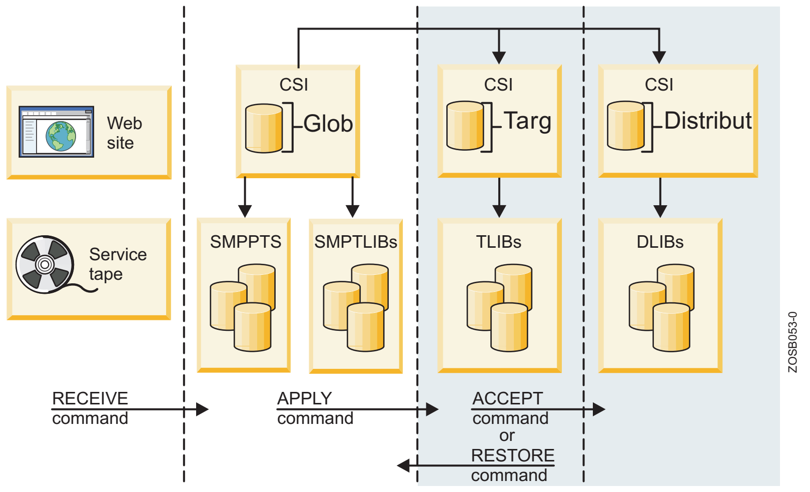


Figure 32. SMP/E ACCEPT processing

The ACCEPT command performs the following tasks, shown in Figure 32:

- Updates CSI entries with the targeted elements in the distribution zone.
- Rebuilds or creates the targeted elements in the distribution libraries using the content of the SYSMOD as input.

- Verifies the target zone CSI entries for the affected modules and SYSMODs, ensuring that they are consistent with the library content.
- Performs housekeeping of obsolete or expired elements. ACCEPT processing deletes the global zone CSI entries, PTS members and SMPTLIBs for those SYSMODs affected. For example, ACCEPT deletes the global zone SYSMOD entries and MCS statements in the SMPPTS data set for those SYSMODs that have been accepted into the distribution zone.

As a further option, you can skip having SMP/E clean up the global zone cleanup. If so, SMP/E saves this information.

There is a ″stop″ ACCEPT processing that SMP/E provides so you can ensure that all prerequisites are satisfied before the installation of the SYSMODs. This is a check for you to see what will happen (assist you in detecting problems) without actually modifying the distribution libraries.

After applying the SYSMODs into the Target zone, you can then tell SMP/E to install only the eligible PTF SYSMODs into the Distribution zone:

```
SET     BDY(ZOSDLB1).
ACCEPT  PTFS.
```

Examples of ACCEPT commands

To install PTF SYSMODS selecting the particular ones:

```
SET     BDY(ZOSDLB1).
ACCEPT  SELECT(UZ00001,UZ00002).
```

There are situations where you may want to update a particular product with all SYSMODs:

```
SET     BDY(ZOSDLB1).
ACCEPT  PTFS
        FORFMID(H28W500).
```

Or:

```
SET     BDY(ZOSDLB1).
ACCEPT  FORFMID(H28W500).
```

In these two examples, SMP/E accepts all applicable PTFs for the product whose FMID is H28W500 located in the Distribution zone ZOSDLB1.

Note: Should the SYSMOD be in error, do not accept it. Use the RESTORE process which takes the updated modules and rebuilds the copies in the Target libraries from the specific modules in the Distribution libraries. Additionally, RESTORE updates the Target zone CSI entries to reflect the removal of the SYSMOD.

Note: When ACCEPT processing is complete, it cannot be backed out. ACCEPT should only be performed after you are satisfied with the performance and stability of the elements of the SYSMOD. **The changes from ACCEPT processing are permanent.**

The ACCEPT processing for prerequisite SYSMODs

When installing a SYSMOD, you may not know whether it has prerequisites (sometimes, an ERROR SYSMOD is held). In these situations, you can direct SMP/E to check whether an equivalent (or superseding) SYSMOD is available by specifying the GROUPEXTEND operand:

```
SET     BDY(ZOSDLB1).
ACCEPT  PTFS
        FORFMID(H28W500)
        GROUPEXTEND.
```

A good way to see which SYSMODs are included before you actually install them is with the CHECK operand:

```
SET     BDY(ZOSTGT1).
ACCEPT  PTFS
        FORMFMID(H28W500)
        GROUPEXTEND
        CHECK.
```

Note: If SMP/E cannot find a required SYSMOD, it looks for and uses a SYSMOD that supersedes the required one.

ACCEPT reporting

When this last phase is completed, the following reports will assist you to assess the results:

- SYSMOD Status Report: Provides a summary of the processing that took place for each SYSMOD, based on the operands you specified on the ACCEPT command.
- Element Summary Report: Provides a detailed look at each element affected by the ACCEPT processing and in which libraries they reside.
- Causer SYSMOD Summary Report: Provides a list of SYSMODs that caused other SYSMODs to fail and describes the errors that must be fixed in order to be successfully processed.
- File Allocation Report: Provides a list of the data sets used for the ACCEPT processing and supplies information about these data sets.

Part 6. Appendixes

Notices

This information was developed for products and services offered in the U.S.A.

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
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Programming interface information

This book documents information that is NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

Printed in USA