GC28-0627-2
File No. S370-36

Systems

OS/VS2 MVS System Programming Library: Job Management

VS2 Release 3.8

Includes Selectable Units:

IBM 3800 Printing Subsystem	VS2.03.810
MSS Enhancements	5752-824
3838 Vector Processing Subsystem Support	5752-829
System Security Support	5752-832

Third Edition (October, 1978)

This is a major revision of, and obsoletes, GC28-0627-1 and incorporates changes released in the following System Library Supplements and Technical Newsletters:

IBM 3800 Printing Subsystem	VS2.03.810	GN28-2726	(dated May 28, 1976)
MSS Enhancements	5752-824	GC28-0791	(dated February 14, 1977)
3838 Vector Processing Subsystem Support	5752-829	GC28-0924	(dated August 15, 1977)
System Security Support	5752-832	GC28-0836	(dated May 27, 1977)
Service TNLs		GN28-2825 and GN28-2741	

This edition, with Technical Newsletter GN28-4681 applies to Release 3.8 of OS/VS2 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM Systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming or services which are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming or services in your country.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

Since the system must be built for the entire user community, certain parts of job management might need to be tailored to fit the needs of individual users. This manual provides the system programmer with information about the following facilities, so that he can use them in his environment:

- Allocation services, described in Chapter 1. This chapter describes performance considerations associated with allocation, volume attributes, details of allocation processing, which information might be needed to diagnose JCL error messages, and volume demounting information.
- Dynamic allocation (SVC 99) services, described in Chapters 2 and 3. Chapter 2 described each function provided by SVC 99, features of SVC 99 processing whose applicability depends on whether you are using SVC 99 function in a batch or time-sharing environment, and installation option with which you can control SVC 99 processing. Chapter 3 provides details on requesting SVC 99 functions, including a description of the parameter list, the information that can be coded in the parameter list, and SVC 99 return codes, error codes, and information codes. Chapter 3 also includes an example of a dynamic allocation request.
- Internal reader, described in Chapter 4. Chapter 4 describes how you may use the internal reader facility to build jobs and place them directly in the input stream.
- Job Scheduler Restart Support, described in Chapter 5. This section describes how scheduler restart allows a failing job to terminate or resume processing.
- Assigning Special Program Properties, described in Chapter 6. This chapter shows you how to assign special properties to programs such as: non-swappable, do not cancel, bypass password protection, privileged, and no waiting allowed.
- System Log, described in Chapter 7. This chapter explains where and how the operating system, problem programs, and operator communications are recorded.
- Updating the Master Job Control Language, described in Chapter 8.
- Updating the Subsystem Name Table, described in Chapter 9.
- External Writers, described in Chapter 10. This chapter describes the processing of the JES, IBM-supplied and installation written writers. This chapter also describes the rules for external writers written by user-installations.

In addition, the documentation on dynamic allocation functions and internal readers can be used by application programmers.

Note: This publication describes system allocation; however, JES3 main device scheduling is functionally similar to system allocation. For a detailed description of JES3 allocation, see *OS/VS2 MVS System Programming Library: JES3*.

The following list of manuals should be used with this manual to provide additional information and details:

- *OS/VS2 MVS System Programming Library: Initialization and Tuning Guide*, GC28-0681.
- *OS/VS2 MVS System Programming Library: System Generation Reference*, GC26-3792.
- *OS/VS2 System Logic Library* (7 volumes), SY28-0713, SY28-0714, SY28-0715, SY28-0716, SY28-0717, SY28-0718, and SY28-0719.
- *OS/VS2 MVS System Programming Library: System Management Facilities (SMF)*, GC28-0706.
- *Operator's Library: OS/VS2 MVS System Commands*, GC38-0229.
- *IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3846.
- *OS/VS2 MVS JCL*, GC28-0692.
- *OS/VS2 MVS Data Management Macro Instructions*, GC26-3873.
- *OS/VS2 MVS Checkpoint/Restart*, GC263877.
- *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor*, GC28-0648.
- *OS/VS2 IBM 3540 Programmer's Reference*, GC24-5111.
- *OS/VS2 MVS System Programming Library: Service Aids*, GC28-0674.
- *OS/VS2 MVS System Programming Library: Debugging Handbook* (2 volumes), GC28-0708 and GC28-0709.
- *OS/VS2 MVS System Programming Library: JES2*, GC23-0002.
- *OS/VS2 MVS System Programming Library: JES3*, GC28-0608.

Contents

Figures

Summary of Amendments for GN28-4681 TNL to GC28-0627-2 OS/VS2 Release 3.8

This technical newsletter is being issued in support of a number of service updates (both technical and editorial) to OS/VS2 Release 3.7. This edition also contains two sections ("Guidelines for Improving Allocation Response" and "TSO Allocation Suggestions") which were originally documented in *OS/VS2 MVS System Programming Library: Initialization and Tuning*.

Summary of Amendments for GC28-0627-2 OS/VS2 Release 3.7

The dynamic allocation function section of this manual has been rewritten and reorganized into two sections (Chapter 2 and Chapter 3) to improve text clarity and usability. Changes have been made to distinguish the difference between the time-sharing and batch users when using this function.

Changes have also been made throughout this publication to reflect updates to OS/VS MVS Release 3.7 and the following SUs: 10, 24, 29, and 32. In addition, technical and editorial changes have been made throughout the manual.

Summary of Amendments for GC28-0627-1 OS/VS2 Release 3.7

Changes have been made throughout this publication to reflect a Service Update to OS/VS2 Release 3.7. In addition, pertinent technical and editorial changes have been made in the following areas:

Dynamic Allocation

- Allocation of a data set name and a ddname is clarified.
- Conditions under which the volume serial will be unavailable at the completion of allocation are discussed.

Internal Readers

Dynamically allocating and using an internal is discussed.

Miscellaneous

- IEALIMIT documentation is removed and now appears in *OS/VS2 System Programming Library: Supervisor*.
- Support for the IBM 3344 Direct Access Storage Device and the IBM 3350 Direct Access Storage is documented. This information is for planning purposes only.
- The entire JES3 section is removed. JES3 is described in *OS/VS2 System Programming Library: JES3*, GC28-0608.
- The entire JES2 section is removed. JES2 is described in *OS/VS2 System Programming Library: JES2*, GC23-0001.

Summary of Amendments for GN28-2601 TNL to GC28-0627-0 OS/VS2 Release 3

This TNL primarily provides job management information for the Job Entry Subsystem 3 (JES3). The major topics include:

- JES3 Configuration
- JES3 Job Management
- JES3 Reliability, Availability, Serviceability
- JES3 User Services

This TNL also provides miscellaneous corrections and additions unrelated to JES3.

Chapter 1: Allocation Services

The allocation routines assign units, volumes, and data sets in support of job processing. They allocate resources in response to JCL DD statements at step initiation and permit data set allocation for jobs in progress (dynamic allocation), a capability formerly available only to TSO users.

Dynamic allocation is described in detail in chapters 2 and 3 of this book. The considerations and rules for coding DD statements are in *OS/VS2 MVS JCL*.

A description of how the design of the allocation routines relates to their performance is in *OS/VS2 System Programming Library: Initialization and Tuning Guide*. This *Initialization and Tuning Guide* is of particular value to the system programmer interested in suggesting coding practices and selecting procedures to maximize allocation efficiency.

Performance Considerations Associated With Allocation

One of the biggest concerns associated with performance in allocation is serialization. The design of the allocation routines in MVS attempts to minimize the following types of serialization:

- Serialization to ensure that the status of a device is unchanged (that is, static) while the device is being selected
- Serialization of the use of some devices

The allocation routines process requests in the order listed below. To reduce serialization, you should allocate data sets, volumes, and devices from categories highest in the list, if possible. As you move down the list, the degree of serialization and processing time increases.

1. Requests that require no specific units or volumes, for example, DUMMY, VIO, and subsystem data sets (not serialized).
2. Requests to sharable units, that is, direct access units with permanently resident or reserved volumes mounted on them (not serialized).
3. Teleprocessing devices (serializes only requested teleprocessing devices).
4. Mounted volumes and devices that do not need volumes (serialized only on the set of devices eligible to satisfy the request). During this processing, the automatic volume recognition (AVR) function reads the volume serial number of any volumes that have been premounted on the serialized devices.
5. Online, non-allocated devices that need volumes mounted by the operator or by MSS (serialized only on the set of devices eligible to satisfy the request).
6. All remaining requests — for example, requests that need offline devices and/or devices that are allocated to other jobs and that cannot be shared (serialized only on the set of devices eligible to satisfy the request).

Note: Allocation treats MSS devices (3330V) as direct access storage devices.

Within steps 4, 5, and 6, the device preference table determines the order in which allocation selects devices, if a request is eligible for more than one generic device type.

You can define a device preference table for your installation or use the default device preference table provided by IBM. The default device preference table lists the fastest generic device types first. If such specification causes heavy contention for the fastest eligible devices, you can list generics with many devices (and many channels) first in the device preference table; they will then receive preference. As a secondary consideration, the increased number of preferred units and channels will give the system resources manager a large selection for its choices. *OS/VS2 MVS System Programming Library: System Generation Reference* contains information on the default device preference table and on how to specify an installation-defined device preference table.

Guidelines for Improving Allocation Response

The following suggestions should help your installation to make best use of the redesigned device allocation routines:

- Within the limit of page space availability, encourage the use of VIO data sets. (For further information, see the topic entitled "VIO Performance".)
- Set up a sufficient number of permanently resident and reserved DASD volumes on line, to avoid contention for a few volumes of these types. You can check for contention by running MF/1 to obtain device activity reports. The volumes should be spread across channels so that the system resource manager (SRM) can balance the channel load.
- Use the UNITNAME sysgen macro to define separate esoteric subgroups within major generic device types, so that different subsets of users can request separate subgroups of devices. The purpose is to minimize contention for the same devices among the various subsets of users. For example, an installation whose batch and time sharing users request allocation of 3330s could separate the two types of user requests as follows:

  ```
  UNITNAME UNIT=(330,4),NAME=SYSBATCH
  UNITNAME UNIT=(334,4),NAME=SYSTSO
  ```

  The effect of this specification is that allocations to SYSBATCH serialize only requests for units 330-333, instead of the entire 3330 generic devices. Similarly, allocation to SYSTSO serialize only requests for units 334-337.
- Use the DEVPREF keyword of the SCHEDULR sysgen macro to minimize contention for the fastest devices. The DEVPREF keyword sets up the device preference table. This table determines the order in which device types will be selected by allocation if a request is eligible for more than one device type (for example, UNIT=SYSDA). If the keyword is not specified, the default device preference table lists the fastest generics first. If such specification causes heavy contention for the fastest eligible devices, you can specify the DEVPREF list so that generics with many devices (and many channels) are listed first and are therefore given preference. As a secondary consideration, the increased number of preferred units and channels gives the SRM a large selection for its choices.
- Keep all operable devices online if possible. (This is old advice and does not depend on the redesign of device allocation.)
- Try to avoid the use of specific unit address (for example, UNIT=253) in DD statements for volumes that are neither permanently resident nor reserved. A specification of specific unit serializes the request on the entire device type. For example, if unit 253 is a 3330, a specific unit request (UNIT=253) will be serialized with other request for any 3330. Instead of using specific unit address, use subsets of the generic device type, as suggested earlier in this topic.

- Resolve the question of whether the operator should respond HOLD or NOHOLD when a job must wait for other jobs to free devices or volumes, and whether a message is to be issued to the operator. The criteria for resolving the question of whether the operator should respond HOLD or NOHOLD is:

  HOLD — This means that the job should wait while holding devices and volumes already allocated to the job. Select this option if the needed resources are constantly being freed, and allocation requests for other jobs will probably not be held up by the requests made for this job. This job can hold up other requests in either of two ways: it has already allocated units needed for another job, or its allocation requests are serialized on devices it is waiting for.

  NOHOLD — This means that the job waits without holding devices and volumes already allocated to the job. Select this option if the needed resources may not be freed for some time, and allocation requests for this job are likely to hold up requests issued for other jobs.

  Note: Requests for dynamic allocation are not held up by requests waiting for batch allocation, even though the jobs awaiting batch allocation are holding resources.

- Before the end of a job step or TSO session, free data sets, volumes, or devices. The freed resources can then by used for other jobs or sessions. You can free the resource when a data set is closed by specifying FREE=CLOSE on the associated DD statement. (This option is a new facility in MVS.) Note that when subsequent steps of a job require the same data set, the resource must be reallocated prior to being reaccessed (or else the OPEN fails). Use discretion when freeing the resources because once a resource is freed, its continuing availability cannot be guaranteed.
- Invoke dynamic allocation from a batch job by means of respecifying SVC 99. (The details are described in chapters 2 and 3 of this manual.) The advantage of invoking SVC 99 is that the batch job allocates the resource only when it is needed, and frees the resource as soon as it is no longer needed. (FREE=CLOSE can also free the resource, if it is specified on a DD statement.) Resources are thus more readily available to other requesting jobs. A disadvantage is that the batch job must handle a return code if the requested resource is not available. (With conventional allocation via DD statements, the system would cause the job to wait for the requested resource(s) to become available.) Note that an authorized program need not handle a return code if a requested resource is not available. The authorized program can request a 'wait for the resource" when it invokes dynamic allocation. Unfortunately, there is no deadlock detection in this case.
- Premount all private volumes including private catalogs before running the jobs that request these volumes. Premounting of private volume assumes the user knows the I/O demands for the new data sets and volumes, as well as current channel utilization. Note the AVR (Automatic Volume Recognition) is no longer optional.

TSO Allocation Suggestions

The following suggestions should improve TSO allocations during TSO sessions although they might extend logon time:

- DD statements that a user wants in all his TSO sessions should be placed in a LOGON procedure. This technique has these advantages:

  A. Allows volumes to be mounted.

  B. Provides recovery from an offline device condition. Messages tell the operator to VARY the device online.

  C. Saves repeated allocation and freeing of the same data set by successive commands in the same TSO session.

- The DYNAMNBR parameter value in the EXEC statement should be carefully chosen. The value should be large enough so that it is not readily exceeded by dynamic allocation requests. Note that the maximun number of concurrently allocated resources for any TSO session is 1635.

Multiple Versions of Device Allocation Tables

By defining the device groups, you can group devices under various esoteric names (for example, SYSDA for all 3330s and 2314s). However, the set of device groups you define by the eligible device tables (EDT) might not be the ideal way to group devices for every workload mix or processor configuration. By defining multiple versions of the device allocation tables (DEVNAMET, IEFDEVPT, DEVMASKT, and IEFEDTTB), you can define different sets of device groups for the different workload environments.

You define multiple versions of the device allocation tables by multiple invocation of the EDTGEN macro; you may select a specific version of the tables at IPL. Details of creating and selecting a version of the tables are included in see the following topics.

Creating Multiple Versions of the Devices Allocation Tables

Use the EDTGEN generation process to create a version of the device allocation tables. When the EDTGEN generation process completes, the newly-defined device allocation tables are placed in a member of the partitioned data set (PDS) SYS1.MLPALIB or a PDS defined by the user of the EDTGEN macro. Each invocation of the EDTGEN generation process causes the assembling and link-edit of the device allocation tables.

For additional details on using the EDTGEN macro, see *OS/VS MVS System Programming Library: System Generation*.

Using A Version of the Device Allocation Tables

To use a particular version of the device allocation tables you must specify the parameter MLPA=nn or SYSP at IPL. nn is two digits to be appended to IEALPAnn thus providing the member name of the SYS1.PARMLIB partitioned data set that names the library and the device allocation tables (DEVNAMET, IEFDEVPT, DEVMASKT, and IEFEDTTB) to be used. For a detailed description on how to use the MLPA or SYSP facility, see *OS/VS2 MVS System Programming Library: Initialization and Tuning Guide*.

Note: Refer to OS/VS2 MVS System Programming Library: JES3 for any alterations that might be needed for the JES3 INISH deck.

Before using any set of device allocation tables, you should validate the tables by executing the EDT verification routine.

Eligible Device Table Verification Routine (IEFEB400)

The EDT verification routine validates that the entries in the EDT match the device definitions in the UCBs in the nucleus. There are two environments during which IEFEB400 is invoked:

- In problem program mode by the user during Stage II of SYSGEN. For a detailed description of how you may request execution of IEFEB400, refer to *OS/VS2 MVS System Programming Library: System Generation*.
- In supervisor mode during IPL. The verification routine IEFEB400 is invoked by the master scheduler initialization routine (IEEVIPL). The EDT that is being loaded by the system will be the EDT used in the verification process.

The verification processing begins with the EDT-id message (IEF923I) being issued to the hardcopy log. If no errors are found by the verification program, the successful verification message (IEF926I) is written to hardcopy. If EDT does not match the nucleus data, a return code of four is set and the id message is written first to hardcopy and then to the operator. As a result of this error, the message may be written again to the hardcopy log and the request for an EDT will be ignored. If the message describes an unintentional mismatch, the operator should re-IPL using the device allocation tables from the last system generation or I/O generation.

Warning: If the operator does not re-IPL as a result of an unintentional mismatch, the error will not be evident until allocation attempts to reference the mismatched UCBs or an incorrect allocation may occur depending on the type of error.

The verification routine can detect the following type of error conditions:

- A device type defined in the EDT does not match any of the device types in the UCB. Message IEF924I is isued as a result of this situation.

  Note: Teleprocessing device types are not verified; only the device class is checked.
- The EDT defines a device for which there is no pointer in the IOS look-up table. Message IEF925I is issued as a result of this situation.

When the EDT verification routine is executed in problem program mode and an error is found the following messages are issued: IEF923I, IEF924I and IEF925I. When no errors are found a return code of zero is set and message IEF926I is issued.

The following errors are detected by components other than the verification routine:

- An OPEN error. Data management issues an appropriate error message and possibly terminates the task, depending on the severity of the error. A return code of 8 is set.
- A PUT error, which occurred while writing messages to the SYSPRINT data set. Data management issues an appropriate error message and possibly terminates the task, depending on the severity of the error. A return code of 12 is set.
- A LOAD error. Message IEA703I is issued and the task is terminated.

Assigning Volume Attributes

You may assign volume attributes during IPL in the VATLIST member of SYS1.PARMLIB. You can control the following information, depending on the attributes you assign to a volume:

- Eligibility for demounting, determined by the mount attribute
- The types of data sets that are assigned to a volume, determined by the use attribute

You assign mount and use attributes to tape and direct access volumes. In addition, the system can assign the nonsharable attribute to direct access volumes. The next two topics describe these attributes.

Mount and Use Attributes

Every volume is assigned a mount and use attribute at IPL via a VATLIST entry, as the result of a MOUNT command, or when first used by a job. The mount attribute controls volume demounting. The use attribute is one of the factors that controls allocation of mounted volumes to data set requests. The mount and use attributes are:

- Mount
  - — Permanently resident
  - — Reserved
  - — Removable
- Use
  - — Public
  - — Private
  - — Storage

A *public* volume is a direct access volume that is eligible for allocation to a temporary data set when a specific volume is not requested or the PRIVATE volume subparameter is not specified. It can also be allocated when its volume serial number is specified.

A *private* volume is one that can only be allocated when its volume serial number is explicitly or implicitly specified.

A *storage* volume is a direct access volume that is eligible for allocation to both non-temporary and temporary data sets when no specific volume is requested and PRIVATE is not specified. Storage volumes usually contain non-temporary data sets, but temporary ones will be assigned to storage volumes if they cannot be assigned to public volumes.

The following points list the mount attributes and describe how the mount and use attributes are assigned to a volume:

- **Permanently resident** volumes cannot be demounted. Only direct access volumes can be permanently resident. Although the user may designate all direct access volumes as permanently resident in the "volume attribute list" (VATLSTxx) in SYS1.PARMLIB, the following volumes are always permanently resident:
  - All volumes that cannot be physically demounted, such as drum storage and fixed disk volumes
  - The IPL volume
  - The volume containing the system data sets, such as SYS1.LINKLIB.

  An installation can assign a permanently resident volume the use attribute of public, private, or storage in the VATLST member of SYS1.PARMLIB. The default value is public.

Note: 3344 emulated 3340s and 3350 emulated 3330-1s and 3330-11s must be made permanently resident by using the VATLST member in SYS1.PARMLIB.

- **Reserved volumes** remain mounted until the operator issues an UNLOAD command. Both direct access and tape volumes can be assigned reserved attribute. A tape volume becomes reserved as a result of a MOUNT command; a direct access volume, as a result of a MOUNT command or a VATLST entry. A volume is usually designated as a reserved volume to avoid repeated mounting and demounting of the volume when it is to be used by many jobs.

  An installation can assign a reserved *direct access* volume the use attribute of public, private, or storage. The use attribute is assigned to the volume either in the VATLST member of SYS1.PARMLIB or in the use parameter of the MOUNT command, depending on how the volume becomes reserved.

  A reserved *tape* volume can be assigned the use attribute of private or public.
- **Removable volumes** are those that are neither permanently resident nor reserved. Removable volumes can be demounted either after the end of the job in which they are last used or when the unit on which the volume is mounted is needed for another volume.

  The use attribute of public or private can be assigned to a removable *direct access* volume as follows. The use attribute of public is assigned when the JCL PRIVATE volume subparameter is not coded. The use attribute of private is assigned when the PRIVATE volume subparameter is coded.

  A removable *tape* volume can be assigned the use attribute of public or private. The use attribute of public is assigned when the PRIVATE subparameter is not coded, a nonspecific volume request is made, and the data set is temporary (a system-generated data set name or a disposition of DELETE). The use attribute of private is assigned when the PRIVATE subparameter is coded, a specific volume request is made, or the data set is nontemporary (a non system-generated data set name or a disposition other than DELETE).

Figure 1 summarizes the type of volume that can be assigned to satisfy a specific or nonspecific volume request for a temporary or nontemporary data set; how these attributes are assigned; and how the volume is demounted.

Volume State	Temporary Data Set	Nontemporary Data Set	How Assigned	How Demounted
	Type of Volume Request			
Public/ Permanently Resident[1]	Nonspecific or Specific	Specific	VATLST entry or by default	Always[2] mounted
Private/ Permanently Resident[1]	Specific	Specific	VATLST entry	Always[2] mounted
Storage/ Permanently Resident[1]	Nonspecific or Specific	Nonspecific or Specific	VATLST entry	Always[2] mounted
Public/ Reserved (Tape and direct access)	Nonspecific or Specific	Specific	Direct access: VATLST entry on MOUNT command Tape: MOUNT command	UNLOAD or VARY OFFLINE commands
Private/ Reserved (Tape and direct access)	Specific	Specific	Direct access: VATLST entry on MOUNT command Tape: MOUNT command	UNLOAD or VARY OFFLINE commands
Storage/ Reserved[1]	Nonspecific or Specific	Nonspecific or Specific	VATLST entry or MOUNT command	UNLOAD or VARY OFFLINE commands
Public/ Removable (Tape and direct access)	Nonspecific or Specific	Specific	VOLUME=PRIVATE is not coded on the DD statement. (For tape, nonspecific volume request and a temporary data set also causes this assignment.)	When unit is required by another volume; or by UNLOAD or VARY OFFLINE commands.
Private/ Removable (Tape and direct access)	Specific	Specific	VOLUME-PRIVATE is coded on the DD statement (For tape, a specific volume request or a nontemporary data set also causes this assignment.)	At job termination for direct access; at step termination or dynamic unallocation for tape (unless VOL= RETAIN or a disposition of PASS was specified); or when the unit is required by another volume.

[1]Direct access volumes only.

[2]Note: VARY OFFLINE accomplishes demounting without resetting the UCB permanently-resident flag such that, after a subsequent VARY ONLINE command, the volume on the device will be permanently resident.

Figure 1. Combinations of Mount and Use Attributes

The Nonsharable Attribute

The system assigns the nonsharable attribute to volumes that might require demounting during step execution. When this attribute is assigned to a volume, the volume cannot be assigned to any other data set until the nonsharable attribute is removed at the end of step execution.

The following types of volume requests result in the nonsharable attribute:

- Specific volume requests that specify more volumes than devices.
- A nonspecific PRIVATE volume request that has a volume count greater than the device count. (For MSS, the MSVGP parameter has the same effect as the PRIVATE parameter.)
- A request for unit affinity to a data set defined earlier in the same job step, when the requested data set resides on a different volume.
- A request for deferred mounting of a volume on which the requested data set resides.

The system automatically assigns the nonsharable attribute as a result of the preceding cases. The system will not; however, assign the nonsharable attribute to a permanently resident or reserved volume. If your request is for three permanently resident volumes and only two devices the system will allocate the three permanently resident volumes.

Figure 2 shows the system action for sharable and nonsharable requests.

The Request is:	The Volume is Allocated:	
	Sharable	Nonsharable
Sharable	allocate the volume	wait[1]
Nonsharable	wait[1]	wait[1]
[1]The operator has the option of failing the request. The request will always fail if waiting is not allowed.		

Figure 2. Sharable and Nonsharable Volume Requests

Recovery of Allocation Resources

When an address space abnormally terminates, the allocation routines attempt to unallocate all unit control blocks (UCBs) allocated to the address space. Unallocation occurs for the following types of UCBs:

- Tape
- Unit record
- Teleprocessing
- Nonsharable, direct access

The allocation routines **cannot** unallocate **sharable, direct access** UCBs because they might be allocated to more than one address space. Therefore, sharable, direct access UCBs are unallocated during the next IPL of the system. These units (UCBs) cannot be varied offline or unloaded until the next IPL of the system.

Processing of Allocation Requests

This topic provides details on how allocation satisfies volume requests, determines the number of volumes and units to assign to a request, and satisfies requests for volume demounting. This information might be needed to understand how allocation routines interpret the JCL parameters you code to request allocation services.

Satisfying Specific Volume Requests

A specific volume request informs the system of the volume serial number of the volume required. In the following cases the system can satisfy a request for a specific volume that is already mounted:

- The volume is permanently resident or reserved. (The volume is assigned regardless of the requested use attribute, and the use attribute is not changed by the allocation.)
- The direct access volume is a removable volume that does not have the nonsharable attribute and is being used by a concurrently executing step. (If your request would make the volume nonsharable, the system will assign you that volume only when all other job steps using the volume have terminated.)
- The direct access volume is removable but not allocated. The use attribute (private or public) assigned to the volume when it is allocated is determined by the presence or absence of the PRIVATE subparameter.
- The tape volume is a scratch volume and is not in use. The use attribute of private is assigned to the volume if the request is for a permanent data set or if PRIVATE is coded.

Figure 3 shows the effect of the user's request on use attributes.

The Request is:	The Volume's Use Attribute is:		
	Private	Public	Storage
Private	stays private	changes to private	no change
Public	stays private	stays public	no change

Figure 3. Private and Public Volume Requests

Satisfying Nonspecific Volume Requests

A nonspecific volume request allows the system to choose the volume and unit to be assigned to the data set from the group of units name in the UNIT parameter. This type of request is used for new data sets.

Listed below are possible types of nonspecific volume requests:

1. A private volume for a temporary or nontemporary data set
2. A non-private volume for a temporary data set
3. A non-private volume for a nontemporary data set

The system satisfies these different types of requests as described below:

1. For a nonspecific volume request for a private direct access or tape volume (temporary or nontemporary data set), the system requests the operator to mount a volume. The operator should mount a volume whose space is unused; this gives the user control over all space on the volume. Once mounted, the volume is assigned the use attribute of private.
2. For a nonspecific volume request for a non-private **direct access** volume that is to contain a temporary data set, the system attempts to assign a public or storage volume that is already mounted, or, if no space is available, it requests the operator to mount a removable volume.

If the system selects a mounted volume, its use attribute is not changed. If a removable volume is mounted, the system assigns it the use attribute of public.

For a nonspecific volume request for a non-private **tape** volume that is to contain a temporary data set, the system assigns a scratch volume that is already mounted or it requests the operator to mount a tape volume. Once mounted, the system assigns the volume the use attribute of public.

3. For a nonspecific volume request for a non-private **direct access** volume that is to contain a nontemporary data set, the system assigns a storage volume if one is mounted on an eligible device. Otherwise, the system treats the request as a nonspecific volume request for a private volume, as described in (1) above.

   For a nonspecific volume request for a non-private **tape** volume that is to contain a nontemporary data set, the system treats the request as a nonspecific volume request for a private volume, as described in (1) above.

The following section notes the differences in handling nonspecific requests for MSS volumes.

Satisfying a Nonspecific Request for an MSS Device

An MSS (3330V) nonspecific request has several characteristics possibly different from non-MSS requests; however, MSS requests are still handled in basically the same manner as non-MSS requests. Following are several types of MSS nonspecific volume requests:

1. A private volume for a temporary or nontemporary data set
2. A private, MSS group volume for a temporary or nontemporary data set
3. A non-private volume for a temporary data set
4. A non-private volume for a nontemporary data set

The system satisfies these different types of requests as described below. (Coding MSVGP=grpname implies PRIVATE and identifies to the system an installation-defined group of MSS volumes.)

1. A request for a private volume (temporary or nontemporary data set) defaults to a private MSS group request with a default name of SYSGROUP. If the installation has defined one or more volumes in this group, the system selects one with sufficient space to satisfy the request and causes the volume to be mounted.
2. For private MSS group volume requests, the system selects and mounts a volume with sufficient space from the specified group.
3. For a nonspecific request for a non-private 3330V volume that is to contain a temporary data set, the system attempts to assign a public or storage 3330V volume that is already mounted. If none is mounted, the request defaults to SYSGROUP and is handled as in (1) above.
4. For a nonspecific request for a non-private 3330V volume that is to contain a nontemporary data set, the system assigns an already mounted storage 3330V volume. If none is mounted, the request defaults to SYSGROUP and is handled as in (1) above.

Determining Numbers of Volumes/Units per Request

Before assigning volumes and units for a job step or for a dynamic allocation request, the allocation routines must determine:

- The maximum number of volumes per request
- The maximum number of units per request
- The number of units per job step

The maximum numbers are calculated because more units than specified might actually be needed.

Volumes per Request

The maximum number of tape volumes or direct access volumes required to satisfy any request is the greater of:

- The volume count specified in the VOLUME parameter
- The number of volume serials implicitly or explicitly specified

The number of volume serials available is one of the following:

- The number of volume serials specified.
- The number of volumes obtained through VOL=REF (only if VOL=REF was coded).
- The number of volume serials that the data set resided on when it was passed (only if the request is for an existing data set that was passed from a prior step and neither volume serials nor VOL=REF was specified).
- The number of volume serials obtained from the catalog (only if the request is for an existing data set that was not passed from a prior step and neither volume serials nor VOL=REF was specified).
- The number of volume serials minus the volume sequence number plus one (only if the request is for an existing data set in which the volume sequence number specified is not greater than the number of volume serials). For example, if 8 volume serials are calculated to be used and a volume sequence number of 4 is specified, then the number of volume serials to be allocated would be 5 (8 - 4 + 1); in this case, the first three volume serials will be discarded, and the fourth volume would become the first volume allocated.
- The unit count specified in the UNIT parameter (only if the unit count specified is greater than the number of volume serials calculated in the previous statement, or if the request is for a new nonspecific direct access volume that does not specify VOLUME=PRIVATE).

When the number of volume serials numbers calculated for a request is greater than the number of specific volume serial numbers obtained from specified volume serial numbers, VOL=REF, a passed data set, or the catalog, the system assumes that the volumes are requests for nonspecific volumes.

Units per Request

The maximum number of tape units or direct access units required to satisfy any request is equal to the greater of:

- The unit count specified in the UNIT parameter
- The total number of volumes required if parallel mounting is requested

When UNIT=AFF is specified, the unit requirements are obtained from the referenced request. The number of units shared with the referenced request is the number of units used by the referenced request.

The number of units required to satisfy a request specifying a generation data group (GDG) name depends upon the unit requirements of each member of that GDG. Therefore, each member is handled as a single request.

The number of units required to satisfy a VSAM data set depends upon the unit/volume configuration of the data set. If the data set spans multiple device types, the total number of units required is determined by catalog management. Additional tables will then be generated to cause the allocation of the required number of units. For VSAM data sets, a specified unit count or parallel mount may be overridden by the system once the unit requirements for the data set are determined.

If the unit name specified in a request is eligible to several generic device types, allocation for all units requested will be processed to the same device type. The following example, where SYSDA is eligible to 2314s and 3330s, illustrates this point.

```
//DD1 DD UNIT=(SYSDA,2),DSN=A,SPACE=(TRK,(50,1)),DISP=(,CATLG)
```

DD1 will be allocated to either two 2314s or two 3330s.

Units per Job Step

The number of units required for a job step is not necessarily the sum of the unit requirements for each request.

The following rules tend to reduce the total unit requirements for a step:

- A volume can be allocated only to one unit. Therefore, if more than one request asks for the same volume, all requests are allocated the same unit.
- Storage and/or public direct access requests can be allocated on the same volume. Therefore, two or more such requests may be satisfied with one unit.
- For tape, if VOL=REF is specified, more than one public request can be allocated on the same volume. Therefore, two or more public tape requests may be satisfied with one unit.

The following rules tend to increase the total unit requirements for a step:

- A permanently resident or reserved volume cannot be demounted. Therefore, a volume which is permanently resident or reserved will be assigned its own unit (where it is mounted) even if, through JCL specification, it was to share a unit with one or more other volumes.
- When more than one direct access request within a job step requires the same volume, that volume must be shared. Therefore, a direct access volume that is required by more than one request will be assigned its own unit even if, through JCL specification, it was to share a unit with one or more other volumes.
- A VSAM data set will require additional units if the data set resides on more than one device type.
- An additional unit is required for a direct access private catalog volume if it is associated with and/or used to retrieve volume information about a particular data set.
- For direct access, when GDG name is specified, additional units might be required to satisfy the device type requirements of each individual member of the GDG.

- When conflicting unit assignments are specified for tape volumes, the volume involved in the conflict will be assigned its own unit. For example, such a conflict would exist for VOLUM2 in the following DD statements:

```
//DD1    DD  UNIT=2400,VOL=SER=(VOLUM1,VOLUM2)
//DD2    DD  UNIT=2400,VOL=SER=(VOLUM2,VOLUM3)
```

  In this case, three units, one for each volume, would be assigned. If the user had requested via unit affinity that the same tape unit be used for both DD1 and DD2, then only one unit would have been assigned.

Volume Demounting

A demountable, unallocated, private direct access volume is demounted at job unallocation. A demountable, unallocated, private tape volume is demounted at dynamic or step unallocation, unless a passed data set is on the volume or RETAIN was specified in the JCL for the volume. An exception to this occurs when more than one data set is allocated to the same tape volume. In this situation, the volume will be unloaded whether or not the volume contains a passed data set or had RETAIN specified for it.

When a volume must be demounted by allocation to mount another volume, the operator might be told to 'retain' the demounted volume near the system. All private volumes, tape and direct access, are treated in this manner (retained) when demounted by allocation. Whether or not a public volume is retained depends, in part, on whether the volume is direct access or tape. A specification of PASS at any point within a job will cause a public direct access volume to be retained if it is demounted by allocation.

Notes:

- Whenever a tape or direct access volume is demounted, the effect of specifying PASS or RETAIN is lost.
- Each time a volume reference is used in a request for public tape volumes, you must specify whether or not you desire the PASS or RETAIN disposition.
- Coding RETAIN on a DD statement which is setup by JES3 has no effect since the JES3 Main Device Scheduler (MDS) ignores RETAIN.

Chapter 2: Dynamic Allocation Functions

The allocation performed in response to JCL at step allocation (or at LOGON for time-sharing users) may be altered prior to step unallocation (or LOGOFF) by invoking dynamic allocation functions. Because device requirements might not be fully known prior to execution, dynamic allocation functions provide the facility to acquire resources as the need develops. They also allow resources to be used more efficiently, because the resources can be acquired just before use and/or released immediately after use. (The term "resource" means a ddname-data set combination with its attendant volumes and devices, if any.)

"Dynamic allocation functions" refer to the allocation of I/O resources during program execution and all of the related functions of resource allocation performed by SVC 99. These functions include:

- Dynamic allocation — acquire a resource
- Dynamic unallocation — free a resource
- Dynamic concatenation — associate acquired data sets
- Dynamic deconcatenation — separate associated data sets
- Dynamic information retrieval — obtain certain data set information

To avoid confusion between (1) the specific function of dynamically allocating a resource and (2) the set of functions provided by the dynamic allocation routines, the first will be called dynamic allocation and the second, SVC 99 functions, throughout the rest of this book.

A typical use for SVC 99 function is in a program that needs temporary use of a device, volume, or data set for which there is heavy contention. In such a case, dynamic allocation and dynamic unallocation provides the means for a program to tie up the resource for only as long as necessary rather than for the total execution time of the program.

Another common use for SVC 99 functions is in a program whose need for allocation resources varies according to the input. Dynamic allocation and dynamic unallocation permits such programs to dynamically allocate and free only the files necessary to process the input, so the specific resources supporting the required files can be in use for the minimum time.

You request SVC 99 functions with the DYNALLOC macro instruction[1] and the SVC 99 parameter list. You request a specific SVC 99 function via information in the following two fields of the parameter list:

- Verb code – User-supplied one-byte field that describes the function routine being requested. (There are seven verb codes.)
- Key – User-supplied two-byte field that describes the processing being requested from the function routine. (A number of keys are available for each verb code.)

For example, verb code 1 with the key of 2 requests that a new data set be allocated; while verb code 2 with key 7 requests that a particular data set be unallocated. You use other fields in the parameter list to supply the information required to process your request, such as data set disposition or space information.

The functions you request and the information you supply in the SVC 99 parameter list depend, in part, on whether you are coding a program for a batch or time-sharing environment. The next topic describes features of SVC 99 processing that are more applicable to time-sharing environments, but not necessarily to batch environments. You should read the next topic before trying to use SVC 99 functions in either a batch or time-sharing

[1]SVC 99 routines can also be invoked through the dynamic allocation interface routine (DAIR). DAIR; however, does not support all the SVC 99 options; the DAIR interface remains only to provide compatibility. All new requests for SVC 99 functions should use DYNALLOC.

environment. The rest of this chapter describes each of the SVC 99 functions and installation options you can use to control the processing of all SVC 99 requests.

Chapter 3 describes how to request SVC 99 functions, including details on coding the parameter list and information on return codes issued by SVC 99 functions. Chapter 3 also includes an example of a dynamic allocation request.

Understanding Features of SVC 99 Processing

Because of the unpredictable nature of an interactive environment, SVC 99 routines provide some controls that probably are not needed in a batch environment. In addition, dynamic allocation's use of a feature called the *convertible attribute* can be transparent to a batch user.

Controls Designed for a Time-Sharing Environment

Time-sharing command processors use SVC 99 functions to dynamically allocate data sets required for their own processing (for example, work areas), in addition to the data sets the user requests via the time-sharing commands. Because the same command processor can be called again and different command processors might need the same data sets, each command processor does not unallocate the data sets it allocated for its own use. This saves allocation processing that would be required when a subsequent command processor requests the same data sets. However, keeping all data sets allocated until the end of a terminal session can also tie up resources that might no longer be needed. The following features were implemented to avoid tying up resources that are not being used:

- An *in-use bit* for each data set (located in the data set association block (DSAB)). This in-use bit is turned on when a data set is dynamically allocated. In a time-sharing environment, the terminal monitor program (TMP) turns off the in-use bits of all data sets that were dynamically allocated by a command processor when that command processor completes execution. (Turning off the in-use bit does **not** unallocate the data set.) If a subsequent command dynamically requests a previously allocated data set, the in-use bit is turned on again, until the command processor completes execution and returns control to the TMP. In this way, the system can keep track of data sets that have been "not-in-use" for the longest time.
- A *control limit*, which limits the number of data sets that can be allocated but marked "not-in-use" (that is, the in-use bit is turned off). This control limit is determined by the JCL parameter DYNAMNBR on the EXEC statement in the LOGON procedure and the number of DD statements in the LOGON procedure. If a request for a new allocation would cause the control limit to be exceeded, SVC 99 routines automatically attempt to unallocate enough data sets to meet the control limit, starting with eligible data sets that have not been in use for the longest time. If the control limit is still exceeded after all eligible resources have been unallocated, the request for a new allocation fails. In this case, the user must explicitly request unallocation of an existing allocation before the new allocation can be satisfied.
- A *permanently allocated attribute*, which prevents the SVC 99 routines from automatically unallocating a data set to meet the control limit. (In effect, this attribute determines a request's eligibility for automatic unallocation). The permanently allocated attribute is automatically assigned to data sets allocated through JCL and the ALLOCATE command. In addition, you can request that a data set be assigned this attribute (via the SVC 99 parameter list) when you dynamically allocate the data set. For example, the ATTRIBUTE command processor assigns the permanently allocated attribute to a dummy data set which is associated with other attributes specified on the ATTRIBUTE command. The dummy data set must not be automatically unallocated if the control limit is exceeded; it must remain available, to be referenced by a USING keyword on an ALLOCATE command, until the user FREEs the data set.

Note: Because permanently allocated resources are not automatically unallocated and all resources allocated via the ALLOCATE command and JCL are permanently allocated, the control limit limits the number of resources that a terminal user can have allocated at the same time. If no control limit is established, the limit is 1635.

Because a batch application is more predictable than a time-sharing terminal session, the programmer who is using SVC 99 functions in a batch application probably will not need to be concerned with the in-use bit, the control limit, or the permanently allocated attribute.

The Convertible Attribute

Because a data set requested by a command processor might already be allocated, the dynamic allocation routines first check for an existing allocation that matches the current request. This saves redundant allocation processing. In some cases, an existing allocation matches the current request except for some parameters. The dynamic allocation routines can change certain unmatching parameters of the existing allocation to meet the current request if the existing allocation has the *convertible attribute*. The convertible attribute allows the dynamic allocation routines to change the following parameters of the existing allocation: ddname, member name, status, normal and conditional dispositions, space, unallocation at CLOSE, input only, output only, DCB attributes, password, and the permanently allocated attribute.

The convertible attribute is automatically assigned to all data sets dynamically allocated without the permanently allocated attribute. You can; however, assign both the convertible attribute and the permanently allocated attribute to a resource: although you might want to prevent a data set from being automatically unallocated, you might want to allow some of its parameters to be changed to satisfy a new allocation.

Because a batch programmer will probably have no reason to assign the permanently allocated attribute to resources he dynamically allocates, all the resources will have the convertible attribute. If, for example, the programmer dynamically allocates a data set for input only and later allocates the same data set for output only, the dynamic allocation routines will automatically save redundant allocation processing by using the first allocation and changing "input only" to "output only." The setting and use of the convertible attribute can be transparent to the batch programmer.

Dynamic Allocation

You can request one of two types of dynamic allocation: allocation by dsname or allocation by ddname.

Dsname Allocation

Dynamic allocation by dsname is equivalent to data set allocation during job step initiation; the SVC 99 parameter list is equivalent to a DD statement. In the parameter list, you request the allocation-by-dsname function by specifying verb code 1. You can request most of the JCL services that you can code in a DD statement — such as data set disposition, volume label information, expiration date, and SYSOUT destination — by specifying different text units in the parameter list. Figure 4 lists JCL DD statement facilities that **cannot** be used in dynamic allocation. In addition, you can specify the following, which do not have a JCL equivalent, via text units:

- Data set password for password-protected data sets. If you specify the password, the system need not prompt the operator.
- The permanently allocated attribute.

- The convertible attribute.
- Return of certain information.

Restricted DDnames	JOBCAT, STEPCAT, JOBLIB, and STEPLIB	
Keyword Parameters	AMP, BURST, CHARS, CHKPT, DDNAME, DLM, DSID, FLASH, and MODIFY	
Positional Parameters	*, DATA, and DYNAM	
Selected Subparameter of Keywords	**Keyword**	**Subparameter Not Supported**
	COPIES	group values
	DCB	reference to ddname of a previous step CYLOFL NTM RKP
	DISP	PASS specification
	DSN	reference to ddname ISAM area name
	SPACE	ABSTR specification
	UNIT	DEFER specification AFF
	VOLUME	RETAIN specification REF = ddname

Figure 4. Non-Supported JCL DD Statement Facilities

Notes: The user is cautioned to consult the detailed description of each text unit key (see Chapter 3) for the specific capability supported by the key. While a subparameter may be supported (e.g. DCB=DSORG), not all values of that subparameter may be supported (e.g. DCB=DSORG=IS).

The following rules apply to dsname allocation:

- A unique ddname is generated if a ddname is not specified. The ddname created consists of the characters 'SYS' followed by five digits.
- Passwords may be specified as part of a dynamic allocation request to bypass prompting the operator.
- Any data set allocated with a disposition of MOD and for which you don't specify volume information that cannot be found in the catalog is treated as a new data set. If you specify a normal disposition of CATLG, the system catalogs the data set when it is allocated rather than when it is unallocated. If the data set cannot be cataloged, then no allocation will be processed. If the data set cannot be allocated, it will not be cataloged.
- Rather than wait for another user to release a data set, volume, or device in order to obtain use of it, the dynamic allocation routines will fail a request.
- A dynamic allocation request can specify that the ddname, data set name, and volume serial number that are assigned be returned in the SVC 99 parameter list. A user can also request the data set organization (DSORG) of the allocated data set. If a DSORG is specified with the allocation request, that DSORG is returned otherwise, DSORG will be returned as follows:

  - If the allocation request is for a terminal as an I/O device or for a SYSOUT data set, 'PS' is returned as a default value.
  - If the allocation request is a tape data set, 'PS' is returned as a default value.
  - If the allocation request is for a NEW direct access data set, 'PO' will be returned if a directory space quantity was specified; otherwise, 'PS' will be returned.
  - If the allocation request is for an existing direct access data set, the data set organization obtained from the Data Set Control Block (DSCB) is returned. If the organization cannot be obtained from the DSCB, the allocation request is failed.
  - For other types of allocation requests, zeroes will be returned.
- ISAM data sets cannot be created through dynamic allocation.
- For time-sharing users allocating new data sets, (using the TSO ALLOCATE command), DSORG is defaulted to partitioned organization if a directory quantity is specified, or to physical sequential otherwise.
- If the request was eligible to be satisfied by an existing allocation, but no existing allocation of the specified dsname could be used to satisfy the request, the volume and unit information associated with an existing allocation of the specified dsname is copied and associated with the request.

The allocation routines will not use passed data set information to retrieve volume information.

Ddname Allocation

Dynamic allocation by ddname is requested by specifying verb code 6 and the ddname to be allocated in the SVC 99 parameter list. This type of request allows the user to reuse, by specifying only the associated ddname, a previously allocated data set that was marked not-in-use. Ddname allocation processing sets the in-use bit on.

Ddname allocation is useful in time-sharing command processors for re-allocating a group of data sets that were allocated and concatenated by an earlier command processor but whose in-use bits were then turned off by the TMP. Because batch users probably need not be concerned with the in-use bits (that is, no component such as the TMP turns off the in-use bits of data sets dynamically allocated by a batch application), batch users need not use allocation by ddname.

In MVS, the HELP command processor uses ddname allocation to allocate the SYSHELP data set. As a user, you may allocate, in your LOGON procedure or via the ALLOCATE command, a group of data sets to be searched for HELP information. These data sets are concatenated with the ddname of SYSHELP. When the TMP receives control after the LOGON or ALLOCATE command processor completes execution, it turns off the in-use bits of all data sets allocated by the command processor. Then, when you issue the HELP command, the HELP command processor invokes the allocation-by-ddname function, specifying SYSHELP as the ddname. As a result, all of the data sets concatenated to SYSHELP will have the in-use bit turned on. (If the system cannot locate the SYSHELP ddname, the HELP command processor then uses the allocation-by-dsname function to allocate the SYS1.HELP data set and it will be associated with a system-generated ddname.) To satisfy a ddname request, an existing allocation with the specified ddname must:

- Not be in use
- Not have the convertible attribute or must be permanently concatenated, that is, must have properties that ensure that the ddname could not have been disassociated from the existing allocation. (See "The Permanently Concatenated Attribute" in the topic "Dynamic Concatenation" for a description of this attribute.)

If the existing allocation with the specified ddname does not meet the above requirements, or if the ddname is not associated with any existing allocation of the user, the request is failed and an error return is made to the user. If the existing allocation meets the above requirements, it is assigned the in-use attribute and the request has been satisfied. If the existing allocation is a member of a concatenated group, all members of the group are assigned the in-use attribute, so the entire group has been allocated.

The user may specify that an indication be returned if the existing allocation that satisfies the request is associated with a DUMMY data set.

Dynamic Unallocation

The dynamic unallocation routines unallocate resources when requested via the appropriate verb code. Two functions are available through dynamic unallocation:

- Releasing a data set, specified by verb code 2 with key 7 in the SVC 99 parameter list. This facility involves the following processes:
  - Disassociating the ddname from the data set name, which allows the ddname to be used in subsequent dynamic allocations
  - Process the data set disposition
  - Releasing the data set for use by other jobs
  - Freeing the unit(s) to which the data set was allocated
  - Releasing the volume(s) on which the data set was allocated
- Removing the in-use attribute, specified by verb code 2 with key 8 in the SVC 99 parameter list. This function turns off the in-use bit of the data set. When this processing is completed, the data set is referred to as "not-in-use." You can also request remove in-use processing based on task-id by specifying verb code 5 -- see the following topic, "Identifying a Resources by Task-Id."

If you code verb code 2 without specifying key 7 or key 8, the dynamic unallocation routines will remove the in-use attribute from resources allocated through JCL, through the time-sharing ALLOCATE command, or dynamically with the permanently allocated option; the routines release data sets allocated dynamically without the permanently allocated option. However, a user may explicitly specify the type of processing to be performed by also specifying key 7 or key 8. The explicit specification will be satisfied in all but one case— the in-use attribute will not be removed from non-permanently allocated, non-&dsname data sets with a disposition of DELETE, because such a resource cannot be used to satisfy a subsequent request. Such a resource will be released.

Either dynamic unallocation function can be performed for a dsname or a ddname.

Unallocation Processing

The following rules apply to all dynamic unallocation requests:

- If a data set is open, is a member of an open concatenated group, or is a private catalog, it will not be unallocated.
- When a SYSOUT data set is released, it is immediately made available for output (unless you specify an overriding disposition of DELETE, in which case the data set is deleted).

The following rules apply to unallocation requests that specify dsname:

- If no ddname is specified and the dsname is associated with more than one ddname, all associations are unallocated. If an error occurs while unallocating one ddname, processing continues for the others and an error code is returned. If errors occur for more than one ddname, the error code applies to the last ddname for which there was an error.

- If a membername is specified with the dsname, only those associations containing both the membername and dsname are unallocated. (Both the membername and dsname keys must be coded for a valid request.)

The following rules apply to unallocation requests that specify a ddname:

- Only the occurrence of the data set associated with the specified ddname is unallocated, even if that data set is associated with other ddnames.
- If a dsname or dsname and membername are specified in addition to the ddname, they must be associated with that ddname or the request fails.

Concatenated Groups: If the specified resource is associated with a permanently concatenated group, the in-use attribute is removed from all members of the group, and the count of the number of resources held for reuse is increased by the number of members in the group. (An exception is when the concatenated group was generated by the system, that is, VSAM data sets spanning device types and GDG ALL groups. In these cases, the group is treated as a single resource.)

If the concatenated group does not have the permanently concatenated attribute, the group is deconcatenated and the member associated with the specified dsname is released. (The first member is released if the group's ddname is specified.)

If a concatenated group has the permanently concatenated attribute and you specify a ddname with a dsname, a VSAM dsname, or GDG ALL the entire group is released. If you specify a dsname with a VSAM dsname or GDG ALL, the request for unallocation fails.

Changing the Parameters at Dynamic Unallocation: If a request for remove in-use processing also includes text units to change the following, the remove in-use processing will be honored but the changes will not be made until the resource is released (verb code 2 with key 7):

- Output class
- HOLD/NOHOLD parameters
- Remote work station destination
- Disposition

Allocation disposition cannot be overridden for passed data sets, VSAM data sets, and system-named data sets. The disposition for all other types of data set during unallocation processing.

Members of partitioned data sets cannot be deleted with a disposition of DELETE; the entire data set is deleted. An overriding disposition of DELETE for data sets allocated as shared is invalid; the overriding disposition request is failed.

Identifying a Resource by Task-Id

In addition to specifying a ddname or dsname, the user may specify via verb code 5, that the in-use attribute be removed based on task-id. The attribute may be removed from all resources associated with a specified task, or all resources except those associated with the current task, its higher-level tasks, and the initiator. This function is used by the time-sharing terminal monitor program (TMP) to remove the in-use attribute from any data sets allocated by a command processor when a command processor completes execution.

Dynamic Concatenation

Dynamic concatenation logically connects allocated data sets into a concatenated group. You request this facility by specifying verb code 3 in the SVC 99 parameter list. You can identify data sets to be concatenated only by their associated ddnames. These data sets must not be open or the request for dynamic concatenation will fail.

The order of the data sets in the concatenated group will be the order in which the associated ddnames are specified. The name associated with the concatenated group will be the ddname that was specified first. The other ddnames are no longer associated with any data set. If a specified ddname is already associated with a concatenated group, that group will be included in the new concatenation.

After the request for dynamic concatenation is satisfied, all members of the dynamically concatenated group are assigned the in-use attribute.

The Permanently Concatenated Attribute

The permanently concatenated attribute may be assigned when concatenation is requested. In addition, a concatenated group defined via JCL is automatically assigned the permanently concatenated attribute. Step and dynamic allocation requests that result in a concatenated group defined by the system are also automatically assigned this attribute. A GDG ALL request and a request for a VSAM data set that spans device types are examples of such requests.

A group having the permanently concatenated attribute has the following characteristics:

- To dynamically release a non-system-defined permanently concatenated group, you must specify the ddname, not the dsname, in the unallocation request.
- The group cannot be dynamically deconcatenated into its member data sets.
- If a permanently concatenated group is dynamically concatenated with other data sets to form a new non-permanently concatenated group, the permanently concatenated group remains intact if the new group is dynamically deconcatenated.
- If the group is not a system-defined permanently concatenated group, the group is automatically assigned the permanently allocated attribute.

Dynamic Deconcatenation

Dynamic deconcatenation logically disconnects the members of a concatenated group. You request dynamic deconcatenation by specifying verb code 4 in the SVC 99 parameter list. You identify the concatenated group to be deconcatenated by specifying the ddname of the group.

The request for dynamic deconcatenation fails if the concatenated group is open. A permanently concatenated group, or members of a concatenated group that are permanently concatenated, remain concatenated.

When a concatenated group is dynamically deconcatenated, the ddnames that were associated with the data sets before they were concatenated are restored unless this would result in duplicate ddnames. This situation could arise if a dynamic allocation with the ddname to be restored occurred after a dynamic concatenation. In this case the deconcatenation request fails.

Dynamic deconcatenation has no effect on the in-use attributes associated with the members of the group.

Dynamic Information Retrieval

Dynamic information retrieval provides you with information about your current allocation environment. You request this facility by specifying verb code 7 in the SVC 99 parameter list. You can request information about ddnames or dsnames. In addition, you may ask for information about any or all of your currently allocated requests by specifying a relative request number.

For example, information about all requests can be obtained by successively asking for information about the 1st, 2nd, ...nth allocation request. A unique return code is provided when information is requested for a nonexistent relative entry.

The following information can be requested:

- Data set name
- Ddname
- Member name
- Data set organization
- Status
- Normal disposition
- Conditional disposition
- Attribute status, including the permanently allocated, in-use, permanently concatenated, convertible, or dynamically allocated attributes
- Data set types, including dummy, SYSIN, SYSOUT, and allocation of the user's terminal as an I/O device
- The number of resources held in anticipation of reuse that exceeds the control value, that is, the number of existing allocations that must be unallocated before a request for a new allocation can be satisfied
- Whether or not the allocation is the last relative request (that is, by specifying successive relative request numbers (1st, 2nd, 3rd, etc.), a unique return code is provided when the relative request given is the last.)

Installation Options

This section describes the values and options your installation might desire to modify in order to control SVC 99 processing. The values and options being described in the following topics are: default values for space and unit information, volume mounting and bringing devices online, and the writing of installation validation routines.

Space and Unit Defaults

This section describes how to change the allocation default values for space and unit. It also describes the remote user work station defaulting.

If *space information* is not specified in a request for a new direct access data set and the request is eligible to MSS exclusively, with the MSVGP specified, the MSVGP space defaults are used. If the request if not eligible to MSS exclusively or MSVGP is not specified, a default of 1000 block length, 10 primary blocks, and 50 secondary blocks with the release unused space (RLSE) is used. These space defaults are contained in the allocation default CSECT, IEFAB445 (a member of load module IEFW21SD), so the installation may modify them. The contents of the module are (beginning at offset zero; the parenthesized numbers show the bit setting supplied by IBM):

- Three bytes for the binary value of primary quantity (X'00000A')
- Three bytes for the binary value of secondary quantity (X'000003')
- Three bytes for the binary value of the average block length (X'0003E8')
- Three bytes for the binary value of the number of directory blocks (X'000000')
- One byte of flags with the following bit meanings:
  - bit 0 TRK (0)
  - bit 1 CYL (0)

- bit 2 blocklength (1)
- bit 3 RLSE (1)

- bit 4 CONTIG (0)
- bit 5 MXIG (0)
- bit 6 ALX (0)
- bit 7 ROUND (0)

If *unit information* is not specified, a unit description is obtained from a time-sharing user's UADS entry. If the user is not a time-sharing user, or if the UADS entry does not contain a unit description, a default of 'SYSALLDA', that is, all direct access devices, is used. This default is contained in the allocation default CSECT IEFAB445, in the eight bytes beginning at offset 13 (decimal). The unit description supplied is eligible to override the unit type for a cataloged data set; however, the default unit description for the UADS is not eligible for unit override.

If the *remote user workstation* (or *destination)* is not specified by time-sharing users allocating a SYSOUT data set, this value is defaulted from the time-sharing user's UADS entry.

Volume Mounting and Bring Devices Online

Dynamic allocation can bring *devices online and have volumes mounted.* Because this is a time-consuming operation and requires operator communication, and therefore is not always desirable in an interactive environment, this function is an option for time-sharing users. This option is assigned via the UADS entries. Non-time-sharing users always have volume mounting ability and the ability to have devices brought online. However, any user may indicate in the SVC 99 parameter list that volumes are not to be mounted and that devices are not to be brought online for a request.

The operator may inform the dynamic allocation routines that a volume is not to be mounted or that a device is not to be brought online. In this case, the request is failed. In order to support this operator communication, dynamic allocation must wait for tape volumes to be mounted. (Step allocation does not wait for tape volumes to be mounted.) When the volume is mounted, OPEN verifies that the correct volume has been mounted.

If the option to have volumes mounted and devices brought online is not in effect, tape and direct access devices that have an outstanding mount request or that are not ready are not eligible for use by dynamic allocation.

Installation Input Validation Routine

An exit (IEFDB401) from the allocation control routine provides for a user-written routine to validate or alter any request to SVC 99. The routine is entered for all system and user SVC 99 requests. The routine must be coded so as not to interfere with system requests.

The validation routine may test and modify the SVC 99 input request, and it may indicate through a return code whether processing of the request is to continue. For example, the routine may perform the following functions:

- Control the amount of direct access space requested
- Check for authorization to use specified units
- Check for authorization to use certain data sets
- Check for authorization to hold certain resources for reuse

Programming Considerations

The input validation routine must observe the following programming conventions and must receive the following input:

- Its CSECT name must be IEFDB401 and it must reside in load module IEFW21SD.
- The routine must use register 15 to return to SVC 99 a code of zero if processing of the request is to continue, or any other code if processing is to terminate.
- It receives control in supervisor state under the scheduler's protection key (key 1). At entry, register 1 points to a list of addresses for the following parameters:
  - A copy of the SVC 99 input request block, text unit pointers, and text units in scheduler-key fetch-protected storage.
  - The address of a work area for the use of the routine. This area is contiguous with the text unit pointer list so that it can be used to extend the list and provide additional text units.
  - A fullword that contains the length of the work area (500 bytes).
  - The eight-character job name
  - The twenty-byte programmer name
  - An area that contains accounting information from the JOB statement. The first byte of this area contains the number of accounting fields; the accounting fields follow this byte. Each entry for an accounting field contains the length of the field (one byte, binary), followed by the field itself. The entry for a null field contains a length of zero.
  - The eight-character step name
  - The eight character program name
  - An area containing accounting information from the EXEC statement. The first byte of this area contains the number of accounting fields (0 for no fields); the accounting fields follow this byte. Each entry for an accounting field contains the length of the field (one byte, binary), followed by the field itself. The entry for a null field contains a length of zero.

The IBM-supplied routine that your routine may replace allows all requests to continue processing.

Chapter 3: Requesting SVC 99 Functions

To request an SVC 99 function, you must code the DYNALLOC macro instruction (it has no operands) and supply the SVC 99 parameter list. This chapter describes the programming considerations for using SVC 99 functions, the SVC 99 parameter list, and SVC 99 return codes. It also includes an example of a request for dynamic allocation.

Programming Considerations When Using SVC 99 Functions

Before deciding to use any of the SVC 99 functions, you should consider the environment of the program that is going to invoke SVC 99. For example:

- Consider the fact that your program might serialize the same resources as SVC 99. For example, the following resources are serialized, depending on the path taken in SVC 99 processing.

Major Name	Minor Name
SYSDSN	data set name
SYSIEFSD	CHNGDEVS
SYSIEFSD	DDRDA
SYSIEFSD	DDRTPUR
SYSIEFSD	Q4
SYSZOPEN	data set name
SYSZPCCB	PCCB
SYSZTIOT	address of the DSAB QCB.asid
SYSZVMV	ucbaddr
SYSZVOLS	volume serial number

- System routines invoked by various paths of SVC 99 processing might also serialize a system resource. Some of the system functions invoked by SVC 99 processing are LOCATE, OBTAIN, CATALOG, SCRATCH, DADSM, and allocate (SVC 126).
- Avoid use of SVC 99 functions in routines that run under the control of an interruption request block (IRB), especially when that program might be interrupted issues OPEN, OPENJ, CLOSE, EOV, or FEOV. An SVC 99 request issued in such an environment can cause a 138 ABEND when SVC 99 tries to enqueue on the SYSZTIOT resource.
- The program that issues SVC 99 functions must not receive control when the job entry subsystem is being started or an unending wait could result, causing the system to crash.
- The program that issues SVC 99 must not have an STIMER macro outstanding, because certain paths in SVC 99 issue another STIMER, which causes an overlay of the first STIMER. This situation causes the program that issued the SVC 99 request never to receive control because of the expiration of the timer.
- The system routines that invoke SVC 99 functions should be aware that a non-zero return code might be returned. System routines cannot always diagnose non-zero return codes. To aid serviceability, it is suggested that the system routine print an error message including the error code (S99ERROR) and information code (S99INFO) fields in the SVC 99 request block (S99RB).
- Routines should not allocate data sets that are cataloged in OS CVOLs or VSAM private catalogs to long-running tasks, since the private catalog or CVOL will be allocated and will remain allocated until the step terminates. This is especially important in installation exits for system tasks since the private catalog or CVOL might be allocated to an initiator or subsystem such as JES2 or JES3.
- Volumes that contain a CVOL or VSAM private catalog for data sets allocated to long-running steps should be assigned the permanently resident attribute.

SVC 99 Parameter List

When you code DYNALLOC, you must supply a parameter list, which includes a request block, text pointers, and text units. Figure 5 illustrates the structure of the SVC 99 parameter list. IBM supplies two macros IEFZB4D0 and IEFZB4D2 to aid in constructing the SVC 99 parameter list. IEFZB4D0 provides symbolic names (dummy sections) for the positional information in the structure; IEFZB4D2 provides mnemonics for the text unit keyword values. The names in Figure 5 are those assigned by the macro IEFZB4D0.

Register 1 must point to a pointer to the request block. The text pointers in the request block and the text units must be created in real storage (storage obtained via the GETMAIN macro instruction). Upon entry to SVC 99, these fields are copied into the SVC's workarea and, before exit from SVC 99, these fields are restored into the real storage area. This copy and restore function is always performed. An 0C4 ABEND will occur if the parameter list pointer, the request block pointer, or the text unit pointers contain addresses of real storage not owned by the caller.

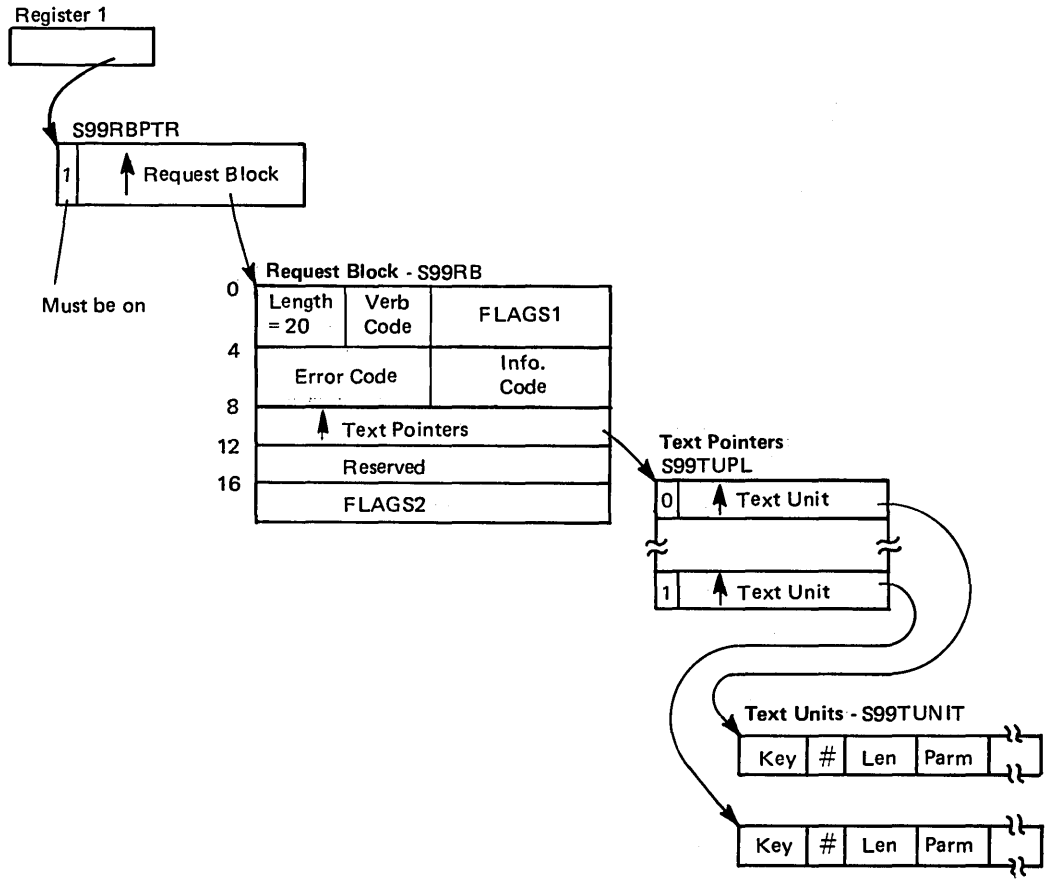


Figure 5. Structure of the SVC 99 Parameter List

Request Block

The request block must begin on a fullword boundary. It contains the following fields (the names in parentheses are those assigned by the mapping macro IEFZB4D0):

- LENGTH (S99RBLN) This field is a one-byte field that contains the length of the request block. The length is always 20 bytes.
- VERB CODE (S99VERB) The verb code is a one-byte field that identifies the SVC 99 function to be performed. The following codes may be specified:

Verb Code	Name	Meaning
01	S99VRBAL	Request for dsname allocation
02	S99VRBUN	Request for unallocation (based on dsname or ddname)
03	S99VRBCC	Request for concatenation
04	S99VRBDC	Request for deconcatenation
05	S99VRBRI	Request for removing the in-use attribute based on task-id
06	S99VRBDN	Request for ddname allocation
07	S99VRBIN	Request for information retrieval

- FLAGS1 (S99FLAG1)The FLAGS1 field is a two-byte field that instructs the system on how to satisfy dsname allocation requests. The meaning of the bits in the field are as follows:

Bit	Bit Name	Meaning When On
0	S99ONCVN	Do not use an existing allocation that does not have the convertible attribute to satisfy the request.
1	S99NOCNV	Do not use an existing allocation to satisfy this request.
2	S99NOMNT	Do not mount volumes or consider offline units. (This bit overrides S99MOUNT and S99OFFLN in FLAGS2.) If this bit is one and the request causes a private catalog to be allocated, mounting will not be allowed for that catalog.
3	S99JBSYS	Treat the data set as part of the job's normal output. The data set is not expected to be dynamically unallocated (spun off). This flag is used for SYSOUT data sets. If the data set is dynamically unallocated the data set will be printed immediately but paging space will not be released until the job ends.
4	S99CNENQ	Issue a conditional ENQ on the TIOT resource. If not available, an error code is returned to user.
5-16		Reserved; must be zero

Note: The FLAG2 indicators are used only for dsname allocation requests.

- ERROR CODE (S99ERROR) This field is a two-byte field that SVC 99 uses to return error reason codes. See the topic "SVC 99 Return Codes."
- INFO CODE (S99INFO) This two-byte field is used by SVC 99 to return information reason codes. See the topic "SVC 99 Return Codes."
- TEXT POINTERS ADDRESS (S99TXTPP) This fullword field contains the address of a list of pointers to the text units.
- RESERVED An area, one fullword in length, that contains zeros.
- FLAGS2 (S99FLAG2) The FLAGS2 field is a four-byte field of indicators. These indicators may be set only by authorized programs. To be authorized, the requesting program must meet at least one of the following criteria:
  - It must have a system storage protection key (0-7).
  - It must be in supervisor state.
  - It must be in APF authorized.

The meanings of the bits are:

Bit	Bit Name	Meaning When On
0	S99WTVOL	Wait for volumes.
1	S99WTDSN	Wait for dsname.
2	S99NORES	Do not reserve data sets.
3	S99WTUNT	Wait for units.
4*	S99OFFLN	Consider offline devices. If S99NOMNT in FLAGS1 is off and this bit is on for a background job or if a time-sharing user does not have the mount attribute in his UADS entry, the system will consider offline devices. This bit is ignored if S99NOMNT is on or if the time-sharing user hs the mount attribute in his UADS entry.
5	S99TIONQ	TIOT ENQ already performed.
6	S99CATLG	Set special catalog data set indicators.
7*	S99MOUNT	Volumes may be mounted. If S99NOMNT in FLAGS1 is off and this bit is on for a background job or if a time-sharing user does not have the mount attribute in his UADS entry, the system will allow volumes to be mounted. This bit is ignored if S99NOMNT is on or if the time-sharing user has the mount attribute in his UADS entry.
8	S99UDEVT	Unitname parameter is a device type.
9	S99PCINT	Allocate a private catalog on behalf of the initiator.
10-31		Reserved. Must be zero.

*These fields override the NOMOUNT option from the user attribute data set (UADS) for TSO users.

Text Pointers

The text pointer part of the parameter list is a variable-length list of fullword pointers to text units. The end of the list is indicated by setting on the high-order bit of the last pointer. A fullword of zeros is ignored. Mapping macro IEF2B4DO assigns the label S99TUPL to the list, the label S99TUPTR to each pointer in the list, an label S99TUPLN to an equate that allows you to turn on the end-of-list indicator.

Text Units

Each text unit is a variable-length field (labeled S99UNIT by macro IEFZB4D0) that contains the following subfields:

- KEY (S99KEY) A two-byte field that contains a unique binary number that identifies the type of information to be found in the PARM subfield. For example, a key of '0004' for a dsname allocation request indicates that the value of the PARM subfield specifies data set status. SVC 99 ignores a KEY field of zero. Each SVC 99 function has an associated set of text units, and each set is independent of any other. For example, the functions of both allocation and unallocation may use a KEY value of '0007' but that value does not necessarily have the same meaning for each function. See the topic "Text Units by Function" for a description of the text units that can be coded for each SVC 99 function.
- NUMBER (S99TUNUM) A two-byte binary number that specifies the number of length and parameter combinations in the text unit. If a key of zero is specified, S99TUNUM must also be zero.
- COMBINATION (S99UENT) The label for a length an parameter combination. IEFZB4D0 provides the following DSECT for use when specifying multiple parameters in a single text unit. This DSECT places the length field at zero displacement for the second an subsequent combinations:

S99TUFLD	Label for the DSECT
S99TULEN	Label for the length field
S99TUPRM	Label for the parameter

- LENGTH (S99TULNG) A two-byte binary number that specifies the length of the following parameter field.
- PARM (S99TUPAR) This field contains a value that provides the parameter information identified by the KEY field. See "Text Units by Function" for a description of the values that can be coded for each key.

SVC 99 Return Codes

Note: The labels used in this topic are assigned by macros IEFZB4D0 and IEFZB4D2.

When the SVC 99 routines return control to the requesting program, register 15 contains a return code. Depending on the return code, the S99ERROR and S99INFO fields in the input request block (S99RB) may additionally contain error and information reason codes respectively. The return codes that can be returned in register 15 are shown in Figure 6.

Code	Meaning
0	Successful completion; there will also be an information reason code if a non-terminating error occurred during request processing.
4	An error resulted from the current environment, the unavailability of a system resource, or a system routine failure; there will also be an error reason code.
8	The installation validation routine denied this request. (See "Installation Input Validation Routine" for additional information.)
12	The error is due to an invalid parameter list; there will also be an error reason code from class 3. (Class 3 reason codes are listed in Figure 7.)

Figure 6. SVC 99 Return Codes

The DAIRFAIL TSO service routine can be used to issue write-to-programmer of TSO PUTLINE failure messages for both DAIR and SVC 99 error codes. Refer to "OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor" for information on using DAIR and DAIRFAIL.

The next two topics describe the information and error reason codes that can be returned in the SVC 99 request block. The last topic in this chapter describes dsname allocation processing in detail; you might need this information to understand the error and information codes returned by dsname allocation.

Information Reason Codes

The codes below are returned in the two-byte field (S99INFO) in the request block.

Code	Meaning
0004	Reserved
0008	Overriding disposition ignored for one of the following reasons: • Data set was originally allocated with a disposition of PASS • Data set is a non-subsystem data set that has a system-generated name; you cannot override disposition on this type of data set • Data set is a VSAM data set In these cases, the data set is unallocated using the disposition specified when the request was allocated.
000C-001C	Reserved
002n	The data set was successfully unallocated but processing of the requested CATLG or UNCATLG disposition was unsuccessful. The digit "n" is a code representing the reason for the failure. Below is a list of the possible codes and meanings.

Code	Meaning
1	A control volume was required and a utility program must be used to catalog the data set.
2	The data set to be cataloged had previously been cataloged, or the data set to be uncataloged could not be located, or no change was made to the volume serial list of a data set with a disposition of CATLG.
3	The specified index did not exist.
4	The data set could not be cataloged because the space was not available in the catalog.
5	Not enough storage was available to perform the specified cataloging.
6	The data set to be cataloged in a generation index is improperly named.
7	The data set to be cataloged had not been opened and no density information was provided (for dual density tape requests only).
8	Reserved
9	An uncorrectable I/O error occurred in reading or writing the catalog.

003n	The data set was successfully unallocated but processing of requested DELETE disposition was unsuccessful. The digit "n" code represents the reason for the failure. Following is a list of the possible codes and their meanings.

Code	Meaning
1	The expiration date had not occurred.
2	Reserved
3	Reserved
4	No device was available for mounting for the volume during deletion.
5	Not enough storage was available to perform the specified deletion.
6	Either no volumes were mounted or volumes that were mounted could not be demounted to permit the remaining volumes to be mounted.
8	The SCRATCH routine returned an error code. If the user's JOB statement requested allocation/termination messages, message IEF283I will appear in the SYSOUT listing. This message will list the volume serial numbers of the data sets that were not deleted; following each number will be a code that explains why each data set was not deleted.

Error Reason Codes

Error reason codes are divided into the following classes:

Class	Description
1	Reserved
2	Unavailable system resource
3	Invalid parameter list
4	Environmental error
5	Reserved
6	Reserved
7	System routine error

The error reason codes are contains the codes shown in Figure 7. The second hexadecimal digit will be one of the class designations above. The error code field in the SVC 99 request block is labeled S99ERROR.

Note: The explanations of the codes in Figure 7 are followed by an indication of the kind of request associated with the code.

CLASS 2 CODES

Code	Meaning
0204	Real storage unavailable (dsname allocation).
0208	Reserved.
020C	Request for exclusive use of a shared data set cannot be honored (dsname allocation)
0210	Requested data set unavailable. The data set is allocated to another job and its usage attribute conflicts with this request. (dsname allocation)
0214	Unit(s) not available (dsname allocation)
0218	Specified volume or an acceptable volume is not mounted, and user does not have volume mounting authorization. (dsname allocation)
021C	Unit name specified is undefined. (dsname allocation)
0220	Requested volume not available. (dsname allocation)[3]
0224	Eligible device types do not contain enough units. (dsname allocation)
0228	Specified volume or unit in use by system. (dsname allocation)
022C	Volume mounted on ineligible permanently resident or reserved unit. (dsname allocation)
0230	Permanently resident or reserved volume on required unit. (dsname allocation)
0234	More than one device required for a request specifying a specific unit. (dsname allocation)
0238	Space unavailable in Task Input Output Table (TIOT). (dsname allocation, concatenation)
023C	Required catalog not mounted, and user does not have volume mounting authorization. (dsname allocation)
0240	Requested device is a console. (dsname allocation)
0244	Telecommunication device not accessible. (dsname allocation)
0248	MSS virtual volume unable to be mounted. (dsname allocation)
024C	Operating system managed resource was unavailable to the subsystem. (dsname allocation) [8]
0250	Subsystem resource not available. (dsname allocation [8]
0254	The TIOT resource is currently unavailable and the user requested conditional ENQ on the resource. (all SVC 99 functions)

Note: The failing system routine returns the code represented by "zz".

1 The informational reason code field contains 0004 if the requested function was performed, although an error occurred as the error reason code indicates.
2 The informational reason code contains the value of the key that caused the error.
3 For MSS requests, the MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages**. For non-MSS request this code will be accompanied by IEF485I, or it may result from a JES3 failure because of a busy or unavailable situation.
4 The MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
5 This code corresponds to MSSC reason code X'007', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
6 This code corresponds to MSSC reason code X'207', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
7 The informational reason code field contains the MSSC reason code. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
8 The information reason code contains a subsystem defined value to further describe the error. This value is documented in publications associated with the particular subsystem.

Figure 7. Error Reason Codes (Part 1 of 5)

CLASS 3 CODES

Code	Meaning
0304-0338	Assigned by DAIR. (See **OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor.**)
033C-0354	Reserved.
0358	Overriding disposition of DELETE invalid for data set allocated as SHR. (unallocation)[1]
035C	Invalid PARM specified in text unit. (all SVC 99 functions)[2]
0360	Invalid KEY specified in text unit. (all SVC 99 functions)[2]
0364	JOBLIB/STEPLIB/JOBCAT/STEPCAT specified as ddname, or associated with specified dsname; dsname allocation, ddname allocation, unallocation, concatenation, deconcatenation[1]
0368	Authorized function requested by unauthorized user. (all SVC 99 functions)
036C	Invalid parameter list format. (all SVC 99 functions)
0370	Reserved.
0374	Invalid # specified in text unit. (all SVC 99 functions)[2]
0378	Duplicate KEY specified in text unit. (all SVC 99 functions)[2]
037C	Invalid LEN specified in text unit. (all SVC 99 functions)[2]
0380	Mutually exclusive KEY specified. Two keys that cannot be used together were used in the text unit. (dsname allocation, unallocation, information retrieval, remove in-processing)[2]
0384	Mutually inclusive KEY not specified. One key was used; two should have been used. (unallocation, dsname allocation)[2]
0388	Required key not specified. (ddname allocation, information retrieval, concatenation, deconcatenation, remove in-processing, unallocation)
038C	Duplicate ddnames specified. (concatenation)
0390	GDG group name specified with relative generation number exceeds 35 characters. (dsname allocation)
0394	Status and relative generation number are incompatible. (dsname allocation)
0398	Volume sequence number exceeds the number of volumes. (dsname allocation)
039C	Device type and volume are incompatible. (dsname allocation)
03A0	Subsystem detected an invalid parameter. (dsname allocation)[8]
03A4	Unable to PROTECT data set/volume because of conflicting keyword specification.

Note: The failing system routine returns the code represented by "zz".

1 The informational reason code field contains 0004 if the requested function was performed, although an error occurred, as the error reason code indicates.
2 The informational reason code contains the value of the key that caused the error.
3 For MSS requests, the MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
4 The MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
5 This code corresponds to MSSC reason code X'007', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
6 This code corresponds to MSSC reason code X'207', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
7 The informational reason code field contains the MSSC reason code. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
8 The information reason code contains a subsystem defined value to further describe the error. This value is documented in publications associated with the particular subsystem.

Figure 7. Error Reason Codes (Part 2 of 5)

CLASS 4 CODES

Code	Meaning
0404-040C	Reserved.
0410	Specified ddname unavailable. (dsname allocation, ddname allocation)
0414-041C	Reserved.
0420	Specified ddname or dsname associated with an OPEN data set. (ddname allocation, concatenation, deconcatenation, unallocation, dsname allocation)[1]
0424	Deconcatenation would result in duplicate ddnames (deconcatenation).[1]
0428-0430	Reserved.
0434	Ddname specified in ddname allocation request is associated with a convertible or non-permanently allocated resource. (ddname allocation)
0438	Specified ddname not found. (information retrieval, ddname allocation, concatenation, deconcatenation, unallocation)
043C	Resources could not be unallocated to decrease the number of resources held in anticipation or reuse to meet the limit of the control value. (dsname allocation)
0440	Specified dsname not found. (information retrieval, unallocation)
0444	Relative entry number specified in information retrieval request not found. (information retrieval)
0448	Request for a new data set failed; the data set already exists. (dsname allocation)
044C	Request was made for a data set that has a disposition of delete; this request cannot be honored because the data set may be deleted at any time. (dsname allocation)
0450	Request would cause the limit of 1635 concurrent allocations to be exceeded. (dsname allocation)
0454	Ddname in DCB reference not found. (dsname allocation)
0458	Dsname in DCB reference or volume reference is a GDG group name. (dsname allocation)
045C	Specified dsname to be unallocated is a member of permanently concatenated group. (unallocation)[1]
0460	Specified dsname or member to be unallocated is not associated with specified ddname. (unallocation)

Note: The failing system routine returns the code represented by "zz".

1 The informational reason code field contains 0004 if the requested function was performed, although an error occurred as the error reason code indicates.
2 The informational reason code contains the value of the key that caused the error.
3 For MSS requests, the MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
4 The MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
5 This code corresponds to MSSC reason code X'007', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
6 This code corresponds to MSSC reason code X'207', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
7 The informational reason code field contains the MSSC reason code. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
8 The information reason code contains a subsystem defined value to further describe the error. This value is documented in publications associated with the particular subsystem.

Figure 7. Error Reason Codes (Part 3 of 5)

CLASS 4 CODES

Code	Meaning
0464	Specified dsname to be unallocated is a private catalog. (unallocation)[1]
0468	Error while allocating or opening a private catalog. (allocation)
046C	Remote workstation not defined to Job Entry Subsystem. (dsname allocation, unallocation)
0470	User unauthorized for Job Entry Subsystem request. (dsname allocation)
0474	Error while attempting to select optimum device. (dsname allocation).
0478	Unable to process Job Entry Subsystem request. (dsname allocation, unallocation)
047C	Unable to establish ESTAE environment. (all SVC 99 functions)
0480	The number of units to satisfy the request exceeds the limit. (dsname allocation)
0484	Request denied by operator. (dsname allocation)
0488	GDG pattern DSCB not mounted. (dsname allocation)
048C	GDG pattern DSCB not found. (dsname allocation)
0490	Error changing allocation assignments. (dsname allocation)
0494	Error processing OS CVOL. (dsname allocation)
0498	MSS virtual volume not accessible. (dsname allocation)[4]
049C	MSS virtual volume not defined. (dsname allocation)[5]
04A0	Specified MSVGP name not defined. (dsname allocation)[6]
04A4	Subsystem request in error. (dsname allocation)[8]
04A8	Subsystem does not support allocation via key DALSSNM. (dsname allocation)
04AC	Subsystem is not operational.
04B0	Subsystem does not exist.
04B4	PROTECT not processed; RACF not in system or not active.
04B8	MSS not initialized for allocation. (dsname allocation)
04BC	MSS volume select error. (dsname allocation)[7]
04C4	The last request was for a VOL = REF to a dsname or DCB = dsname which exceeded the maximum allowable dsname referbacks. (A maximum of 972 referbacks are allowed if the data set names are 44 characters in length.)

Note: The failing system routine returns the code represented by "zz".

1 The informational reason code field contains 0004 if the requested function was performed, although an error occurred as the error reason code indicates.

2 The informational reason code contains the value of the key that caused the error.

3 For MSS requests, the MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**

4 The MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**

5 This code corresponds to MSSC reason code X'007', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**

6 This code corresponds to MSSC reason code X'207', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**

7 The informational reason code field contains the MSSC reason code. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**

8 The information reason code contains a subsystem defined value to further describe the error. This value is documented in publications associated with the particular subsystem.

Figure 7. Error Reason Codes (Part 4 of 5)

CLASS 7 CODES

Code	Meaning
17zz	LOCATE error; dsname allocation. (Note: Hexadecimal '08', '18', and '2C' are the only expected LOCATE return codes. 'FF' is returned as the value of zz if an unexpected return code is returned by LOCATE)[9]
27zz	Reserved
37zz	Reserved
47zz	DADSM error. (dsname allocation)[9]
57zz	CATALOG error. (dsname allocation)[9]
67zz	OBTAIN error. (dsname allocation, information retrieval)[9]
7700	Subsystem error. (dsname allocation)[8]
7704	A subsystem interface system error occurred while processing key DALSSNM

Note: The failing system routine returns the code represented by "zz".

1 The informational reason code field contains 0004 if the requested function was performed, although an error occurred as the error reason code indicates.
2 The informational reason code contains the value of the key that caused the error.
3 For MSS requests, the MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
4 The MSSC reason code for this failing job step is contained in message IEF710I on the hardcopy log. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
5 This code corresponds to MSSC reason code X'007', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
6 This code corresponds to MSSC reason code X'207', which is explained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
7 The informational reason code field contains the MSSC reason code. An explanation of the MSSC reason code is contained in **OS/VS Message Library: Mass Storage System (MSS) Messages.**
8 The information reason code contains a subsystem defined value to further describe the error. This value is documented in publications associated with the particular subsystem.
9 For LOCATE, DADSM, CATALOG, and OBTAIN return code detailed description see **OS/VS2 MVS System Programming Library: Data Management.**

Figure 7. Error Reason Codes (Part 5 of 5)

Details on DSNAME Allocation Processing

When the dynamic allocation function is invoked for dsname allocation, an "allocation environment" already exists for the user. This allocation environment consists of the user's step and dynamic allocation requests that have not been dynamically unallocated. SVC 99 considers these allocations "existing allocations."

For dsname allocation, the dynamic allocation routines first check for environmental conflicts by noting the types of resources that are currently available to the task, then try to satisfy the request with an existing allocation that matches or can be made to match the request using the flexibility of the convertible attribute. If the routines are not able to make the match, a new allocation is processed. However, if an existing allocation can be used, much allocation processing is avoided.

Checking for Environmental Conflicts

If a dsname allocation request is not valid with respect to the user's existing allocation environment, the request is failed. The following is a list of environmental conflicts that cause a request to be failed:

- The specified ddname is associated with an existing allocation that is in use.
- The specified ddname is associated with one of a group of concatenated data sets that the user defined as permanently concatenated. (For a definition of permanently concatenated, see "The Permanently Concatenated Attribute."
- The specified ddname is associated with an existing allocation that does not have the convertible attribute or that does not fulfil the conditions listed under "Using an Existing Allocation."
- The request specifies a new non-temporary data set with the same data set name as an existing allocation, unless a different volume serial number(s) is specified.
- A status of OLD or SHR is specified for a dsname that is associated with an existing allocation that is not permanently allocated, not in-use, and has a disposition of DELETE, unless specified volume serial numbers are different from those associated with the existing allocation.

Using an Existing Allocation

When successive processes require the same resource, the overhead of releasing and reobtaining the resource can be avoided. For example, time-sharing command processors often use the same data sets. Therefore, the data sets are not dynamically released at the end of the command process, but are designated "not-in-use," thus avoiding unallocation and reallocation processing.

An existing allocation can be used to satisfy only a request for the allocation of an explicitly specified dsname, a request for the allocation of the user's terminal as an I/O device, or a request for the allocation of a DUMMY data set. In addition, if any of the following are specified, the request is not eligible to be satisfied by an existing allocation: data set sequence number, label type, unit description (unless the dsname is an & dsname, in which case the unit description is ignored), unit count or parallel mounting, volume sequence number, volume count, volume reference, private volume, or DCB reference. If MSVGP is specified, it will be ignored if an existing allocation is used to satisfy the request.

An existing allocation of the specified dsname, terminal, or DUMMY allocation must have the following properties to satisfy an eligible request:

- It must not be in use.
- It must not be a member of a concatenated group.

- It must have the same volume serial number as any that are explicitly specified in the request.
- It must be permanently allocated if it has a disposition of DELETE and the request specified a status of MOD.
- It must not be a generation data group data set.
- It must have the convertible attribute or, if not, all of the following must be true (only the first requirement must be true for requests specifying an & dsname):
  - The request does not specify a ddname or the specified ddname matches the ddname associated with the existing allocation. A terminal request that does not specify a ddname cannot be satisfied by an existing allocation that does not have the convertible attribute.
  - For partitioned data sets, the member name specified in the request is the same as the member name associated with the existing allocation or a member name is not specified in the request and no member name is associated with the existing allocation.
  - DCB parameters, input only, or output only are not specified in the request.
  - A status of MOD is either specified in the request and associated with the existing allocation, or is not specified and not associated with the existing allocation.
  - The request does not specify that the convertible attribute be assigned to the allocation.

  The request does not specify that only convertible existing allocations may be used to satisfy the request.

If more than one existing allocation can satisfy the request, dynamic allocation selects: The existing allocation that is associated with the specified ddname. If no ddname was specified, dynamic allocation selects the existing allocation for which the in-use attribute has been most recently removed. (Data sets allocated via JCL are considered to have had their in-use attribute removed at step allocation.)

If the specified ddname is associated with an existing allocation that was not selected to satisfy the request, a ddname that is unique within the step is generated and associated with that existing allocation. This ddname consists of the characters 'SYS' followed by five digits. The association of a system-generated ddname with an existing allocation cannot occur in the following cases:

- The existing allocation is in use.
- The existing allocation is open.
- The existing allocation does not have the convertible attribute.
- The existing allocation is associated with a permanently concatenated group that does not represent an entire generation data set group or a multi-device type VSAM data set.

Changing the Parameters of an Existing Allocation When dynamic allocation uses an existing allocation to satisfy a dsname allocation request, some of the parameters of the existing allocation might have to be changed to reflect the parameters specified in the request. The only existing allocations that can have parameters changed are those allocated dynamically with the convertible attribute. Resources allocated via JCL or the TSO ALLOCATE command cannot have their parameters changed (with the exception of status and disposition specified via JCL), but they may be used if no changes are necessary.

Notes:

- Allocation requests that do not specify the permanently allocated attribute are automatically assigned the convertible attribute.
- Allocation requests may specify both the permanently allocated and the convertible attributes in their parameter list.

The following parameters are eligible for change:

- Ddname
- Membername
- Status
- Normal disposition
- Conditional disposition
- Space
- Unallocation at CLOSE
- Input only
- Output only
- DCB attributes
- Password
- Permanently allocated attribute

No other parameters are eligible to be changed.

Satisfying New Allocations

A new allocation is attempted when existing allocations cannot be used to satisfy a request. New allocations are not processed by the dynamic allocation routines while a job step holds (for possible reuse) more dynamically allocated resources than permitted. The number of allocated resources permitted is determined in two ways:

- The maximum number of allocation requests allowed per job step or time-sharing terminal session is 1635. This includes both dynamic requests and requests made via JCL.
- The control limit set by the DYNAMNBR parameter on the EXEC statement plus the number of DD statements limit the number of resources that can be held for reuse (that is, resources that are allocated but whose in-use bits are off).

When the maximum number of resources have been allocated and the user requests additional allocations, the dynamic allocation routines automatically attempt to unallocate enough resources to meet the control value limit.

Automatic Unallocation of Resources Held for Re-use: The only resources that are eligible for automatic unallocation are those that were allocated dynamically without the permanently allocated attribute and whose in-use bit has been turned off. (Resources allocated through JCL and through the time-sharing ALLOCATE command are not eligible because they automatically have the permanently allocated attribute.)

When many resources are eligible for automatic unallocation, the dynamic alllocation routines choose those that have been designated as not-in-use for the longest time. These are unallocated and the new allocation is processed.

If the control value is still exceeded after all eligible resources have been unallocated, the request for a new allocation fails.

In this case the user must explicitly request unallocation of an existing allocation before the new allocation can be performed.

Text Units by Function

Following are topics for each of the SVC 99 functions; each topic describes the key that you can specify for that function. The topics are ordered according to the order of the functions' verb codes, as follows:

- Dsname allocation (verb code X'01')
- Unallocation (verb code X'02')
- Concatenation (verb code X'03')
- Deconcatenation (verb code X'04')
- Remove-in-use processing based on task-id (verb code X'05')
- Ddname allocation (verb code X'06')
- Information retrieval (verb code X'07')

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dsname Allocation Function
0001	DALDDNAM	Associates a ddname with an allocation request.
0002	DALDSNAM	Names the data set to be allocated.
0003	DALMEMBR	Allocates only a particular data set member.
0004	DALSTATS	Specifies the data set status.
0005	DALNDISP	Specifies the data set's normal disposition.
0006	DALCDISP	Specifies the data set's conditional disposition.
0007	DALTRK	Specifies the space allocation in tracks.
0008	DALCYL	Specifies the space allocation in cylinders.
0009	DALBLKLN	Specifies the average data block length.
000A	DALPRIME	Specifies a primary space quantity.
000B	DALSECND	Specifies a secondary space quantity.
000C	DALDIR	Specifies the number of PDS directory blocks.
000D	DALRLSE	Deletes unused space at data set closure.
000E	DALSPFRM	Ensures a specific allocated space format.
000F	DALROUND	Specifies space allocation in whole cylinders.
0010	DALVLSER	Specifies volume serial numbers.
0011	DALPRIVT	Specifies the private volume use attribute.
0012	DALVLSEQ	Specifies the volume sequence number processing.
0013	DALVLCNT	Specifies the data set's volume count.
0014	DALVLRDS	Specifies volume reference to a cataloged data set.
0015	DALUNIT	Describes the unit specification.
0016	DALUNCNT	Specifies the number of devices to be allocated.
0017	DALPARAL	Specifies parallel mounting for a data set's volumes.
0018	DALSYSOU	Specifies the SYSOUT data set and defines its class.
0019	DALSPGNM	Specifies the SYSOUT program name.
001A	DALSFMNO	Specifies the SYSOUT form number.
001B	DALOUTLM	Limits the SYSOUT data set's logical record count.
001C	DALCLOSE	Frees a data set at closure.
001D	DALCOPYS	Specifies the SYSOUT listing copies count.
001E	DALLABEL	Specifies the type of volume label.
001F	DALDSSEQ	Specifies a tape data set's relative position.
0020	DALPASPR	Password protects the created data set.
0021	DALINOUT	Specifies "input only" or "output only" data set processing.
0022	DALEXPDT	Specifies the data set's expiration date.
0023	DALPRETPD	Specifies the data set's retention period.
0024	DALDUMMY	Allocates a dummy data set.
0025	DALFCBIM	Identifies the forms control buffer image.
0026	DALFCBAV	Requests operator verification of the image display or forms alignment.
0027	DALQNAME	Names a TPROCESS macro.
0028	DALTERM	Specifies a time sharing terminal as an I/O device.
0029	DALUCS	Specifies a universal character set.
002A	DALUFOLD	Specifies "fold mode" for loading the requested print chain or train.
002B	DALUVRFY	Requests operator verification of the correct print chain or train mounting.
002C	DALDCBDS	Specifies the retrieval of DCB information from a cataloged data set's label.
002D	DALDCBDD	Specifies the retrieval of DCB information from a ddname-related, currently allocated data set.
0058	DALSUSER	Specifies remote workstation routing for the SYSOUT data set.
0059	DALSHOLD	Specifies hold queue routing for the SYSOUT data set.
005E	DALMSVGP	Specifies a group of MSS virtual volumes.
005F	DALSSNM	Requests allocation of a subsystem data set.
0060	DALSSPRM	Specifies subsystem defined parameters for use with key DALSSNM.
0061	DALPROT	Requests that the direct access data set or the tape volume be RACF-protected.

Figure 8. Dsname Allocation (Verb Code 01) - Text Unit Keys, Mnemonics, and Functions

Dsname Allocation Text Units

Most of the information that can be specified on a JCL DD card may be specified in text units for the dsname allocation function (VERB code X'01'). These text units are described below and listed in Figure 8. The text units that represent DCB attributes are described under "DCB Attribute Text Units" and listed in Figure 9. The meaning of the parameters is the same as when specified on a DD statement as described in *OS/VS2 MVS JCL*. For dsname allocation text units that do not have a JCL equivalent, see Figure 10 and the topic "Non-JCL Dsname Allocation Functions."

Ddname specification - Key = X'0001'

DALDDNAM specifies a ddname to be associated with an allocation request. When this key is specified, # must be one, LEN is the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname DD1, code

```
KEY     #       LEN     PARM
0001    0001    0003    C4 C4 F1
```

Dsname specification - Key = X'0002'

DALDSNAM specifies the name of the data set to be allocated. The user cannot refer to a previously defined dsname. The QNAME and IPLTXTID keys are mutually exclusive with this key. When this key is specified, # must be one, LEN is the length of the dsname field, and PARM contains the dsname.

Example: to specify the dsname MYDATA, code

```
KEY     #       LEN     PARM
0002    0001    0006    D4 E8 C4 C1 E3 C1
```

Example: to specify the temporary dsname &LOAD, code

```
KEY     #       LEN     PARM
0002    0001    0005    50 D3 D6 C1 C4
```

Example: to specify the dsname A.B, code

```
KEY     #       LEN     PARM
0002    0001    0003    C1 4B C2
```

Member name specification - Key = X'0003'

DALMEMBR specifies that a particular member of a data set is to be allocated, rather than the entire data set. A relative generation group number may be specified as the member name. The dsname verb code (X'01) should be specified in confunction with this key. The QNAME and IPLTXTID keys are mutually exclusive with this key. When this key is specified, # must be one, LEN is the actual length of the member name, and PARM contains the member name.

Example: to specify the membername MEM1, code

```
KEY     #       LEN     PARM
0003    0001    0004    D4 C5 D4 F1
```

Example: to specify the relative generation number +1, code

```
KEY     #       LEN     PARM
0003    0001    0002    4E F1
```

Data Set Status specification - Key = X'0004'

DALSTATS specifies the data set status. It is mutually exclusive with the SYSOUT key. When this key is specified, # and LEN must be one, and PARM contains the value:

X'01' if OLD is desired
X'02' if MOD is desired
X'04' if NEW is desired
X'08' if SHR is desired

Example: to specify a status of NEW, code

```
Key     #       LEN     PARM
0004    0001    0001    04
```

Data Set Normal Disposition specification - Key = X'0005'

DALNDISP specifies the data set normal disposition. This key is mutually exclusive with a SYSOUT key. When this key is specified, # and LEN must be one, and PARM contains the value:

X'01' if UNCATLG is desired
X'02' if CATLG is desired
X'04' if DELETE is desired
X'08' if KEEP is desired

Example: to specify a normal disposition of DELETE, code

```
KEY     #       LEN     PARM
0005    0001    0001    04
```

Data Set Conditional Disposition specification - Key = X'0006'

DALCDISP specifies the conditional data set disposition. The values for #, LEN, and PARM are the same as for normal disposition. This key is mutually exclusive with the SYSOUT key.

Example: to specify a conditional disposition of DELETE, code

```
KEY     #       LEN     PARM
0006    0001    0001    04
```

Track Space Type (TRK) specification - Key = X'0007'

DALTRK specifies that space is to be allocated in tracks. The primary quantity space key or the secondary quantity space key (see below) must also be specified when this key is specified. The cylinder and block space type keys are mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Track Space Type, code

```
KEY     #       LEN     PARM
0007    0000    -       -
```

Cylinder Space Type (CYL) specification - Key = X'0008'

DALCYL specifies that space is to be allocated in cylinders. The primary quantity space key or secondary quantity space key must also be specified when this key is specified. The track and block space type keys are mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify Cylinder Space Type, code

```
KEY     #       LEN     PARM
0008    0000    -       -
```

Block Space Type specification - Key = X'0009'

DALBLKLN specifies the average data block length to be used by the system in computing the amount of space to allocate. The primary quantity space key or the secondary quantity space key must also be specified when this is specified. The track and cylinder space type keys are mutually exclusive with this key. When this key is specified, # must be one, LEN must be three, and PARM contains the average data block length. The maximum PARM value is X'FFFF' (65,535).

Example: to specify an average data block length of 80, code

```
KEY     #       LEN     PARM
0009    0001    0003    00 00 50
```

Primary Space Quantity specification - Key = X'000A'

DALPRIME specifies a primary space quantity. A space type key must also be specified when this key is specified. When this key is specified, # must be one, LEN must be three, and PARM contains the primary quantity value.

Example: to specify a primary quantity of 20, code

```
KEY     #       LEN     PARM
000A    0001    0003    00 00 14
```

Secondary Space Quantity specification - Key = X'000B'

DALSECND specifies a secondary space quantity. When this key is specified, # must be one, LEN must be three, and PARM contains the secondary quantity value.

Example: to specify a secondary space quantity of 10, code

```
KEY     #       LEN     PARM
000B    0001    0003    00 00 0A
```

Directory Block specification - Key = X'000C'

DALDIR specifies the number of blocks to be contained in the directory of a partitioned data set. A space type key and the primary space quantity key must also be specified when this key is specified. When this key is specified, # must be one, LEN must be three, and PARM contains the number of directory blocks.

Example: to specify two directory blocks, code

```
KEY     #       LEN     PARM
000C    0001    0003    00 00 02
```

Unused Space Release (RLSE) specification - Key = X'000D'

DALRLSE specifies that unused space is to be deleted when the data set is closed. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the release of unused space, code

```
KEY     #       LEN     PARM
000D    0000    -       -
```

Format of Allocated Space specification - Key = X'000E'

DALSPFRM specifies a particular format of allocated space. When this key is specified, # and LEN must be one, and PARM contains:

X'02' if different areas of contiguous space are to be allocated (ALX)
X'04' if maximum contiguous space is required (MXIG)
X'08' if space must be contiguous (CONTIG)

Example: to specify contiguous space format, code

```
KEY     #       LEN     PARM
000E    0001    0001    08
```

Whole Cylinder Allocation (ROUND) specification - Key = X'000F'

DALROUND specifies that allocated space be equal to one or more whole cylinders when space is requested in units of blocks. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify allocation of whole cylinders, code

```
KEY     #       LEN     PARM
000F    0000    -       -
```

Volume Serial specification - Key = X'0010'

DALVLSER specifies volume serial numbers. It is mutually exclusive with the SYSOUT key and volume reference key (see below). When this key is specified, # contains the number of volume serials being specified, LEN contains the length of the immediately following volume serial, and PARM contains the volume serial.

Example: to specify the volume serials 231400 and 231401, code

```
KEY     #       LEN     PARM                    LEN     PARM
0010    0002    0006    F2 F3 F1 F4 F0 F0       00 06   F2 F3 F1 F4 F0 F1
```

Private Volume specification - Key = X'0011'

DALPRIVT specifies that the volume(s) allocated be assigned the PRIVATE volume use attribute. This key is mutually exclusive with a SYSOUT key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the PRIVATE volume attribute, code

```
KEY     #       LEN     PARM
0011    0000    -       -
```

Volume Sequence Number specification - Key = X'0012'

DALVLSEQ specifies which volume, of a multi-volume data set, processing is to begin with. This key is mutually exclusive with the SYSOUT key. When this key is specified, # must be one, LEN must be two, and PARM contains the volume sequence number. The maximum PARM value is X'FF' (255).

Example: to specify a volume sequence number of two, code

```
KEY     #       LEN     PARM
0012    0001    0002    0002
```

Volume Count specification - Key = X'0013'

DALVLCNT specifies the maximum number of volumes an output data set may require. This key is mutually exclusive with the SYSOUT key. When this key is specified, # and LEN must be one, and PARM contains the volume count.

Example: to specify a volume count of 10, code

```
KEY     #       LEN     PARM
0013    0001    0001    0A
```

Volume Reference to a Dsname specification - Key = X'0014'

DALVLRDS specifies that the system is to obtain volume serial information from the specified cataloged data set. This key is mutually exclusive with the SYSOUT key and volume serial key. (Volume reference to a ddname can not be done through dynamic allocation.) When this key is specified, # must be one, LEN is the actual length of the dsname, and PARM contains the non-blank dsname.

Example: to specify volume reference to the data set DSN1, code

```
KEY     #       LEN     PARM
0014    0001    0004    C4 E2 D5 F1
```

Unit Description specification - Key = X'0015'

DALUNIT specifies a unit group (esoteric) name, device type, or specific unit address (in EBCDIC). When this key is specified, # must be one, LEN is the actual length of the unit description, and PARM contains the unit description.

Example: to specify the unit group name SYSDA, code

```
KEY     #       LEN     PARM
0015    0001    0005    E2 E8 E2 C4 C1
```

Example: to specify the device type 3330, code

```
KEY     #       LEN     PARM
0015    0001    0004    F3 F3 F3 F0
```

Example: to specify the unit address 230, code

```
KEY     #       LEN     PARM
0015    0001    0003    F2 F3 F0
```

Unit Count specification - Key = X'0016'

DALUNCNT specifies the number of devices to be allocated. It is mutually exclusive with the parallel mount key (see below). When this key is specified, # and LEN must be one, and PARM contains the unit count. The maximum PARM value is X'3B' (59).

Example: to specify a unit count of ten, code

```
KEY     #       LEN     PARM
0016    0001    0001    0A
```

Parallel Mount specification - Key = X'0017'

DALPARAL specifies that each volume of a data set is to be assigned a device. It is mutually exclusive with the unit count key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify parallel mount, code

```
KEY     #       LEN     PARM
0017    0000    -       -
```

SYSOUT specification - Key = X'0018'

DALSYSOU specifies that a system output data set is to be allocated and defines the output class of the data set. When this key is specified and a class other than the default of the message class is desired, # and LEN must be one, and PARM contains the output class. To obtain the default of the message class, # must be zero. LEN and PARM are not specified. Volume, QNAME, status, disposition, subsystem name request and subsystem parameter key are mutually exclusive with the SYSOUT key.

Example: to specify a SYSOUT data set in class A, code

```
KEY     #       LEN     PARM
0018    0001    0001    C1
```

Example: to specify a SYSOUT data set and to default the class, code

```
KEY     #       LEN     PARM
0018    0000    -       -
```

SYSOUT Program Name specification - Key = X'0019'

DALSPGNM specifies the SYSOUT program name. The SYSOUT key must also be specified when this key is specified. When this key is specified, # must be one, LEN is the actual length of the name, and PARM contains the program name. The subsystem name request and subsystem parameter keys are mutually exclusive with the SYSOUT program name key.

Example: to specify the program name MYWRITER, code

```
KEY     #       LEN     PARM
0019    0001    0008    D4 E8 E6 D9 C9 E3 C5 D9
```

SYSOUT Form Number specification - Key = X'001A'

DALSFMNO specifies the SYSOUT form number. The SYSOUT key must also be specified when this key is specified. When this key is specified, # must be one, LEN is the actual length of the form number, and PARM contains the form number. The subsystem name request and subsystem parameter keys are mutually exclusive with the SYSOUT form number key.

Example: to specify the form number 1234, code

```
KEY     #       LEN     PARM
001A    0001    0004    F1 F2 F3 F4
```

SYSOUT Output Limit specification - Key = X'001B'

DALOUTLM specifies the number of logical records in a SYSOUT data set. The SYSOUT key must also be specified when this key is specified. When this key is specified, # must be one, LEN must be three, and PARM contains the output limit.

Example: to specify an Output Limit of 1000, code

```
KEY     #       LEN     PARM
001B    0001    0003    00 03 E8
```

Unallocation at CLOSE specification - Key X'001C'

DALCLOSE specifies that unallocation is to occur when a DCB is closed rather than at step unallocation. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify unallocation at CLOSE, code

```
KEY     #       LEN     PARM
001C    0000    -       -
```

SYSOUT Copies specification - Key = X'001D'

DALCOPYS specifies up to 255 hardcopy listings of a particular SYSOUT data set. The SYSOUT key must also be specified when this key is specified. When this key is specified, # and LEN must be one, and PARM contains the number of copies being requested.

Example: to specify a request for 25 copies, code

```
KEY     #       LEN     PARM
001D    0001    0001    19
```

Label Type specification - Key = X'001E'

DALLABEL specifies the type of label associated with a volume. This key is mutually exclusive with the SYSOUT key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' if the volume has no label (NL)
X'02' if the volume has an IBM standard label (SL)
X'04' if the volume has a non-standard label (NSL)
X'0A' if the volume has both an IBM standard label and a user label (SUL)
X'10' if label processing is to be bypassed (BLP)
X'21' if the system is to check for and bypass a leading tape mark on DOS unlabeled tape (LTM)
X'40' if the volume has an American National Standard label (AL)
X'48' if the volume has an American National Standard label and an American National Standard user label (AUL)

Example: to specify no labels, code

```
KEY     #       LEN     PARM
001E    0001    0001    01
```

Data Set Sequence Number specification - Key = X'001F'

DALDSSEQ specifies the relative position of a data set on a tape volume (data set sequence number). This key is mutually exclusive with the SYSOUT key. When this key is specified, # must be one, LEN must be two, and PARM contains the sequence number. The maximum PARM value is X'270F' (9999).

Example: to specify a data set sequence number of 2, code

```
KEY     #       LEN     PARM
001F    0001    0002    00 02
```

Password Protection specification - Key = X'0020'

DALPASPR specifies that the data set being created is to be password protected. This key is mutually exclusive with the SYSOUT key. When this key is specified, # and LEN must be one, and PARM contains:

X'10' if the data set should not be read, changed, extended, or deleted without the password.
X'30' if the data set should not be changed, extended, or deleted without the password. Reading is permitted.

Example: to specify complete password protection, code

```
KEY     #       LEN     PARM
0020    0001    0001    10
```

Input Only or Output Only specifications - Key = X'0021'

DALINOUT specifies that the data set is to be processed for input only or output only. This key is mutually exclusive with the SYSOUT key. When this key is specified, # and LEN must be one, and PARM contains:

X'40' if output only is to be requested.
X'80' if input only is to be requested.

Example: to specify processing for input only, code

```
KEY     #       LEN     PARM
0021    0001    0001    80
```

Expiration Date specification - Key = X'0022'

DALEXPDT specifies the date when the data set can be deleted or overwritten by another data set. This key is mutually exclusive with the retention period and SYSOUT key. When this key is specified, # must be one, LEN must be five, and PARM contains five digits, a two-digit year number and a three-digit day number (YYDDD).

Example: to specify an expiration date of January 1, 1976 (76001), code

```
KEY     #       LEN     PARM
0022    0001    0005    F7 F6 F0 F0 F1
```

Retention Period specification - Key = X'0023'

DALPRETPD specifies the number of days that must pass before the data set can be deleted or overwritten by another data set. This key is mutually exclusive with the expiration date and SYSOUT keys. When this key is specified, # must be one, LEN must be two, and PARM contains the retention period. The maximum PARM value is X'270F' (9999).

Example: to specify a retention period of 10 days, code

```
KEY     #       LEN     PARM
0023    0001    0002    000A
```

DUMMY Data Set specification - Key = X'0024'

DALDUMMY specifies that a DUMMY data set is to be allocated. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify a DUMMY data set is to be allocated, code

```
KEY     #       LEN     PARM
0024    0000    -       -
```

Forms Control Buffer (FCB) Image Identification specification - Key = X'0025'

DALFCBIM specifies the code that identifies the image to be loaded into the FCB. This keys is mutually exclusive with the DCB INTVL and FRID keys (see below). When this key is specified, # must be one, LEN contains the length of the image-id (maximum of 4), and PARM contains the image-id.

Example: to specify the image-id STD1, code

```
KEY     #       LEN     PARM
0025    0001    0004    E2 E3 C4 F1
```

Form Alignment and Image Verification specification - Key = X'0026'

DALFCBAV specifies that the operator check the alignment of the printer forms before the data set is printed or that he visually verify the image displayed on the printer as the desired one. The FCB image-id key must also be coded when this key is specified. When this key is specified, # and LEN must be one, and PARM contains:

X'04' if verification is requested (VERIFY).
X'08' if alignment is requested (ALIGN).

Example: to specify verification, code

```
KEY     #       LEN     PARM
0026    0001    0001    04
```

QNAME specification - Key = X'0027'

DALQNAME specifies the name of a TPROCESS macro. The dsname, member name, IPLTXTID, and SYSOUT keys are mutually exclusive with this key. The DCB BLKSIZE, BUFL, LRECL, OPTCD and RECFM keys (see "DCB Attribute Text Units") are meaningful with this key. When this key is specified, # must be one, LEN is the length of the process name, and PARM contains the process name.

Example: to specify the process name TP1, code

```
KEY     #       LEN     PARM
0027    0001    0003    E3 D7 F1
```

Terminal specification - Key = X'0028'

DALTERM is used to specify that a time-sharing terminal is to be used as an I/O device. In a batch environment, the specification is not used, but is checked for syntax. In a time-sharing environment, all other specifications except DCB specifications are ignored. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify a terminal allocation, code

```
KEY     #       LEN     PARM
0028    0000    -       -
```

Universal Character Set (UCS) specification - Key = X'0029'

DALUCS identifies a special character set to be used for printing a data set. The DCB INTVL and RESERVE keys (see "DCB Attribute Text Units") are mutually exclusive with this key. When this key is specified, # must be one, LEN is the length of the character set code name (maximum is four) and PARM contains the character set code.

Example: to specify the character set code AN, code

```
KEY     #       LEN     PARM
0029    0001    0002    C1 D5
```

Fold Mode specification - Key = X'002A'

DALUFOLD specifies that the chain or train corresponding to the desired character set be loaded in the fold mode. The universal character set key must also be specified when this key is coded. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify fold mode, code

```
KEY     #       LEN     PARM
002A    0000    -       -
```

Character Set Image Verification specification - Key = X'002B'

DALUVRFY specifies that the operator is to verify that the correct chain or train is mounted before the data set is printed. The universal character set key must also be specified when this key is coded. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify character set image verification, code

```
KEY     #       LEN     PARM
002B    0000    -       -
```

DCB Reference to a Dsname specification - Key = X'002C'

DALDCBDS specifies that DCB information is to be retrieved from the data set label of a cataloged data set. This data set must reside on a direct access volume and the volume must currently be mounted. The DSORG, RECFM, OPTCD, BLKSIZE, LRECL, RKP, and KEYLEN DCB attributes, and the volume sequence number and expiration date are copied from the data set label. If text units for these parameters are specified in addition to this key, that specification overrides the corresponding parameter that was copied. This key is mutually exclusive with a DCB reference to a ddname (see below). When this key is specified, # must be one, LEN is the length of the dsname, and PARM contains the non-blank dsname.

Example: to specify DCB reference to the dsname ABC, code

```
KEY     #       LEN     PARM
002C    0001    0003    C1 C2 C3
```

DCB Reference to a Ddname specification - Key = X'002D'

DALDCBDD specifies that DCB information is to be retrieved from the currently allocated data set associated with the specified ddname. For time sharing users, the expiration date and INPUT/OUTPUT ONLY specifications are also retrieved. This key is mutually exclusive with a DCB reference to a dsname specification. Any DCB attributes, expiration date, and INPUT/OUTPUT ONLY keys specified in addition to this key override the corresponding keys associated with the ddname. When this key is specified, # must be one, LEN is the length of the ddname, and PARM contains the ddname.

Example: to specify DCB reference to the ddname DD1, code

```
KEY     #       LEN     PARM
002D    0001    0003    C4 C4 F1
```

SYSOUT Remote Workstation specification - Key = X'0058'

DALSUSER specifies that the SYSOUT data set being allocated is to be routed to a remote workstation when it is unallocated. The SYSOUT key must also be specified when this key is specified. When this key is coded, # must be one, LEN is the length of the remote workstation name (maximum of 8), and PARM contains the remote user name.

Example: to specify the remote work station USER01, code

```
KEY     #       LEN     PARM
0058    0001    0006    E4 E2 C5 D9 F0 F1
```

SYSOUT Hold Queue specification - Key = X'0059'

DALSHOLD specifies that the SYSOUT data set being allocated is to be placed on the Hold Queue when it is unallocated. The SYSOUT key must also be specified when this key is specified. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify hold, code

```
KEY     #       LEN     PARM
0059    0000    -       -
```

MSVGP specification - Key = X'005E'

DALMSVGP specifies a group of MSS virtual volumes. This key is mutually exclusive with SYSOUT, QNAME, and volume serial keys. When this key is specified, # must be one, LEN is the actual length of the MSVGP name, and PARM contains the group name.

Example: to specify a MSS volume group of SYSGROUP, code

```
KEY     #       LEN     PARM
005E    0001    0008    E2 E8 E2 C7 D9 D6 E4 D7
```

Subsystem Name Request specification - Key = X'005F'

DALSSNM specifies a subsystem data set. You must specify the name of the subsystem that is to process the request for allocation unless you want the request processed by the default subsystem.

- When you request a subsystem other than the default subsystem, # must be one, LEN specifies the length of the subsystem name and must contain a number ranging in value from 1 to 4, and PARM must contain a 1- to -4 character subsystem name.

  The first character of the subsystem name must be either alphabetic or national and the remaining characters must be either alphameric or national. See *OS/VS MVS JCL* for a list of the alphameric and national character sets.
- When you request the default subsystem, # must be zero, and LEN and PARM must not be specified.

This key is mutually exclusive with the SYSOUT, SYSOUT program name, and SYSOUT form number keys.

The system programming staff at your installation can identify the subsystems at your installation that support DALSSMN requests.

Example 1: to request subsystem SUB1, code

```
KEY     #       LEN     PARM
005F    0001    0004    E2 E4 C2 F1
```

Example 2: to request the default subsystem, code

```
KEY     #       LEN     PARM
0005F   0000    -   -
```

Subsystem Parameter specification - Key = X'0060'

DALSSPRM specifies parameters that will be processed by a subsystem. When you specify DALSSPRM, you must also specify the subsystem name request keys. When you specify this key, # represents the number of LEN and PARM combinations that are present and must contain a number ranging in value from 1 to 254. LEN specifies the length of the parameter that follows it and must contain a number ranging in value from 0 to 67. When LEN is zero, do not specify PARM.

This key is mutually exclusive with the SYSOUT, SYSOUT program name, and SYSOUT form number keys.

Example, to specify two parameters, PARM1 and PARM2 code

```
KEY     #       LEN     PARM
0060    0002    0005    D7 C1 D9 D4 F1
                000A    D7 C1 D9 C1 D4 C5 E3 C5 D9 F2
```

Note: For additional information about subsystem data sets and subsystem parameters, refer to the documentation for the particular subsystem.

PROTECT specification - Key = X'0061'

DALPROT specifies that the direct access data set or the tape volume be RACF-protected when the direct access data set is defined or the tape volume used. This key is mutually exclusive with the SYSOUT, FCB, QNAME, TERM, and UCS keys. When this key is specified, # must be zero, and LEN and PARM must not be specified. See *OS/VS2 MVS JCL* for additional information about specifying the PROTECT function.

Example: to specify PROTECT, code

```
KEY     #       LEN     PARM
0061    0000    -       -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
002E	DALBFALN	Specifies buffer alignment.
002F	DALBFTEK	Specifies the buffering technique.
0030	DALBLKSZ	Specifies blocksize.
0031	DALBUFIN	Specifies the receiving buffer count.
0032	DALBUFL	Specifies the buffer length.
0033	DALBUFMX	Specifies the buffer count per line.
0034	DALBUFNO	Specifies the buffer count per DCB.
0035	DALBUFOF	Specifies the buffer offset.
0036	DALBUFOU	Specifies the sending buffer count.
0037	DALBUFRQ	Specifies the buffer count per GET macro instruction.
0038	DALBUFSZ	Specifies the line group buffer size.
0039	DALCODE	Specifies the data's paper tape code.
003A	DALCPRI	Specifies the relative sending and receiving priority.
003B	DALDEN	Specifies the magnetic tape density.
003C	DALDSORG	Specifies the data set organization.
003D	DALEROPT	Specifies reading and writing error options.
003E	DALGNCP	Specifies the GAM-I/O count per WAIT macro instruction.
003F	DALINTVL	Specifies the line polling interval per group.
0040	DALKYLEN	Specifies the data set key lengths.
0041	DALLIMCT	Specifies the search limit.
0042	DALLRECL	Specifies the logical record length.
0043	DALMODE	Specifies card punch/reader operational mode.
0044	DALNCP	Specifies the READ/WRITE count per CHECK.
0045	DALOPTCD	Specifies the control program's operational services.
0046	DALPCIR	Specifies the relationship of the receiving PCI to the allocation and freeing of buffers.
0047	DALPCIS	Specifies the relationship of the sending PCI to the allocation and freeing of buffers.
0048	DALPRTSP	Specifies printer line spacing.
0049	DALRECFM	Specifies the record format.
004A	DALRSRVF	Specifies the first buffer's reserve byte count for insertion of data.
004B	DALRSRVS	Specifies the secondary buffer's reserve byte count for insertion of data.
004C	DALSOWA	Specifies the user's telecommunications input work areas size.
004D	DALSTACK	Specifies the card punch's stacker bin.
004E	DALTHRSH	Specifies the use percentage of nonreusable direct access message queue records per flush closedown.
004F	DALTRTCH	Specifies the 7-track tape recording technique.
0051	DALIPLTX	Specifies a TCAM network control program name.
0054	DALDIAGN	Requests OPEN/CLOSE/EOV diagnostic trace option.
005A	DALFUNC	Specifies the type of data set to be opened for the 3525 Card-Read-Punch-Print.
005B	DALFRID	Specifies a SYS1.IMAGELIB member for 3886 input.

Figure 9. DCB Attributes (Used with Verb Code 01) – Text Unit Keys, Mnemonics, and Functions

DCB Attribute Text Units

The following keys are used to specify DCB attributes. These attributes are described in *OS/VS2 MVS JCL* under the DCB keyword subparameter specification and in *OS/VS2 MVS Data Management Macro Instructions*.

BFALN specification - Key = X'002E'

DALBFALN specifies buffer alignment. The GNCP specification is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for fullword not a doubleword boundary (F)
X'02' for doubleword boundary (D)

Example: to specify doubleword boundary, code

```
KEY     #       LEN     PARM
002E    0001    0001    02
```

BFTEK specification - Key = X'002F'

DALBFTEK specifies the buffering technique. The GNCP key is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'08' for dynamic buffering (D)
X'10' for exchange buffering (E)
X'20' for record buffering (R)
X'40' for simple buffering (S)
X'60' for record area buffering (A)

Example: to specify exchange buffering, code

```
KEY     #       LEN     PARM
002F    0001    0001    10
```

BLKSIZE specification - Key = X'0030'

DALBLKSZ specifies the block size. The BUFSIZE key is mutually exclusive with this key. When this key is specified, # must be one, LEN must be two, and PARM contains the block size. The maximum PARM value is X'7FF8' (32,760).

Example: to specify a block size of 80, code

```
KEY     #       LEN     PARM
0030    0001    0002    00 50
```

BUFIN specification - Key = X'0031'

DALBUFIN specifies the number of buffers to be initially assigned for receiving operations for each line in the line group. The BUFNO and BUFRQ keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers. The maximum PARM value is X'0F' (15).

Example: to specify 2 buffers, code

```
KEY     #       LEN     PARM
0031    0001    0001    02
```

BUFL specification - Key = X'0032'

DALBUFL specifies the buffer length. When this key is specified, # must be one, LEN must be two, and PARM contains the buffer length. The maximum PARM value is X'7FF8' (32,760).

Example: to specify a buffer length of 80, code

```
KEY     #       LEN     PARM
0032    0001    0002    00 50
```

BUFMAX specification - Key = X'0033'

DALBUFMX specifies the maximum number of buffers to be allocated to a line at one time. The NCP key is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers. The maximum PARM value is X'0F' (15).

Example: to specify 4 buffers, code

```
KEY     #       LEN     PARM
0033    0001    0001    04
```

BUFNO specification - Key = X'0034'

DALBUFNO specifies the number of buffers to be assigned to the data control block. The BUFIN, BUFOUT, and BUFRQ keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers.

Example: to specify 2 buffers, code

```
KEY     #       LEN     PARM
0034    0001    0001    02
```

BUFFOFF specification - Key = X'0035'

DALBUFOF specifies the buffer offset. When this key is specified, # and LEN must be one, and PARM contains one of the following:

X'80' the block prefix is four bytes long and contains the block length (L)
X'nn' the length of the block prefix (maximum of X'63') (99))

Example: to specify offset of 16, code

```
KEY     #       LEN     PARM
0035    0001    0001    10
```

BUFOUT specification - Key = X'0036'

DALBUFOU specifies the number of buffers to be assigned initially for sending operations for each line in the line group. The BUFNO and BUFRQ keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers. The maximum PARM value is X'0F' (15).

Example: to specify 4 buffers, code

```
KEY     #       LEN     PARM
0036    0001    0001    04
```

BUFRQ specification - Key = X'0037'

DALBUFRQ specifies the number of buffers to be requested in advance for the GET macro instruction. The BUFNO, BUFIN, and BUFOUT keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of buffers.

Example: to specify 4 buffers, code

```
KEY     #       LEN     PARM
0037    0001    0001    04
```

BUFSZ specification - Key = X'0038'

DALBUFSZ specifies the length in bytes of each of the buffers to be used for all lines in a particular line group. The BLKSIZE key is mutually exclusive with this key. When this key is specified, # must be one, LEN must be two, and PARM contains the buffer length.

Example: to specify a buffer length of 80, code

```
KEY      #        LEN      PARM
0038     0001     0002     00 50
```

CODE specification - Key = X'0039'

DALCODE specifies the paper tape code in which the data is punched. The KEYLEN, MODE, PRTSP, STACK, and TRTCH keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'02' for Teletype 5-track (T)
X'04' for USASCII 8-track (A)
X'08' for National Cash Register 8-track (C)
X'10' for Burroughs 7-track (B)
X'20' for Friden 8-track (F)
X'40' for IBM BCD 8-track (I)
X'80' for no conversion (N)

Example: to specify USASCII, code

```
KEY      #        LEN      PARM
0039     0001     0001     04
```

CPRI specification - Key = X'003A'

DALCPRI specifies the relative priority to be given to sending and receiving operations. The THRESH key is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for send priority (S)
X'02' for equal priority (E)
X'04' for receiving priority (R)

Example: to specify equal priority, code

```
KEY      #        LEN      PARM
003A     0001     0001     02
```

DEN specification - Key = X'003B'

DALDEN specifies the magnetic tape density. When this key is specified, # and LEN must be one, and PARM contains:

X'03' for 200 bpi 7 - track (0)
X'43' for 556 bpi 7 - track (1)
X'83' for 800 bpi 7 - track, 800 bpi 9 - track (2)
X'C3' for 1600 bpi 9 - track (3)
X'D3' for 6250 bpi 9 - track (4)

Example: to specify 1600 bpi 9 - track, code

```
KEY      ##       LEN      PARM
003B     0001     0001     C3
```

DSORG specifications - Key = X'003C'

DALDSORG specifies the data set organization. When this key is specified, # must be one, LEN must be two, and PARM contains:

X'0004' for TCAM 3705
X'0008' for VSAM
X'0020' for TCAM message queue (TQ)
X'0040' for TCAM line group (TX)
X'0080' for graphics (GS)

X'0200' for partitioned organization (PO)
X'0300' for partitioned organization unmovable (POU)
X'0400' for government of message transfer to or from a telecommunications message processing queue (MQ)
X'0800' for direct access message queue (CQ)
X'1000' for communication line group (CX)
X'2000' for direct access (DA)
X'2100' for direct access unmovable (DAU)
X'4000' for physical sequential (PS)
X'4100' for physical sequential unmovable (PSU)

Example: to specify Partitioned Organization, code

```
KEY     #       LEN     PARM
003C    0001    0002    02 00
```

EROPT specification - Key = X'003D'

DALEROPT specifies the option to be executed if an error occurs in writing or reading a record. When this key is specified, # and LEN must be one, and PARM contains:

X'10' for online BSAM testing (T)
X'20' to cause abnormal end of task (ABE)
X'40' to skip the block causing the error (SKP)
X'80' to accept the block causing the error (ACC)

Example: to specify the SKP error option, code

```
KEY     #       LEN     PARM
003D    0001    0001    40
```

GNCP specification - Key = X'003E'

DALGNCP specifies the maximum number of GAM input/output macro instructions that will be issued before a WAIT macro instruction is issued. This key is mutually exclusive with the BFTEK and BFALN keys. When this key is specified, # and LEN must be one, and PARM contains the GNCP value. The maximum PARM value is X'63' (99).

Example: to specify a GNCP value of four, code

```
KEY     #       LEN     PARM
003E    0001    0001    04
```

INTVL specification - Key = X'003F'

DALINTVL specifies the polling interval for the lines in the line group. This key is mutually exclusive with UCS and FCB keys. When this key is specified, # and LEN must be one, and PARM contains the INTVL value.

Example: to specify an INTVL value of 10, code

```
KEY     #       LEN     PARM
003F    0001    0001    0A
```

KEYLEN specification - Key = X'0040'

DALKYLEN specifies the length, in bytes, of the keys used in the data set. The CODE, MODE, PRTSP, STACK, and TRTCH keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the key length.

Example: to specify a key length of eight, code

```
KEY     #       LEN     PARM
0040    0001    0001    08
```

LIMCT specification - Key = X'0041'

DALLIMCT specifies the search limit. When this key is specified, # must be one, LEN must be three, and PARM contains the search limit value. The maximum PARM value is X'007FF8' (32,760).

Example: to specify a search limit of 1000, code

```
KEY      #        LEN      PARM
0041     0001     0003     0003E8
```

LRECL specification - Key = X'0042'

DALLRECL specifies the actual or maximum length, in bytes, of a logical record. When this key is specified, # must be one, LEN must be two, and PARM contains one of the following:

X'8000' for variable length spanned records processed under QSAM and BSAM, the logical records exceed 32,756 bytes (X)

X'7FF8' the logical record length. The maximum PARM value is X'7FF8' (32,760).

Example: to specify a logical record length of 80, code

```
KEY      #        LEN      PARM
0042     0001     0002     0050
```

MODE specification - Key = X'0043'

DALMODE specifies the mode of operation for a card reader or punch. This key is mutually exclusive with the CODE, KEYLEN, PRTSP, and TRTCH keys. When this key is specified, # and LEN must be one, and PARM contains:

X'40' for EBCDIC mode (E)
X'50' for EBCDIC, read column eliminate mode (ER)
X'60' for EBCDIC, optical mark read mode (EO)
X'80' for card image mode (C)
X'90' for card image, read column eliminate mode (CR)
X'A0' for card image, optical mark read mode (CO)

Example: to specify EBCDIC mode, code

```
KEY      #        LEN      PARM
0043     0001     0001     40
```

NCP specification - Key = X'0044'

DALNCP specifies the maximum number of READ or WRITE macro instructions issued before a CHECK macro instruction is issued. This key is mutually exclusive with the BUFMAX key. When this key is specified, # and LEN must be one, and PARM contains the NCP value. The maximum PARM value is X'63' (99).

Example: to specify an NCP value of two, code

```
KEY      #        LEN      PARM
0044     0001     0001     02
```

OPTCD specification - Key = X'0045'

DALOPTCD specifies optional services to be performed by the control program. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for relative block addressing (R)
X'02' for user totaling facility (T)
X'04' for reduced tape error recovery or direct access search direct (Z)

X'08' for direct addressing (A) or for translation of ASCII to or from EBCDIC (Q)
X'10' for feedback (F), or for hopper empty exit (H), or for online correction for Optical Readers (O)
X'20' for chained scheduling or TCAM segment identification (C), or for extended search (E)
X'40' for end-of-file recognition to be disregarded for tapes (B), or for allowance of data checks caused by an invalid character, or TCAM work unit is to be handled as a message (U)
X'80' for write validity check (W)

Note: When more than one OPTCD value is to be specified, PARM contains the sum of the values.

Examples: to specify OPTCD value U, code

```
KEY     #       LEN     PARM
0045    0001    0001    40
```

Example: to specify OPTCD values U and C, code

```
KEY     #       LEN     PARM
0045    0001    0001    60
```

Receiving PCI specification - Key = X'0046'

DALPCIR specifies the relationship of program-controlled interrupts (PCI) during receiving operations to the allocation and freeing of buffers. When this key is specified, # and LEN must be one, and PARM contains:

X'02' for a PCI and no new buffer allocated (R)
X'08' for no PCIs (N)
X'20' for a PCI and new buffer allocated (A)
X'80' for a PCI, new buffer allocated, and the first buffer remains allocated (X)

Example: to specify no PCI's during receiving operations, code

```
KEY     #       LEN     PARM
0046    0001    0001    08
```

Sending PCI specification - Key = X'0047'

DALPCIS specifies the relationship of PCI's during sending operations to the allocation and freeing of buffers. When this key is specified, # and LEN contain one, and PARM contains:

X'01' for a PCI and no new buffer allocated (R)
X'04' for no PCIs (N)
X'10' for a PCI and a new buffer allocated (A)
X'40' for a PCI, new buffer allocated, and first buffer remains allocated (X)

Example: to specify no PCI's during sending operations, code

```
KEY     #       LEN     PARM
0047    0001    0001    04
```

PRTSP specification - Key = X'0048'

DALPRTSP specifies printer line spacing. The CODE, KEYLEN, MODE, STACK, and TRTCH keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for no spacing (0)
X'09' for one-line spacing (1)
X'11' for two-line spacing (2)
X'19' for three-line spacing (3)

Example: to specify no spacing, code

```
KEY      #        LEN      PARM
0048     0001     0001     01
```

RECFM specification - Key = X'0049'

DALRECFM specifies the record format. When this key is specified, # and LEN must be one, and PARM contains:

X'02' for machine code printer control characters in record (M), or for complete QTAM record (R)
X'04' for ASA printer control characters in record (A), or for complete QTAM message (G)
X'08' for standard fixed records, spanned variable records, or segment of QTAM message (S)
X'10' for blocked records (B)
X'20' for variable ASCII records (D), or for track overflow (T)
X'40' for variable records (V)
X'80' for fixed records (F)
X'C0' for undefined records (U)

Notes: When more than one RECFM value is to be specified in combination, PARM contains the sum of the values.

Example: to specify fixed records, code

```
KEY      #        LEN      PARM
0049     0001     0001     80
```

First Buffer Reserve specification - Key = X'004A'

DALRSRVF specifies the number of bytes to be reserved in the first buffer for insertion of data by the DATETIME and SEQUENCE macros. The UCS key is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of bytes to reserve.

Example: to reserve 8 bytes in the first buffer, code

```
KEY      #        LEN      PARM
004A     0001     0001     08
```

Secondary Buffer Reserve specification - Key = X'004B'

DALRSRVS specifies the number of bytes to be reserved in other than the first buffer for insertion of data by the DATETIME and SEQUENCE macro instructions. The UCS key is mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains the number of bytes to reserve.

Example: to reserve 8 bytes in secondary buffers, code

```
KEY      #        LEN      PARM
004B     0001     0001     08
```

SOWA specification - Key = X'004C'

DALSOWA specifies the size, in bytes, of the user-provided input work areas for telecommunication jobs. When this key is specified, # must be one, LEN must be two, and PARM contains the number of bytes. The maximum PARM value is X'7FF8' (32,760).

Example: to specify a 256-byte work area, code

```
KEY      #        LEN      PARM
004C     0001     0002     0100
```

STACK specification - Key = '004D'

DALSTACK specifies the stacker bin to receive cards. The CODE, KEYLEN, PRTSP, and TRTCH keys are mutually exclusive with this key. When this key specified, # and LEN are one, and PARM contains:

X'01' for bin 1 (1)
X'02' for bin 2 (2)

Example: to specify stacker 2, code

```
KEY     #       LEN     PARM
004D    0001    0001    02
```

THRESH specification - Key = X'004E'

DALTHRSH specifies the percentage of nonreusable disk message queue records to be used before a flush closedown occurs. This key is mutually exclusive with the CPRI key. When this key is specified, # and LEN must be one, and PARM contains the percentage. The maximum PARM value is '64' (100).

Example: to specify a THRESH percentage of 99, code

```
KEY     #       LEN     PARM
004E    0001    0001    63
```

TRTCH specification - Key = X'004F'

DALTRTCH specifies the recording technique for 7-track tape. The KEYLEN, MODE, CODE, STACK, and PRTSP keys are mutually exclusive with this key. When this key is specified, # and LEN must be one, and PARM contains:

X'13' for data conversion (C)
X'23' for even parity (E)
X'2B' for even parity and BCD/EBCDIC translation (ET)
X'3B' for BCD/EBCDIC translation (T)

Example: to specify even parity, code

```
KEY     #       LEN     PARM
004F    0001    0001    23
```

IPLTXTID specification - Key = X'0051'

DALIPLTX specifies the name of a TCAM network control program. This key is mutually exclusive with the DSNAME, MEMBER NAME, and QNAME keys. When this key is specified, # must be one, LEN is the length of the name (maximum of 8), and PARM contains the name.

Example: to specify an IPLTXTID value of PGM, code

```
KEY     #       LEN     PARM
0051    0001    0003    D7 C7 D4
```

Diagnostic Trace specification (DIAGNS = TRACE) - Key = X'0054'

DALDIAGN requests the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's workarea and the user's DCB. When this key is specified, # must be zero. LEN and PARM must not be specified. (GTF must be active in the system while the job that requested the trace is running.)

Example: to specify the diagnostic trace specification, code

```
KEY     #       LEN     PARM
0054    0000    -       -
```

FUNC= specification - Key = X'005A'

DALFUNC can be used with BSAM and QSAM and specifies the type of data set to be opened for the 3525 Card Read-Punch-Print. When this key is specified, # and LEN must be one, and PARM must be one of the following values:

X'10' for W
X'12' for WT
X'14' for WX
X'16' for WXT
X'20' for P
X'30' for PW
X'34' for PWX
X'36' for PWXT
X'40' for R
X'50' for RW
X'52' for RWT
X'54' for PWX
X'56' for PWXT
X'60' for RP
X'68' for RPD
X'70' for RPW
X'74' for RPWX
X'76' for RPWXT
X'78' for RPWD
X'80' for I

Where:

- D is data protection for a punch data set
- I is interpret punch data set
- P is punch
- R is read
- T is two line printer
- W is print
- X is printer

Notes: If this information is not supplied by any source, the system assumes P.

- D, X, and T cannot be coded alone.
- If D is specified as part of a value, the FCB image-id key must also be specified giving the image identifier for the data protection image.

Example: to specify FUNC=RPWD, code

```
KEY     #       LEN     PARM
005A    0001    0001    78
```

FRID= specification - Key = X'005B'

DALFRID specifies the last four characters of a SYS1.IMAGELIB member to be used in the interpretation of documents for input to the IBM 3886 optical character reader. The characters must be alphanumeric or national, and if the member has a name with a length less than four, the entire name must be specified. This key is mutually exclusive with the FCB key. When this key is specified, # must be one, LEN is the number of characters specified, and PARM contains the characters of the member name.

Example: to specify the last four characters of member name SHARK1, code

```
KEY     #       LEN     PARM
005B    0001    0004    C1 D9 D2 F1
```

Hex Test Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0050	DALPASSW	Specifies the password for a protected data set.
0052	DALPERMA	Specifies the permanently allocated attribute.
0053	DALCNVRT	Specifies the convertible attribute.
0055	DALRTDDN	Requests the return of the associated ddname.
0056	DALRTDSN	Requests the return of the allocated data set's name.
0057	DALRTORG	Requests the return of data set organization.
005D	DALRTVOL	Requests the return of the volume serial number.

Figure 10. Non-JCL Dsname Functions (Used with Verb Code 01) – Text Unit Keys, Mnemonics, and Functions

Non-JCL Dynamic Allocation Functions

The keys described below and listed in Figure 10 do not have JCL equivalents, rather they have meaning only to dsname allocation.

Password specification - Key = X'0050'

DALPASSW specifies the password of a password-protected data set. The dsname key must also be specified when this key is specified. When this key is specified, # must be one, LEN contains the length of password, and PARM contains the password.

Example: to specify the password, MYKEY, code

```
KEY      #        LEN      PARM
0050     0001     0005     D4 E8 D2 C5 E8
```

Permanently Allocated Attribute specification - Key = X'0052'

DALPERMA specifies that the permanently allocated attribute is to be assigned to this allocation. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify assignment of the permanently allocated attribute, code

```
KEY      #        LEN      PARM
0052     0000     -        -
```

Convertible Attribute specification - Key = X'0053'

DALCNVRT specifies that the convertible attribute is to be assigned to this allocation. (Note: this specification is defaulted if the permanently allocated attribute text unit is not specified.) When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify assignment of the convertible attribute, code

```
KEY      #        LEN      PARM
0053     0000     -        -
```

Ddname Return specification - Key = X'0055'

DALRTDDN specifies that the ddname that is associated with the allocation be returned to the dynamic allocation caller. When this key is specified, # must be one, LEN must be eight, and PARM is an eight byte field. Dynamic allocation will place the allocated ddname in PARM and update LEN to the length of this ddname.

Example: to specify that the allocated ddname be returned, code

```
KEY     #       LEN     PARM
0055    0001    0008    --------
```

This specification would be updated for the allocation of the ddname DD1 as follows:

```
KEY     #       LEN     PARM
0055    0001    0003    C4 C4 F1-----
```

Dsname Return specification - Key = X'0056'

DALRTDSN specifies that the dsname that is allocated be returned to the dynamic allocation caller. When this key is specified, # must be one, LEN must be forty-four, and PARM is a forty-four byte field. Dynamic allocation will place the allocated dsname in PARM and update LEN to the length of this dsname.

Example: to specify that the allocated dsname be returned, code

```
KEY     #       LEN     PARM
0056    0001    002C    --------...--
```

This specification would be updated for the allocation of the dsname ABC as follows:

```
KEY     #       LEN     PARM
0056    0001    0003    C1 C2 C3-----...--
```

DSORG Return specification - Key = X'0057'

DALRTORG specifies that the data set organization be returned to the dynamic allocation caller. When this key is specified, # must be one, LEN must be two, and PARM is a two-byte field. Dynamic allocation will set PARM as follows:

X'0000' if DSORG cannot be determined by dynamic allocation
X'0004' if TR
X'0008' if VSAM
X'0020' if TQ
X'0040' if TX
X'0080' if GS
X'0200' if PO
X'0300' if POU
X'0400' if MQ
X'0800' if CQ
X'1000' if CX
X'2000' if DA
X'2100' if DAU
X'4000' if PS
X'4100' if PSU
X'8000' if IS
X'8100' if ISU

Example: to specify that the DSORG be returned, code

```
KEY     #       LEN     PARM
0057    0001    0002    -
```

This specification would be updated for a DSORG of PS as follows:

```
KEY     #       LEN     PARM
0057    0001    0002    4000
```

Volume Serial Return specification - Key = X'005D'

DALRTVOL specifies that the volume serial associated with the data set being allocated be returned. Only the first volume serial of the multiple-volume data set is returned, and volume sequence number, if any, is ignored. When this key is specified, # must be one, LEN must be six, and PARM is a six-byte field. Dynamic allocation will place the allocated volume serial in PARM if it is available at the completion of allocation. If the volume serial is not available at the completion of the allocation, LEN will be zero.

The volume serial will not be available at the completion of allocation if either of the following is true:

- No volume serial is allocated to the data set (a VIO or job entry subsystem data set)
- The request results in the allocation of a new data set on magnetic tape without a specific volume serial having been assigned.

Example: to specify that the allocated volume serial be returned, code

```
KEY      #        LEN      PARM
005D     0001     0006     ------
```

Note: This specification would be updated for the allocation of data set ABC on volume 123456 as follows:

```
KEY              LEN      PARM
005D     0001    0006     F1 F2 F3 F4 F5 F6
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DUNDDNAM	Specifies the ddname of the resource to be unallocated.
0002	DUNDSNAM	Specifies the data set to be unallocated.
0003	DUNMEMBR	Specifies the PDS member to be unallocated.
0005	DUNOVDSP	Specifies an overriding disposition.
0007	DUNUNALC	Specifies unallocation even if the resource has the permanently allocated attributed.
0008	DUNREMOV	Specifies removal of the "in-use" attribute, even if the resource does have the permanently allocated attribute.
000A	DUMOVSNH	Specifies "nohold" status for an unallocated SYSOUT data set.
0018	DUNOVCLS	Specifies an overriding SYSOUT class.
0058	DUNOVSUS	Specifies an overriding remote workstation.
0059	DUMOVSHQ	Puts the SYSOUT data set on the hold queue and overrides previous "nohold" specifications.

Figure 11. Dynamic Unallocation (Verb Code 02) – Text Unit Keys, Mnemonics, and Functions

Dynamic Unallocation Text Units

The following text units are used with the dynamic unallocation (verb code X'02').

Ddname specification - Key = X'0001'

DUNDDNAM specifies the ddname of the resource to be unallocated. When this key is specified, # must be one, LEN is the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname DD1, code

```
KEY     #       LEN     PARM
0001    0001    0003    C4 C4 F1
```

Dsname specification - Key = X'0002'

DUNDSNAM specifies the data set name to be unallocated. When this key is specified, # must be one, LEN contains the length of the dsname, and PARM contains the dsname.

Example: to specify the dsname MYDATA, code

```
KEY     #       LEN     PARM
0002    0001    0006    D4 E8 C4 C1 E3 C1
```

Membername specification - Key = X'0003'

DUNMEMBR specifies that a particular member of the data set is to be unallocated. Dsname must also be specified when this key is specified. When this key is specified, # must be one, LEN is the length of the member name, and PARM contains the membername.

Example: to specify the membername MEM1, code

```
KEY     #       LEN     PARM
0003    0001    0004    D4 C5 D4 F1
```

Overriding Disposition specification - Key = X'0005'

DUNOVDSP specifies a disposition that overrides the disposition assigned to a data set when it was allocated. When this key is specified, # and LEN must be one, and PARM contains:

X'01' for an overriding disposition of UNCATLG
X'02' for an overriding disposition of CATLG
X'04 for an overriding disposition of DELETE
X'08' for an overriding disposition of KEEP

Example: to specify an overriding disposition of CATLG, code

```
KEY     #       LEN     PARM
0005    0001    0001    02
```

Note: This key will be ignored if any of the following situations are true:

- The data set was originally allocated with a disposition of PASS.
- The data set was a VSAM data set.
- The data set was a non-subsystem data set that had a system-generated name.

Note: When this key is ignored, the request is still processed but the disposition on the original request will be used.

Unalloc Option specification - Key = X'0007'

DUNUNALC specifies that unallocation is to occur even if the resource has the permanently allocated attribute. The remove option specification (see below) is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the unalloc option, code

```
KEY     #       LEN     PARM
0007    0000    -       -
```

Remove Option specification - Key = X'0008'

DUNREMOV specifies that the in-use attribute is to be removed even if the resource does not have the permanently allocated attribute. The unalloc option key is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the remove option, code

```
KEY     #       LEN     PARM
0008    0000    -       -
```

Overriding SYSOUT Nohold specification - Key = X'000A'

DUMOVSNH specifies that the SYSOUT data set being unallocated is not to be placed on the hold queue. This specification overrides the HOLD/NOHOLD specification assigned when the data set was allocated. This key is ignored if the data set is not a SYSOUT data set. The overriding hold key (see below) is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify nohold, code

```
KEY     #       LEN     PARM
000A    0000    -       -
```

Overriding SYSOUT Class specification - Key = X'0018'

DUNOVCLS specifies a SYSOUT class which overrides the class assigned when the SYSOUT data set was allocated. This key is ignored for non-SYSOUT data sets. When specified, # and LEN must be one, and PARM contains the overriding class.

Example: to specify an overriding class of C, code

```
KEY     #       LEN     PARM
0018    0001    0001    C3
```

Overriding SYSOUT Remote Workstation specification - Key = X'0058'

DUNOVSUS specifies that the SYSOUT data set being unallocated is to be routed to a remote user. This specification overrides the remote workstation specification assigned when the data set was allocated. This specification is ignored if the data set is not a SYSOUT data set. When this key is specified, # must be one, LEN is the length of the remote workstation name (maximum of 8), and PARM contains the remote user name.

Example: to specify the remote work station USER01, code

```
KEY     #       LEN     PARM
0058    0001    0006    E4 E2 C5 D9 F0 F1
```

Overriding SYSOUT hold queue specification - Key = X'0059'

DUMOVSHQ specifies that the SYSOUT data set being unallocated is to be placed on the hold queue. This specification overrides the hold/nohold key assigned when the data set was allocated. This key is ignored if the data set is not a SYSOUT data set. The overriding nohold key is mutually exclusive with this key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify hold, code

```
KEY     #       LEN     PARM
0059    0000    -       -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DCCDDNAM	Specifies the ddnames to be concatenated.
0004	DCCPERMC	Specifies the permanently concatenated attribute.

Figure 12. Dynamic Concatenation (Verb Code 03) – Text Unit Keys, Mnemonics, and Functions

Dynamic Concatenation Text Units

The text units for dynamic concatenation (verb code X'03') are described below and listed in Figure 12.

Ddname specification - Key = X'0001'

DCCDDNAM specifies the ddnames that are associated with the data sets to be concatenated. When this key is specified, # is the number of ddnames being specified (a minimum of two), LEN is length of the immediately following ddname field and PARM contains a ddname.

Example: to specify concatenation of SYSLIB to MYLIB, code

```
KEY     #       LEN     PARM          LEN     PARM
0001    0002    0005    D4E8D3C9C2    0006    E2E8E2D3C9C2
```

Permanently Concatenated Attribute specification - Key = X'0004'

DCCPERMC specifies that the concatenated group be assigned the permanently concatenated attribute. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify assignment of the permanently concatenated attribute, code

```
KEY     #       LEN     PARM
0004    0000    -       -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DDCDDNAM	Specifies the ddname of the group to be deconcatenated.

Figure 13. Dynamic Deconcatenation (Verb Code 04) – Text Unit Key, Mnemonic, and Function

Dynamic Deconcatenation Text Units

The text unit for dynamic deconcatenation (verb code X'04') is described below and listed in Figure 13.

Ddname specification - Key = X'0001'

DDCDDNAM specifies the ddname of the concatenated group that is to be deconcatenated. This key must be specified. In specifying this text unit, # must be one, LEN is the length of the ddname field and PARM contains the ddname.

Example: to specify the ddname, DD1, code

```
KEY      #        LEN      PARM
0001     0001     0003     C4 C4 F1
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DRITCBAD	Removes the "in-use" attribute from all resources associated with the specified TCB address.
0002	DRICURNT	Removes the "in-use" attribute from all resources but those of the current task and its higher-level tasks.

Figure 14. Remove In-Use Processing Based on Task-Id (Verb Code 05) – Text Unit Keys, Mnemonics, and Functions

Text Units for Removing the In-Use Attribute Based on Task-Id

The text units for remove in-use processing based on task-id (verb code X'05') are described below and listed in Figure 14.

TCB Address specification - Key = X'0001'

DRITCBAD specifies that the in-use attribute is to be removed from all resources associated with the specified TCB address. The current task option key (see below) is mutually exclusive with this key. When this key is specified, # must be one, LEN must be four, and PARM contains the TCB address.

Example: to specify the TCB address 22AC0, code

```
KEY     #       LEN     PARM
0001    0001    0004    00022AC0
```

Current Task Option specification - Key = X'0002'

DRICURNT specifies that the in-use attribute is to be removed from all resources except those associated with the current task, its direct ancestors, and the initiator. This key is mutually exclusive with the TCB address key. When this key is specified, # must be zero. LEN and PARM are not specified.

Example: to specify the current task option, code

```
KEY     #       LEN     PARM
0002    0000    -       -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DDNDDNAM	Specifies the ddname to be allocated.
0002	DDNRTDUM	Requests a dummy data set indication.

Figure 15. Ddname Allocation (Verb Code 06) – Text Unit Keys, Mnemonics, and Functions

Ddname Allocation Text Units

The following text units described below and listed in Figure 15 are used with the ddname allocation (verb code X'06').

Ddname specification - Key = X'0001'

DDNDDNAM specifies the ddname to be allocated. This text unit must be specified. When this key is specified, # must be one, LEN contains the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname SYSLIB, code

```
KEY      #        LEN      PARM
0001     0001     0006     E2 E8 E2 D3 C9 C2
```

Return DUMMY Indication specification - Key = X'0002'

DDNRTDUM requests an indication if a DUMMY data set is associated with the specified ddname. When this key is specified, # and LEN must be one, and PARM is a one-byte field. Dynamic allocation sets PARM as follows:

X'80' if a DUMMY data set is associated with the ddname
X'00' otherwise

Example: to specify that the DUMMY indication be returned, code

```
KEY      #        LEN      PARM
0002     0001     0001     -
```

Hex Text Unit Key	IEFZB4D2 Mnemonic	Dynamic Allocation Function
0001	DINDDNAM	Specifies the ddname identifier of the requested information.
0002	DINDSNAM	Specifies the data set for which the information is requested.
0004	DINRTDDN	Requests the return of the associated ddname.
0005	DINRTDSN	Requests the return of the data set name.
0006	DINRTMEM	Requests the return of the PDS membername.
0007	DINRTSTA	Requests the return of the data set's status.
0008	DINRTNDP	Requests the return of the data set's normal disposition.
0009	DINRTCDP	Requests the return of the data set's conditional disposition.
000A	DINRTORG	Requests the return of the data set's organization.
000B	DINRTLIM	Requests the number of resources that must be unallocated before making a new allocation.
000C	DINRTATT	Requests the return of special attribute indications.
000D	DINRTLST	Requests the return of a last relative entry indication.
000E	DINRTTYP	Requests the return of the data set's type (terminal or dummy).
000F	DINRELNO	Specifies the desired allocation information retrieval by relative request number.

Figure 16. Dynamic Information Retrieval (Verb Code 07) – Text Unit Keys, Mnemonics, and Functions

Dynamic Information Retrieval Text Units

The text units for dynamic information retrieval (verb code '07') are described below and listed in Figure 16.

Ddname specification - Key = X'0001'

DINDDNAM specifies a ddname that identifies the allocation about which information is to be returned. It is mutually exclusive with the dsname and relative entry key. When this key is specified, # must be one, LEN is the length of the ddname field, and PARM contains the ddname.

Example: to specify the ddname DD1, code

```
KEY      #        LEN      PARM
0001     0001     0003     C4 C4 F1
```

Dsname specification - Key = X'0002'

DINDSNAM specifies a dsname that identifies the allocation about which information is to be returned. It is mutually exclusive with the ddname and relative entry keys. When specified, # must be one, LEN is the length of the dsname, and PARM contains the dsname.

Example: to specify the dsname MYDATA, code

```
KEY      #        LEN      PARM
0002     0001     0006     D4 E8 C4 C1 E3 C1
```

Return Ddname specification - Key = X'0004'

Code DINRTDDN to determine that the ddname associated with the specified allocation. When this key is specified, # must be one, LEN must be eight, and PARM is an eight-byte field. Upon return to the caller, PARM will contain the ddname and LEN will be set to its length.

Example: to specify that the ddname be returned, code

```
KEY      #        LEN      PARM
0004     0001     0008     --------
```

Return Dsname specification - Key = X'0005'

Code DINRTDSN to determine the dsname associated with the specified allocation be returned. When this key is specified, # must be one, LEN must be forty-four, and PARM is a forty-four byte field. Upon return to the caller, PARM will contain the dsname and LEN will be set to its length.

Example: to specify that the dsname be returned, code

```
KEY     #       LEN     PARM
0005    0001    002C    --------
```

Return Membername specification - Key = X'0006'

Code DINRTMEM to determine the membername associated with the specified allocation. When this key is specified, # must be one, LEN must be eight, and PARM is an eight-byte field. Upon return to the caller, PARM will contain the membername and LEN will be set to its length (zero if none).

Example: to specify that the membername be returned, code

```
KEY     #       LEN     PARM
0006    0001    0008    --------
```

Return Status specification - Key = X'0007'

Code DINRTSTA to determine the data set status of the specified allocation. When this key is specified, # and LEN must be one. PARM is a one-byte field. Upon return to the caller, the PARM field will contain on of the following:

X'01' for OLD
X'02' for MOD
X'04' for NEW
X'08' for SHR

Example: to specify that the status be returned, code

```
KEY     #       LEN     PARM
0007    0001    0001    -
```

Return Normal Disposition specification - Key = X'0008'

Code DINRTNDP to determine the normal disposition of the specified allocation. When this key is specified, # and LEN must be one. PARM is a one-byte field. Upon return to the caller, PARM will contain one of the following:

X'01' for UNCATLG
X'02' for CATLG
X'04' for DELETE
X'08' for KEEP
X'10' for PASS

Example: to specify that the normal disposition be returned, code

```
KEY     #       LEN     PARM
0008    0001    0001    -
```

Return Conditional Disposition specification - Key = X'0009'

Code DINRTCDP to determine the data set conditional disposition of the specified allocation. The values for #, LEN and PARM are the same as for return normal disposition key.

Example: to specify that the conditional disposition be returned, code

```
KEY      #        LEN      PARM
0009     0001     0001     -
```

Return Data Set Organization specification Key = X'000A'

Code DINRTORG to determine the data set organization of the specified allocation. When this key is specified, # must be one, LEN must be two, and PARM is a two-byte field. Upon return to the caller, PARM will contain one of the following:

X'0000' if undetermined
X'0004' if TR
X'0008' for VSAM
X'0020' if TQ
X'0040' if TX
X'0080' for GS
X'0200' for PO
X'0300' for POU
X'0400' for MQ
X'0800' for CQ
X'1000' for CX
X'2000' for DA
X'2100' for DAU
X'4000' for PS
X'4100' for PSU
X'8000' for IS
X'8100' for ISU

Example: to specify that the data set organization be returned, code

```
KEY      #        LEN      PARM
000A     0001     0002     --
```

Return Limit specification - Key = X'000B'

Code DINRTLIM to determine the number of resources that must be unallocated before a request can be made that requires a new allocation. When this key is specified, # must be one, LEN must be two, and PARM is a two-byte field. Upon return to the caller, PARM is set to the number of resources to unallocated.

Example: to specify that the number be returned, code

```
KEY      #        LEN      PARM
000B     0001     0002     --
```

If three resources must be unallocated, this text unit is returned as follows:

```
KEY      #        LEN      PARM
000B     0001     0002     0003
```

Return Dynamic Allocation Attribute specification - Key = X'000C'

Code DINRTATT to determine if the specified allocation has the permanently allocated, convertible, in-use, permanently concatenated, and dynamically allocated attributes. When this key is specified, # and LEN must be one, and PARM is a one-byte field. Upon return to the caller, PARM is set as follows:

Bit 0,	on if, permanently concatenated attribute
Bit 1,	on if, in-use attribute
Bit 2,	on if, permanently allocated attribute
Bit 3,	on if, convertible attribute
Bit 4,	on if, dynamically allocated attribute
Bits 5-7	reserved

Example: to specify that the data set attributes be returned, code

```
KEY     #       LEN     PARM
000C    0001    0001    -
```

If the allocation has the in-use and permanently allocated attributes, this field is returned as follows:

```
KEY     #       LEN     PARM
000C    0001    0001    60
```

Return Last Entry specification - Key = X'000D'

Code DINRTLST to determine if the relative entry number specified corresponds to the last relative entry. When this key is specified, # and LEN must be one, and PARM is a one-byte field. Upon return to the caller, PARM will be set as follows:

X'80'	if last relative entry
X'00'	otherwise

Example: to specify that the last entry indication be returned, code

```
KEY     #       LEN     PARM
000D    0001    0001    -
```

Return Data Set Type specification - Key = X'000E'

Code DINRTTYP to determine if the specified allocation is a DUMMY data set, a terminal allocation, a SYSIN data set, or a SYSOUT data set. When this key is specified, # and LEN must be one, and PARM is a one-byte field. Upon return to the caller, PARM is set as follows:

X'80'	if a DUMMY data set
X'40'	if a terminal
X'20'	if a SYSIN data set
X'10'	if a SYSOUT data set
X'00'	otherwise

Example: to specify that data set type be returned, code

```
KEY     #       LEN     PARM
000E    0001    0001    -
```

Relative Request Number specification - Key = X'000F'

DINRELNO specifies a relative request number that identifies the allocation about which you are requesting. The ddname and dsname keys are mutually exclusive with this key. When this key is specified, # must be one, LEN must be two, and PARM contains the relative number.

Example: to specify information is to be returned about the tenth request, code

```
KEY     #       LEN     PARM
000F    0001    0002    000A
```

Example of a Dynamic Allocation Request

The assembler language example in Figure 17 illustrates a dynamic allocation request for allocating SYS1.LINKLIB with a status of SHARE. It also requests that the dynamic allocation routines return the ddname associated with SYS1.LINKLIB.

Figure 18 shows the parameter list that results from the code in Figure 17.

```
LA 0,75                        AMOUNT OF STORAGE THAT THIS REQUEST NEEDS
GETMAIN R,LV=(0)               GET THE STORAGE NECESSARY FOR THE REQUEST
LR 8,1                         SAVE THE ADDRESS OF THE RETURNED STORAGE
USING S99RBP,8                 ESTABLISH ADDRESSABILITY FOR 'RBPTR' DSECT
LA 4,S99RBPTR+4                POINT FOUR BYTES BEYOND START OF 'RBPTR"
USING S99RB,4                  ESTABLISH ADDRESSABILITY FOR 'RB' DSECT
ST 4,S99RBPTR                  MAKE 'RBPTR' POINT TO 'RB'
OI S99RBPTR,S99RBPND           TURN ON THE HIGH ORDER BIT IN 'RBPTR'
XC S99RB(RBLEN),S99RB          ZERO OUT 'RB' ENTIRELY
MVI S99RBLN,RBLEN              PUT THE LENGTH OF 'RB' IN ITS LENGTH FIELD
MVI S99VERB,S99VRBAL           SET VERB CODE FIELD TO ALLOCATION FUNCTION
LA 5,S99RB+RBLEN               POINT TWENTY BYTES BEYOND START OF 'RB'
USING S99TUPL,5                ESTABLISH ADDRESSABILITY FOR TEXT UNIT PTRS
ST 5,S99TXTPP                  INITIALIZE THE TEXT POINTERS ADDRESS IN 'RB'
LA 6,S99TUPL+12                POINT JUST PAST THE THREE TEXT UNIT POINTERS
USING S99TUNIT,6               SET ADDRESSABILITY FOR THE FIRST TEXT UNIT
ST 6,S99TUPTR                  POINT 1ST TEXT UNIT POINTER TO 1ST TEXT UNIT
LA 7,DALDSNAM                  GET THE KEY FOR DSNAME
STH 7,S99TUKEY                 PUT THE KEY IN THE TEXT UNIT KEY FIELD
LA 7,1                         BECAUSE THE DSNAME KEY REQUIRES ONLY ONE
STH 7,S99TUNUM                  PARAMETER, LOAD AND STORE 1 IN NUMBER FIELD
LA 7,L'LINKDSN                 GET THE LENGTH OF THE DSNAME FIELD AND PUT
STH 7,S99TULNG                  IT INTO THE TEXT UNIT'S LENGTH FIELD
MVC S99TUPAR(12),LINKDSN       PUT THE DSNAME INTO TEXT UNIT PARM FIELD
LA 6,S99TUNIT+18               POINT JUST PAST THE FIRST TEXT UNIT
LA 5,S99TUPL+4                 POINT TO THE 2ND TEXT UNIT POINTER IN LIST
ST 6,S99TUPTR                  POINT 2ND TEXT UNIT POINTER TO 2ND TEXT UNIT
LA 7,DALSTATS                  GET THE KEY FOR STATUS SPECIFICA-
STH 7,S99TUKEY                  TION AND PUT THE KEY IN THE TEXT UNIT
LA 7,1                         BECAUSE THE STATUS KEY REQUIRES ONLY ONE
STH 7,S99TUNUM                  PARAMETER, LOAD AND STORE 1 IN THE NUMBER FIELD
STH 7,S99TULNG                 SET THE STATUS PARM LENGTH FIELD ALSO TO 1
MVI S99TUPAR,X'08'             SET THE PARM FIELD TO INDICATE SHARE DISP
LA 6,S99TUNIT+7                POINT JUST PAST THE SECOND TEXT UNIT
LA 5,S99TUPL+4                 POINT TO 3RD TEXT UNIT POINTER IN THE LIST
ST 6,S99TUPTR                  POINT 3RD TEXT UNIT POINTER TO 3RD TEXT UNIT
OI S99TUPTR,S99TUPLN           TURN ON HIGH ORDER BIT TO INDICATE LAST PTR
LA 7,DALRTDDN                  GET THE KEY FOR 'RETURN DDNAME' AND
STH 7,S99TUKEY                  PUT THE KEY IN THE TEXT UNIT KEY FIELD
LA 7,1                         BECAUSE 'RETURN DDNAME' KEY REQUIRES ONLY 1
STH 7,S99TUNUM                  PARAMETER, LOAD AND STORE 1 IN NUMBER FIELD
LA 7,8                         SET LENGTH OF FIELD FOR RETURNING DDNAME TO 8
STH 7,S99TULNG
LR 1,8                         PUT REQ BLK PTR ADDR IN REG 1 FOR DYNALLOC
DYNALLOC                       INVOKE DYNAMIC ALLOCATION TO PROCESS REQUEST
  .
  .
  .
LINKDSN  DC  C'SYS1.LINKLIB'
  .
  .
  .
 IEFZB4D0
 IEFZB4D2
   .
   .
   .
RBLEN    EQU (S99RBEND-S99RB)
```

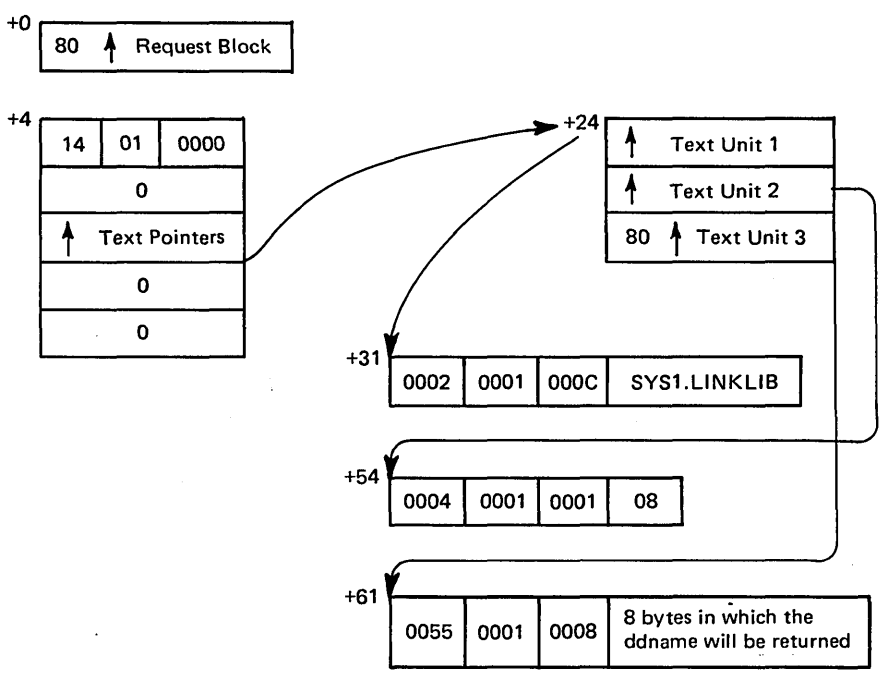
Figure 17. Example of a Dynamic Allocation Request

Note the following concepts that the example illustrates:

- Storage is obtained via a GETMAIN macro instruction.
  In the example, the requirement is for 75 bytes, derived as follows:

Bytes	Purpose
4	Pointer to the request block.
20	Request block space.
12	Four bytes each for three text unit pointers.
18	Text unit space for the data set name.
7	Text unit space for the data set status.
14	Text unit space for the requested return of the ddname.

- The parameter structure is mapped by the DSECTs that IEFZB4D0 provides.
- The example uses IEFZB4D2 mnemonics in the text unit keys.



Note: The storage in this figure is contiguous; however, it need not be contiguous.

Figure 18. Resulting Parameter List From Dynamic Allocation Example

Chapter 4: Internal Readers

The internal reader facility allows a programmer to build a job and place it directly in the input stream, without using intermediate devices such as a card punch (to transform the JCL into cards) and card readers (to place those cards in the input stream). An internal reader is simply a data set that the system places in the input stream. Any data (such as JCL) you place in the internal reader, therefore, is placed in the input stream and subsequently passed to the job entry subsystem for execution. MVS uses internal readers to pass JCL for started tasks and logon requests to the job entry subsystem.

To use the internal reader facility, your program must allocate an internal reader either dynamically or on a DD statement. You use the SYSOUT parameter (or text units), specifying an output class and INTRDR (for example, SYSOUT=(A,INTRDR)). Messages for a job placed in the internal reader are assigned the same class as that specified in the SYSOUT parameter for the internal reader, unless you include the MSGCLASS parameter on the JOB statement for the job.

The following paragraphs describe how a user-written program can dynamically allocate and use an internal reader.

Dynamically Allocating an Internal Reader

Use the following text unit keys to allocate the internal reader:

- DALSYSOU – specifies the SYSOUT data set and defines its class
- DALSPGNM – specifies the SYSOUT program name (for example, INTRDR)
- DALCLOSE – specifies unallocating INTRDR at CLOSE
- DALRTDDN – specifies the return of the associated ddname assigned by dynamic allocation

Chapter 3 of this publication describes the format of these text units and how to invoke dynamic allocation. The TSO SUBMIT command is an example of a program that dynamically allocates an internal reader using these text units.

Opening the Internal Reader

You open the internal reader data set using a VSAM Access-Method Control Block (ACB), specifying the ddname returned by dynamic allocation and the keyword MACRF=(ADR,SEQ,OUT) where:

- ADRs specifies addressed data processing with no index references
- SEQ specifies sequential processing
- OUT specifies writing the data

Passing JCL Records and Jobs to the Internal Reader

Issue a PUT to pass the input JCL record to the INTRDR data set and use the VSAM request parameter list (RPL) for both PUTs and ENDREQs specifying OPT=(ADR,SEQ,SYN,NUP),RECLEN=80 where:

- ADR specifies addressed data processing with no index references
- SEQ specifies sequential processing
- SYN specifies a synchronous request and control should be returned after completion of this request
- NUP specifies nonupdate mode; records being retrieved are not updated or deleted

- RECLEN=80 specifies that the submitted JCL records are 80 bytes in length

You should dynamically specify the ACB address and the address of the JCL record. Issue a ENDREQ, using the VSAM RPL, to tell the job entry subsystem that all JCL records for this job have been "put" into the internal reader. The subsystem will enter the job for background processing and return the jobid it assigned in RPLRBAR. A CLOSE should be issued after the final ENDREQ.

Chapter 5: Job Scheduler Restarting Support

Job scheduler restarting support allows a failing job to resume or terminate processing. This function collects job status and job-related control block information for the reconstruction of a failing job's scheduler work area (SWA). This reconstruction of the failing job's SWA is necessary to support the following restart situations.

- Automatic step – permits execution to resume at the beginning of a job step
- Automatic checkpoint – permits execution to resume from the most recently executed checkpoint in the user's program
- Deferred checkpoint – permits execution to resume from a user-specified checkpoint upon resubmission of a job
- System – permits the termination of active jobs in the event of a system failure
- Continue – permits a job to continue at the next job step if the system fails during step termination

For a detailed discussion of the checkpoint/restart facility, refer to *OS/VS Checkpoint/Restart*.

Job Journal

The job journal is a temporary sequential data set that resides on the spool volume of the job entry subsystem (JES). It preserves a set of selected job-related control blocks that are necessary for restart processing.

The job journal is necessary because scheduler control blocks are maintained in the SWA in pageable storage. When the system fails, the address space containing the SWA is lost. When a job abnormally terminates, the job's SWA is released. Reconstruction of the SWA is possible because the job journal preserves up-to-date copies of the essential control blocks. This facility is necessary in the following restart situations:

- Automatic step
- Automatic checkpoint
- Continue
- System

Unless the user specifies no job journal via the JES initialization parameters, each job is provided with a job journal. Without a job journal, the capability for automatic restarting is lost. In addition, jobs that are executing when the system fails forfeit their data set disposition processing if there is no job journal. For additional information about a "no journal" environment, refer to the discussion of the NOJOURN initialization parameter, under "Job Class Parameters," in *OS/VS2 System Programming Library: JES2*.

Two service routines process the job journal. They are:

- Journal write routine, which determines which scheduler control blocks are necessary to restart a job and writes them to the job journal.
- Journal merge routine, which merges the control blocks from the job journal to the SWA during restarts. This routine reconstructs the SWA to its condition prior to the job or system failure.

The job journal contains the following records:

- Step header
- Job control table (JCT)
- Step control table (SCT)
- Step input/output table (SIOT)
- Job file control block and its extension (JFCB and JFCBX/JFCBE)
- Passed data set information block (PDI block)
- Generation data group name table (GDGNT)
- Volume unload table (VUT)
- Account control table (ACT)
- Virtual input/output data set control blocks (VDSCBs - virtual data set control blocks, and DSPCT – data set page control table header)

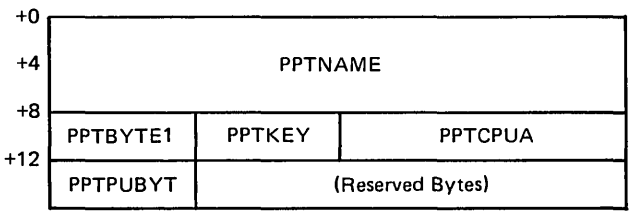
For additional information about restart processing, refer to *OS/VS2 System Logic Library*.

Chapter 6: Assigning Special Program Properties

The installation can assign special properties to programs by placing the program names in the program properties table (IEFSDPPT), a nonexecutable CSECT in load module IEFSD060. Executing the SGIEF0PT macro during system generation makes the table available for the initiators to use. When the initiator initiates a program, it scans the table to determine whether special properties apply to the program.

Format and Contents of Program Properties Table (PPT)

Each entry in the PPT is 16 bytes long, the final entry being X'FF' followed by 15 bytes containing zeros. Each entry has the following format:



Program Name (PPTNAME): the eight-byte name specified in the PGM parameter on the EXEC statement for the job.

Program Properties (PPTBYTE1): a one-byte field that indicates the special properties assigned to a program.

The following describes the contents of PPTBYTE1:

Bit	Bit Name	Meaning When On
1000 0000	PPTNCNCL	The program cannot be canceled.
0100 000	PPTSKEY	Unique protection key is to be assigned to the program. The key is defined in the next byte of the table entry (PPTKEY).
0010 0000	PPTNSWP	The program is nonswappable.
0001 0000	PPTPRIV	The program is privileged. It will not be swapped unless the address space is in a long wait.
0000 1000	PPTSYSTK	The program is a system task that will not be timed. (When a program is not timed, the system does not check for time limits. The program must be a one-step program initiated with a START or MOUNT command.)
0000 0100	PPTNDSI	The program does not require private use of the data sets that it requests, that is, the program does not need to obtain exclusive control over the data set to maintain data set integrity. It must be a one-step program.
0000 0010	PPTNOPAS	The program is to bypass password protection.
0000 0001		Reserved.

Notes:

1. The properties represented by the various bit settings might not be honored by the system. An example is: a program is assigned special properties only if it resides in an APF-authorized library and all JOBLIB and STEPLIBs are APF-authorized libraries.
2. The requirements of the initiator influence the need to maintain data set integrity as follows:
   - If one or more data sets requested by a program are not available when the job is to be initiated, the scheduler waits until the job can acquire control of all data sets that it

requires. Although the job itself may not require data set integrity, the initiation process for the job does.

- Jobs that request the no-data-set-integrity property (bit 5) will not be initiated if *both* of the following conditions exist:
  - The job requests a data set whose name is an alias for a data set that is unavailable during the job's initiation.
  - The job contains either a JOBLIB or STEPLIB.

Protection Key (PPTKEY): a one-byte field that specifies in bits 0-3 the unique protection key to be assigned to the program. (A protection key is assigned if bit 1 of the preceding byte (PPTBYTE1) is on.)

Affinity Mask (PPTCPUA): a halfword that indicates processor affinity, which is a system generation option. Each bit in the 16-bit mask refers to a corresponding processor identifier (0-F) assigned during system generation. For example, bit 0 corresponds to processor 0. If bit 0 is on, the program is eligible to run on processor 0. The bit mask should be set to X'FFFF' if affinity is not required.

Preferred Storage Flags (PPTPUBYT): a one-byte field whose flags indicate whether LSQA and private area fixed pages require frames in preferred storage (nonreconfigurable and non V=R storage). Use these flags for programs whose fixed pages could prevent the successful execution of a VARY STOR,OFFLINE command (or could fragment the V=R area) if those fixed pages were assigned frames in reconfigurable or V=R storage. The first two flags are meaningful for swappable programs (PPTNSWP=0) with a special requirement for preferred frames. The third flag is used only by the SYSEVENT TRANSWAP. Following is a description of the flags:

- PPT2LPU (1000 0000) – when this bit is on, the system assigns all private area short-term fixed pages to preferred frames.
- PPT1LPU (0100 0000) – when this bit is on, the system assigns all private area long-term fixed pages and LSQA pages to preferred storage frames.
- PPTN2LP (0010 0000) – when this bit is on, the system need not assign private area short-term fixed pages to preferred storage frames. (This flag is meaningful only for users of the SYSEVENT TRANSWAP: V=R job steps, nonswappable programs, applications using the BTAM OPEN function, and any applications using a system function that issues SYSEVENT TRANSWAP.)

The initiator maps the preferred storage flags to corresponding flags in the ASCB. The ASCB flags determine how the system allocates frames to the address space.

If PPT1LPU=1 and PPT2LPU=0, the initiator sets ASCB1LPU in the ASCB to 1. If PPT2LPU=1, the initiator sets both ASCB1LPU and ASCB2LPU in the ASCB to 1, regardless of the value of PPT1LPU and PPTN2LP. The value of PPTN2LP is copied to ASCBN2LP. ASCBN2LP merely prevents SYSEVENT TRANSWAP from setting ASCB1LPU to 1 as the address space changes to a nonswappable state; if ASCB2LPU is 1 before the TRANSWAP, it is not reset to 0. The topic, "Examples of Using Preferred Storage Flags" summarizes the effect of the preferred storage flags on the allocation of frames during execution of the program. The five low-order bits of the preferred storage flag-byte are reserved for future use.

Tips for Using the Preferred Storage Flags: The following tips apply to all three flags:

- Do one of the following for an application program that issues SYSEVENT DONTSWAP or issues SYSEVENT REQSWAP and then a SYSEVENT DONTSWAP:
  - List the program in the PPT with the first two preferred storage flags set as desired. This allows the program to be attached as swappable, but all LSQA and private area fixed pages will be assigned preferred frames during the entire job step.
  - Remove SYSEVENTs REQSWAP and DONTSWAP from the program and list the program in the PPT as nonswappable (PPTNSWP=1) and set the third flag (PPTN2LP) as desired. This allows the program to be attached as nonswappable, and all LSQA and private area fixed pages will be assigned preferred frames during the entire job step.
- The OLTEP program IFDOLT must be listed in the PPT with PPT1LPU and PPT2LPU set to one and PPTN2LP set to zero. This allows OLTEP to make itself nonswappable by issuing a REQSWAP and then a DONTSWAP. The system handles OLTEP in the same manner as the application program previously described.
- An I/O device requiring operator intervention can interfere with taking storage offline by fixing pages in reconfigurable storage. An example of this is a printer requiring action to be taken or a tape unit with a mount pending. Until the required action is completed, the storage associated with the I/O operation cannot be taken offline. This problem *cannot* be bypassed through the use of preferred storage flags.
- All three flags are ignored if one or more non-APF-authorized joblibs or steplibs are defined in the JCL for the job step.

The following tips apply to flag PPTN2LP:

- PPTN2LP has meaning only for programs for which SYSEVENT TRANSWAP is issued. The initiator issues TRANSWAP for V=R job steps and nonswappable programs (PPTNSWP=1). Also, the BTAM OPEN routine issues TRANSWAP. TRANSWAP causes the transition of the address space to a nonswappable state. TRANSWAP performs the same function as SYSEVENT DONTSWAP, and in addition, ensures that preferred storage is used whenever necessary.
- PPTN2LP should be set to 1 when a program's short-term fixed pages do not need to be assigned to preferred storage frames; that is, the program's short-term fixes are indeed short-term fixes and can be allowed in reconfigurable storage.

The following tip applies to flags PPT1LPU and PPT2LPU:

- PPT1LPU and PPT2LPU are intended for use with authorized swappable programs that issue SYSEVENT DONTSWAP to become nonswappable for relatively short periods (rather than setting PPTNSWP=1). Use of the preferred storage flags forces the program's private area fixed pages and LSQA pages into preferred storage frames and thus ensures they do not prevent any attempts to take storage offline.

Note: Programs need not be nonswappable to have the system assign their fixed pages to preferred storage frames.

Examples of Using Preferred Storage Flags: The following example shows the effect of setting preferred storage flags for the nonswappable program JES2.

The JES2 entry in the PPT would include the following bit values:

```
PPTNSWP =   1
PPT1LPU =   0
PPT2LPU =   0
PPTN2LP =   1
```

These values indicate nonswappability and that short-term fixes are indeed short-term fixes that can be allowed in reconfigurable storage. After the initiator issues a TRANSWAP and attaches JES2, the ASCB flags are set as shown below:

```
ASCB1LPU    =   1
ASCB2LPU    =   0
ASCBN2LP    =   1
```

These values result in LSQA and long-term fixed pages in preferred storage only; short-term fixed pages; however, are allowed in reconfigurable storage.

The next example shows the effect of setting the flags for a swappable program that issues SYSEVENT DONTSWAP.

The program's entry in the PPT would include the following bit values:

```
PPTNSWP =   0
PPT1LPU =   1
PPT2LPU =   1
PPTN2LP =   0
```

These values indicate that the program is swappable and that all fixed pages and LSQA pages must be in preferred storage. The initiator attaches the program as swappable, with the ASCB flags set as follows:

```
ASCB1LPU    =   1
ASCB2LPU    =   1
ASCBN2LP    =   0
```

The program can then issue DONTSWAP, being assured that its fixed and LSQA pages are in preferred storage and will not prevent storage from being taken offline.

Following is a summary of the most common uses of the PPT preferred storage bits:

PPTNSWP	PPT1LPU	PPT2LPU	PPTN2LP	Effect on Program
1	-	-	0	The initiator makes the address space nonswappable via the SYSEVENT TRANSWAP prior to attaching the job step. LSQA and all private area fixed pages are in preferred storage.
1	-	-	1	Same as preceding case except short-term fixed pages are allowed in reconfigurable or V=R storage.
0	1	1	-	The initiator attaches the job step as swappable. LSQA and all private area fixed pages are in preferred storage. In this case, the program can issue DONTSWAP and be assured that its fixed pages will not prevent reconfiguring storage.

Note: In this example, a dash (-) indicates the setting of the bits is irrelevant.

Updating the PPT

The PPT includes five dummy entries for adding program names. The dummy entries initially contain the properties necessary for TCAM:

- Unique protection key of 6 (PPTSKEY=1; PPTKEY=6)
- Nonswappable (PPTNSWP=1)
- Bypassing of password protection (PPTNOPAS=1)

An installation can execute the AMASPZAP service aid program to change these entries for the installation's purposes. To add more than five program names to the PPT, an installation can either update the macro SGIEF0PT prior to system generation or update the source module IEFSDPPT, then assemble and linkedit it into load module IEFSD060 again after system generation. To update the macro SGIEF0PT prior to system generation, follow these steps:

1. Obtain a deck of the macro SGIEF0PT.
2. Add the desired entries to the deck.
3. Replace SGIEF0PT in SYS1.AMODGEN with the expanded version (from step 2).

TCAM Message Control Program (MCP) names other than IFDQTCAM must be added to the program properties table (IEFSDPPT). These names must be added to the PPT after system generation by using the AMASPZAP service aid. (For information on AMASPZAP, see *OS/VS2 System Programming Library: Service Aids*.) If more than six MCPs are required, the PPT must be reassembled to create more entries. An MCP will not operate unless its name is in the PPT. TCAM OPEN routines must run in key 6 and will abnormally terminate any caller who was not initiated in key 6. Prior to MVS, MCP names were put in the PPT to make the MCP noncancellable, but the MCP would execute properly if its name were not in the PPT.

Chapter 7: System Log

The system log is an integral part of MVS. It consists of dynamically created data sets that record the communications among problem programs, operators, and the operating system. It contains operating data entered by problem programs using the write-to-log (WTL) macro instruction. The log contains the following information:

- Job time, job step time, and data from the JOB and EXEC statements of a job that has ended
- Descriptions of unusual events recorded by the operator via the LOG command
- Write-to-operator (WTO) and write-to-operator with reply (WTOR) messages
- Accepted replies to WTOR messages
- Commands issued through operator's consoles and the input stream and commands issued by the system

If the installation does not modify system log operation, the system automatically allocates the system log data set during IPL as a class A SYSOUT data set. Subsequently, the log keeps track of the number of entries it receives by counting the WTL macro instructions executed against it. After 500 WTLs, the system opens and allocates a new system log and closes and dynamically unallocates the currently full log.

Modifying the System Log

The system programmer can alter the default operation of the system log to control the processing associated with the log data sets. He can change the SYSOUT class of the log data sets and the number of WTLs received before switching log data sets.

The processing of the log data sets can be controlled from the operator console or from a SYS1.PARMLIB member name IEASYSxx, where xx is a unique number (chosen by the installation) that identifies the member. This member must be included in the system during IPL, in response to the request to specify the system parameters.

From the console, the operator can control the processing with commands. (For further information about the operator commands, see *Operator's Library: OS/VS2 MVS System Commands*. For example, the operator can issue a WRITELOG command with the START operand after a system failure or after a WRITELOG command with the CLOSE operand.

The following SYS1.PARMLIB parameters initialize or alter the system log control values:

- LOGLMT which controls the number of WTLs received before the system switches data sets
- LOGCLS which controls the SYSOUT class of the system log data set

The LOGLMT value must be a six-digit number in the range 000001-999999. An all-zero entry value results in the system default of 500. When choosing the LOGLMT value, the system programmer should consider:

- Whether the system log is defined as MCS hardcopy
- Whether the system log data is sufficiently critical to the system to require frequent allocating, switching, and queuing to a SYSOUT class

The LOGCLS value must be one alphameric character. The default value is class A.

The following example shows the correct format for including the LOGLMT and LOGCLS parameters in the IEASYSxx member of SYS1.PARMLIB when specifying the system parameters during IPL:

```
LOGLMT=004852,LOGCLS=L
```

The preceding example would cause the system log task to switch data sets after 4852 WTLs, and the job entry subsystem to queue the current data set to class L for SYSOUT processing.

Chapter 8: Updating the Master Job Control Language Data Set

The master job control language data set (CSECT name and load module name are MSTRJCL) is a nonexecutable module that is created during system generation and resides on SYS1.LINKLIB. As provided by IBM, MSTRJCL contains data definitions for all system input and output data set necessary for communications with the job entry subsystem. MSTRJCL also contains the START command that starts the job entry subsystem at initialization. Figure 19 shows MSTRJCL as it exists before it is assembled. During system generation, &SSNAME is replaced with the name of the job entry subsystem.

```
DC      CL80'//MSTRJCL          JOB MSGLEVEL=(0,0)'
DC      CL80'//                 EXEC PGM=IEEMB860,DPRTY=(15,15)'
DC      CL80'//STCINRDR         DD SYSOUT=(A,INTRDR)'
DC      CL80'//TSOINRDR         DD SYSOUT=(A,INTRDR)'
DC      CL80'//IEFPDSI          DD DSN=SYS1.PROCLIB,DISP=SHR'
DC      CL80'//IEFPARM          DD DSN=SYS1.PARMLIB,DISP=SHR'
DC      CL80'//SYSUADS          DD DSN=SYS1.UADS,DISP=SHR'
DC      CL80'//SYSLBC           DD DSN=SYS1.BRODCAST,DISP=SHR'
DC      CL80'//SMFMANX          DD DSN=SYS1.MANX,DISP=SHR'
DC      CL80'//SMFMANY          DD DSN=SYS1.MANY,DISP=SHR'
DC      CL80'// START &SSNAME'
DC      CL80'/*'
```

Figure 19. MSTRJCL Data Set

If an installation does not plan to use TSO, the system programmer can delete the TSOINRDR, SYSUADS, and SYSLBC data definitions. He can add other data definitions as necessary, provided that the data sets they define are created before the IPL that is to make use of them. If the allocation of any data set defined in MSTRJCL fails, the IPL also fails.

Changes to MSTRJCL fall into two categories: modifying an existing statement, and adding or deleting a statement. To modify a particular statement, the system programmer can use the AMASPZAP service aid program. To delete an existing statement or add a new one, he must reassemble the MSTRJCL statements, including a MSTRJCL CSECT card and an END card with the statements shown in Figure 19. In the figure, &SSNAME should be replaced with the name of the job entry subsystem.

By deleting the statement that contains the START command for the job entry subsystem, the system programmer enables the console operator to specify the job entry subsystem during system initialization.

Note: Until the primary job entry subsystem is started, no work can be done that requires any input or output services.

Chapter 9: Updating the Subsystem Names Table – IEFJSSNT

The subsystem names table (CSECT name and load module name is IEFJSSNT) is a non-executable module that resides in either SYS1.LINKLIB or a library that is concatenated to SYS1.LINKLIB via a LNKLSTxx member of SYS1.PARMLIB. The subsystem names table defines secondary subsystems that were not defined on the SCHEDULR macro during system generation and, optionally, the names of corresponding subsystem initialization routines. MVS processes the table during master scheduler initialization, which allows subsystem initialization routines to execute as part of the master scheduler initialization process.

The format of a table entry is shown in Figure 20. As provided by IBM, IEFJSSNT contains five null entries.

Name	Entry Name	Unused
1 4	5 12	13 80

Where name is one of the following:

- A one-to-four-character alphameric subsystem name. If the name contains fewer than four characters, it must be left-justified and padded with blanks.
- A null entry specified by making the first character of the name field a blank.
- The end-of-table indicator, specified by a value of X'FFFFFFFF'.

Where entry name is one of the following:

- A one-to-eight-character name corresponding to the entry point of the initialization routine for the associated subsystem. If the entry name contains fewer than eight characters, it must be left-justified and padded with blanks. The entry name must be a member name or an alias in either SYS1.LINKLIB or a library concatenated to SYS1.LINKLIB via a LNKLSTxx member of SYS1.PARMLIB.
- A null entry, specified by making the first character of the entry name field a blank.

Notes:

1. MVS ignores bytes 13 through 80 of a table entry. You may use this area for comments.
2. A null name field reserves space in the table. If a name field is null, MVS does not build a subsystem communication vector table (SSCVT), and it does not invoke a subsystem initialization routine.
3. If a table entry has a name field that is not null and an entry name field that is null, MVS builds an SSCVT, but it does not invoke the subsystem initialization routine.
4. Do not specify duplicate subsystem names.

Figure 20. Entry Format for the Subsystem Names Table

Changes to IEFJSSNT fall into two categories: modifying an existing entry; adding or deleting entries. To modify an existing entry, use the AMASPZAP service aid program. To add or delete an entry, reassemble the table. The assembly input must include an IEFJSSNT CSECT card, the desired table entries prepared according to the format described in Figure 20 (the last table entry must always be the end-of-table indicator), and an END card.

If you create several null entries when you initially assemble the table, you can later define a new subsystem to MVS by using AMASPZAP to modify one of these entries; this eliminates the necessity of reassembling the table each time you define a new subsystem. To create a null table entry, make the first character of the name field a blank. When MVS processes the subsystem names table, it ignores null entries.

See Figure 21 for an example of the assembly input.

```
IEFJSSNT CSECT
*                                                                       *
* This entry defines a subsystem named SUB1 and its entry               *
* point EP1                                                             *
        DC      CL4'SUB1'   SUBSYSTEM NAME FIELD
        DC      CL8'EP1'    ENTRY POINT NAME FIELD
        DC      CL68'USER COMMENT'  COMMENT FIELD
*                                                                       *
* This entry defines a null entry. When MVS processes the               *
* table, it will ignore this entry                                      *
*                                                                       *
        DC      CL4' '      SUBSYSTEM NAME FIELD
        DC      CL8' '      ENTRY POINT NAME FIELD
        DC      CL68'NULL ENTRY'    COMMENT FIELD
*                                                                       *
* This entry defines a subsystem named SUB2 and a null entry            *
* point. When MVS processes the table, it will build an                 *
* SSCVT for this entry but it will not invoke an initialization         *
* routine.                                                              *
*                                                                       *
        DC      CL4'SUB2'   SUBSYSTEM NAME FIELD
        DC      CL8'        ENTRY POINT NAME FIELD
        DC      CL68'NULL ENTRY POINT'  COMMENTS FIELD
*                                                                       *
* This entry defines the end-of-table indicator                         *
*                                                                       *
        DC      XL4'FFFFFFFF'   SUBSYSTEM NAME FIELD
        DC      CL8' '          ENTRY POINT NAME FIELD
        DC      CL68'END OF TABLE'  COMMENTS FIELD
        END     IEFJSSNT
```

Figure 21. Sample Input for Reassembling IEFJSSNT

Chapter 10: External Writers

After output is queued by the job entry subsystem the output can be written by the writer associated with the job entry subsystem or by an external writer. An external writer is a non-JES writer that calls an IBM-supplied output writer routine called STDWTR or an installation-defined writer routine named on the SYSOUT DD statement. The operator starts an external writer in a private address space, and the data is written using the QSAM access method.

With an external writer, SYSOUT data sets can be written to devices other than local and remote printers and punches supported by the job entry subsystem. Installation-written writers can effect special processing of SYSOUT data sets accompanied by special separator pages formatted by an installation routine. The user requests an installation-written writer by specifying the name of the writer (other than INTRDR or STDWTR which are reserved) on the SYSOUT parameter of a DD statement.

There are three rules for selecting data sets to be processed by for an external writer:

- Data sets with data characteristics that include an installation-written writer name are not eligible for selection by a job entry subsystem printer or punch. An external writer should be started to the group's output class to process these data sets, or they should be dequeued specifically by the operator, who can cause the external writer to dequeue data sets that specify a specific installation writer name regardless of the output class. The JES2 $DF or JES3 *I,U command can be used to determine the classes in which data sets specifying installation-written writers are queued.
- Data sets that do not specify a special writer can be written by a job entry subsystem writer or an external writer, depending on operator action. If the same class is specified for both a job entry subsystem writer and an external writer, the two routines compete for data sets on a first-come, first-served basis.
- Print train, carriage, forms flash, paper bursting, and, optionally, forms specification are ignored when an external writer selects data set groups. As with the MVT and VS2 Release 1 output writer, FCB and UCS specification must be controlled through separating the data by class.

When there are no more data sets to select, the external writer notifies the operator by issuing a message, which also informs the operator of the writer's current setup.

An external writer can dequeue data sets by any or all of the following characteristics: output class, job (job ID), forms, destination of LOCAL or remote workstation name, and installation-written writer name. Selection by local device name is not available in JES. A characteristic, if not specified, is ignored when selecting data sets. For example, the operator can set up an external writer to dequeue data sets for a certain installation-written writer name and a specific set of forms. Since job, class, and destination are not specified, any data set specifying the forms and writer is eligible regardless of its class or destination.

For compatibility with the MVT and VS2 Release 1 output writer, the external writer dequeues data sets by class and LOCAL destination, if one or more output classes are specified on either the START command or in the cataloged procedure used to call this external writer. Forms are mounted and installation-written writers are called on demand.

The IBM-supplied name STDWTR can be used by the operator for selecting data sets that have not specified any writer. Similarly, the forms name STD causes the external writer to select only those data sets that have not specified a particular form.

For individual SYSOUT data sets, 3800 printer parameters such as FCB, CHARS, and OPTCD=J are ignored. For further information on using the 3800 Printing Subsystem, refer to *IBM 3800 Printing Subsystem Programmer's Guide*.

The following sections describe operator commands that control external writers, the external catalog procedure, installation-defined output writer routines, and output separation programs.

Operator Commands to Control External Writer Processing

The operator is able to start, modify, cancel and stop an external writer with the following commands:

- START or S – start the external writer
- MODIFY or F – change the options of the external writer
- CANCEL or C – cancel a data set(s) being processed by an external writer
- STOP or P – stop the external writer

For additional information on the preceding commands, see *Operator's Library: OS/VS2 MVS System Commands*.

The External Writer Cataloged Procedure

A cataloged procedure for external writers requires two job control statements:

- An EXEC statement named IEFPROC – which specifies the external writer program
- A DD statement named IEFRDER – which defines the output data set

The standard external writer procedure supplied by IBM is named XWTR. The XWTR procedure is:

```
//IEFPROC   EXEC    PGM=IASXWR00,REGION=20K,
//                  PARM='PA'
//IEFRDER   DD      UNIT=2400,VOLUME=(,,,35),
//                  DSNAME=SYSOUT,DISP=(NEW,KEEP),
//                  DCB=(BLKSIZE=133,LRECL=133,BUFL=133,
//                  BUFNO=2,RECFM=FM)
```

When creating an external writer procedure, the procedure format and the statement requirements must be maintained. The IBM-supplied procedure can be used as an example. The statements are explained individually in the following topics.

EXEC Statement

The EXEC statement specifies the output writer program and if ADDRSPC=REAL is requested, its region size. It also passes a set of parameters to the output writer program. The format for the EXEC statement is:

```
//IEFPROC   EXEC    PGM=IASXWR00[,REGION=nnnnnK,ADDRSPC=REAL]
//                  [,PARM=-cxxxxxxxx,seprname-]
```

The step name must be IEFPROC, as shown. The parameter requirements are as follows:

PGM=IASXWR00
specifies the output writer program. The name of the program must be IASXWR00, as shown.

REGION = nnnnnK should be specified only for ADDRSPC=REAL
specifies the region size for the output writer. The value nnnnn represents a number from one to five digits that is multiplied by K (K=1024 bytes) to designate the region size. The region requirements depends on the size of the buffers and the output writer used. An insufficient size specification will result in an abnormal termination.

ADDRSPC=REAL
specifies that the external writer program cannot be paged.

PARM="cxxxxxxxx,seprname"
is a set of parameters for the output writer program. The first part of this parameter field can contain from one to nine characters. The second part of this parameter field, if specified, is separated from the first part by a comma and contains a program name from one to eight characters. Both parts of this parameter field are explained below.

c
an alphabetic character, either P (for printer) or C (for punch card), that specifies the control characters for the class of output to be processed by the writer.

xxxxxxxx
from one to eight (no padding required) single-character class names for system output. These characters specify the classes of output that the writer can process and also establish the priority of the output classes, with the highest priority on the left. If class name parameters are included in the START command, they override this entire set of class names in the cataloged procedure. If no classes are specified on the EXEC statement and none are specified in the START command, the external writer waits for the operator to enter a MODIFY command before processing any output.

seprname
the name of the program (up to eight characters) that provides job separation in the output data set. The named program must reside in the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB). The name IEFSD094 specifies the output separator program supplied by IBM, or the name of a user-written program can specified. This subparameter may be be omitted, in which case no output separator is used.

DD Statement

The procedure for the output writer must include a DD statement that defines the output data set. The format for this statement is:

```
//IEFRDER   DD      UNIT=device,LABEL=(,type),                    X
//                  VOLUME=(,,,volcount),                         X
//                  DSNAME=anyname,DISP=(NEW,KEEP),               X
//                  DCB=(list of attributes),                     X
//                  UCS=(code[,FOLD][,VERIFY]),                   X
//                  FCB=(image-id  {,ALIGN }  )
                                   {,VERIFY}
```

The ddname should be IEFRDER and must be the first DD statement in the procedure IEFPROC. However, the external writer will always treat the first DD statement as an output data set regardless of the ddname specified. The parameter requirements are as follows:

UNIT = device
specifies the printer, magnetic tape, card punch, or direct access device on which the output data set will be written.

LABEL = (,type)
describes the data set label (needed only for tape data sets). If this parameter is omitted, a standard label is assumed.

VOLUME = (,,,volcount)
limits the number of tape volumes that can be used by this writer during its entire operation (from the time it is started to the time it is stopped). This parameter is not required for printers, card punches or DASD.

DSNAME = anyname
specifies a name for the output data set, so that the output data set can be referred to by subsequent job steps. This name is necessary for specification of the KEEP subparameter in the DISP field.

DISP=(NEW,KEEP)
specifies the KEEP subparameter to prevent deletion of the output data set (tape and direct access only) at the conclusion of the job step.

DCB = (list of attributes)
specifies the characteristics of the output data set and the buffers. The BLKSIZE and LRECL subparameter fields must be specified in all cases. The BUFL subparameter field, if not specified, is calculated on the basis of the BLKSIZE value. Other subparameter fields may be specified as needed; if they are not, they will assume the QSAM default attributes, which follow:

BUFNO — three buffers for the 2540 device, two buffers for all other devices.
RECFM — U-format, with no control characters.
TRTCH — odd parity, no data conversion, and no translation.
DEN — lowest density.
OPTCD — printer data checks are suppressed and "select translate table" characters are printed as data. (The external writer does not support OPTCD=J, a 3800 Printing Subsystem specification.)

UCS = (code[,FOLD][,VERIFY])
specifies the code for a universal character set (UCS) image that will be loaded into the UCS buffer. FOLD causes bits 0 and 1 to be ignored when comparing characters between the UCS buffer and the print line buffer. This option allows lowercase alphabetic characters to be printed in uppercase by an uppercase print chain or train. VERIFY causes the specified UCS image to be printed for verification by the operator. The UCS parameter is optional and is valid only when the the output device is a 1403 or 3211.

FCB = (image-id $\left\{\begin{matrix}\text{,ALIGN}\\\text{,VERIFY}\end{matrix}\right\}$)
causes the forms control buffer (FCB) image with the specified image-id to be loaded into the FCB. One of two optional parameters, ALIGN or VERIFY, can be coded. Either parameter allows the operator to align forms. In addition, VERIFY causes the specified FCB image to be printed for visual verification. The FCB parameter is valid when the output device is a 3211 or 3800; it is ignored when the device is not one of these.

Output jobs that require special print chains, specific classes should be assigned for each different chain. The desired chain can be specified in the writer procedure; the chain will be loaded automatically when that writer is started. (Printers used with special chains should be specified with esoteric device specified as defined during system generation.)

The following sequence is an example of a writer-cataloged procedure for the P11 chain.

```
//IEFPROC   EXEC    PGM=IASXWR00,                                  X
//                  PARM='PDEG,IEFSD094'
//IEFRDER   DD      UNIT=SYSPR,DSNAME=SYSOUT,FCB=(STD2,ALIGN),     X
//                  UCS=P11,                                       X
//                  DISP=(,KEEP),DCB=(BLKSIZE=133,BUFL=133,        X
//                  LRECL=133,BUFNO=2,RECFM=FM)
```

If the output device is a 3211, a UCS or FCB image can be loaded dynamically between the printing of data sets. Therefore, a mixture of data sets using different images in a single output class is allowed; however, this might require mounting trains and changing forms, and might not be desirable. When the output device is a 1403 or 3800, the UCS image or 3800 attributes are specified at START XWTR time and cannot be changed until the writer is stopped; all data sets within an output class must be printed using the same train. This parameter cannot be overridden for a specific data set when using the (asynchronous) SYSOUT writer. The FCB image is ignored when the 1403 is specified.

External writer output to an IBM 3800 Printing Subsystem can also make use of the CHARS, COPIES, FLASH, and MODIFY JCL parameters. For information about how to use these parameters, refer to *IBM 3800 Printing Subsystem Programmer's Guide*. For further information about coding rules, defaults, and examples, refer to *OS/VS2 MVS JCL*.

Writing an Output Writer Routine

You can create your own installation-defined external writer subroutine to support devices not supported by the JES writers. The output data set writer routine used for a data set can be specified by name (other than INTRDR or STDWTR) in a DD statement. If it is, the data set must be processed by an external writer. If a data set that does not specify an installation-written writer or output writer is processed by the external writer, a standard IBM-supplied writer routine is used. The standard routine transcribes the data set to the specified output device, making only those data format and control character transformations required to conform to the attributes specified for the output data set.

The following material describes how to write an output data set writer routine.

Characteristics of the Standard External Routine

Before writing or modifying an output writer routine, the functions performed by the standard data set writer should be understood. In general, these functions include opening the data set (referred to as an input data set) that contains the information to be processed, obtaining the records of the data set, making any necessary transformations in record format or control character attributes, and placing these (possibly transformed) records in the output data set, which appears on a specified output device. The standard writer also must close the input data set and restore system conditions to the state they were in before the writer routine was invoked. The output writer cannot be named STDWTR or INTRDR.

The Output Writer Routine

To use the output writer routine, the name of the routine must be specified as a parameter in the SYSOUT operand of a DD statement. The routine must be in the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB).

For VS2 you should:

- Use standard linkage conventions for attaching the output writer routine.
- Create the output writer routine in reentrant mode.

- Attach the output writer routine via the ATTACH macro instruction when your data set requires processing.
- Acquire necessary storage via the GETMAIN macro instruction.
- Release storage via the FREEMAIN macro instruction.
- Return control to the standard writer, at the end of processing, via the RETURN macro instruction.

Parameter List

After job management routines perform initialization requirements and open the output data set into which the writer routine places records, control is given to the routine via the ATTACH macro instruction. At this time, general registers 1 and 13 contain information that the program requires. Register 1 contains the storage address of a 12-byte parameter list. The information in this parameter list follows:

		Output Device
Byte 0	Bits 0-2	Bits 0 thru 2 describe the type of device:
011.....		2520 or 2540 punch unit
001.....		1403, 1443, 3211, or 3800 printer device
010.....		tape device with punch-destined output
000.....		tape device with printer-destined output
	Bit 2	If this bit is on, the output unit is either a printer or a punch.
	Bits 3-7	No significant information.
Bytes 1-3	Not used, but must be present	
Bytes 4-7	This word contains the address of the data control block (DCB) for the opened output data set to be referred to by the writer.	
Bytes 8-11	This word contains the DCB address for the input data set from which the writer will obtain logical records. (At the time this 12-byte parameter list is given to the writer, the input data set is not open.)	

The switches indicated by the three high-order bit settings in byte 0 should be used to translate control character information from the input data set records to the form required by the output data set records.

The writer should save and restore registers.

Programming Conventions

An output writer routine must issue an OPEN macro instruction to open the desired input data set residing on a direct access device as a result of the previous execution of a processing program. (Note: The output data set used by a writer is opened by a job management routine before control is given to the writer. This output data set must be given records by a PUT macro instruction operating in the "locate" mode.)

If the processing program that produces a given data set (to be used as an input data set by a writer) did not open the data set, the data set contains no records, and the DCBBLKSI and DCBBUFL fields of the input DCB contain zero. The DCBBLKSI field may also be zero even if the data set does contain records – if the processing program did not put the block size value for the input data set in the DCB. If both these DCB fields are zero, the writer routine should insert a value (the standard writer inserts the decimal value of 18) in the DCBBLKSI field to permit the open routine to continue. The standard writer does this via a routine pointed to by an entry in the EXLIST parameter of the DCB. Since there is no data set, nothing is put on the output device. The output data set writer provides a SYNAD routine to process errors associated with the output as well as the input data set.

The standard data set writer also includes accounting support for the SMF output writer record (record type 6).

Before the OPEN macro instruction is issued, the DCBD macro instruction can be used to symbolically define the fields of the DCB, and the EXLIST and/or SYNAD routine addresses can be inserted. Other than SYNAD, no modifications can be made to the output DCB.

After the routine finishes writing the output data set, it must close the input data set and return using the RETURN macro instruction. A return code must be placed in register 15. This code should indicate that an unrecoverable output error has occurred (code of 8) or has not occurred (code of 0).

Note: The programming support for the 3525 Interpret Punch includes an INTERPRET PUNCH feature that is supported by BSAM and QSAM. The support for this feature includes the punching and printing of graphically printable punched characters on print lines one and three of the card. Line one includes the first 64 characters and line three includes the last 16 characters (right-justified). Extraneous characters are printed for non-graphic eight-bit codes.

If the INTERPRET PUNCH function is designated via the FUNC parameter in either a DCB or DD statement, an existing output data set will be interpreted as well as punched. The output must be 80 bytes, or 81 bytes if first-character control is being used.

Processing Performed by the Output Writer

Figure 22 provides a general description of the procedures followed by the standard writer. When writing a writer routine, items can be deleted, modified, or added to some of these procedures, depending on the characteristics of the data set(s). However, the procedures must be consistent with operating system conventions.

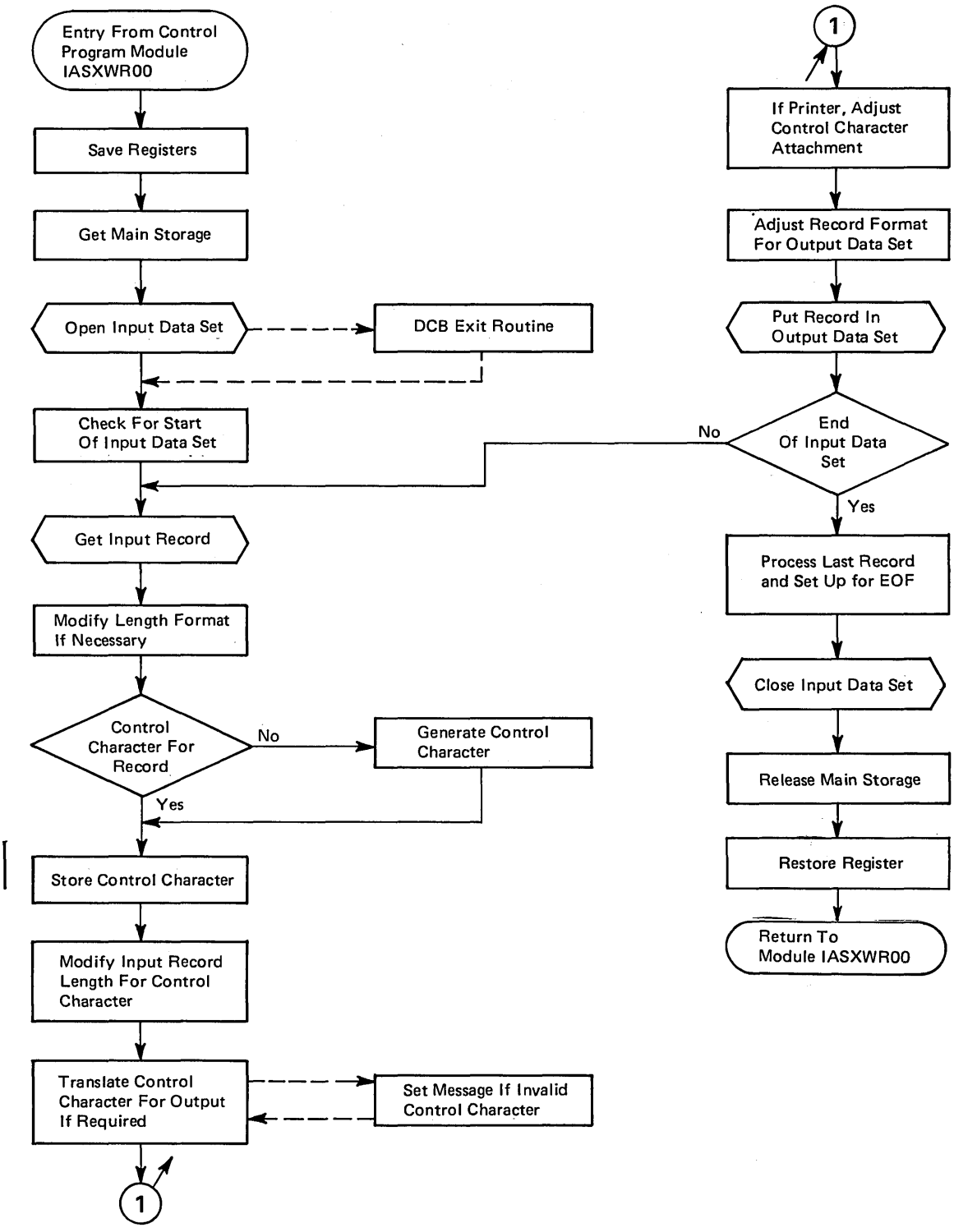


Figure 22. General Logic of Standard External Writer Routine

Saving Register Contents: Upon entering the writer program, the program must save the contents of the general registers, as previously described.

Obtaining Main Storage for Work Areas: The standard writer obtains storage for a work area by means of the GETMAIN macro instruction. In this work area, the standard writer establishes switches and saves record lengths and control characters.

Processing Input Data Set(s): To process a data set, the writer must obtain each record individually from the input data set, transform (if necessary) the record format and the control characters associated with the record in accordance with the output data set requirements, and put the record in the output data set. Data set processing by the standard writer can be considered in three aspects.

1. The first consideration is what must be done before actually obtaining records from an input data set. For example, the output device is a printer, a SETPRT macro instruction can be issued to initialize it. Provision must also be made to handle the two forms of record control character that may accompany a record in an output data set. The printer is designed so that, if the output data set records contain machine control characters, a record (line) is printed before the effect of its control character is considered. However, if ANSI control characters are used in the output data set records, the control character effect is considered before the printer prints a record.

   Thus, if all the input data sets do not have the same type of control characters, it might be desirable to avoid overprinting the last line of one data set with the first line of the following data set. If the records of the input data set have machine control characters (mcc) and the output data set records are to have ANSI control characters (acc), the standard writer produces a control character that indicates one line should be skipped before printing the first line of output data.

   If the input data set records have acc and the output data set records are to be written with mcc, the standard writer prints a line of blanks before printing the first actual output data set record. Following this line of blanks, a one-line space is generated before the first output record is printed. The preceding "printer initialization" procedure (or a similar one based on the characteristics of the data sets) is recommended.

2. After an input data set is properly opened and any necessary printer initialization (such as issuing a SETPRT macro instruction) is completed, the writer obtains records from the input data set. The standard writer uses the locate mode of the GET macro instruction. As each record is obtained, its format and control character must be adjusted, if necessary, to agree with that required for output.

   Since the output data set is previously opened by another routine (job management), a writer routine must adhere to the established conventions. The data set is opened to receive records from the PUT macro instruction operating in the locate mode. For fixed-length record output, the length of the records in the output data set is obtained from the DCBLRECL field of the DCB. If an input record length is greater than the length specified for the records of the output data set, the standard writer truncates the necessary right-hand bytes of the input record. If the input record length is smaller than the output record length, the standard writer left-justifies the input record and adds blanks on the right end to give the correct length.

   When the output record format is variable and the input record format is not variable, the standard writer constructs each output record by adding control character information (if necessary) and variable record control information to the output record. The record control information is four bytes long and the control character information is one byte long. Both additions are made to the left end of the record. If the output record is not at least 18 bytes long, it is further modified by adding bytes (blanks) to the right end of the record. If the output record length does not agree with the length of the output buffer, the standard writer makes the proper adjustment.

3. The third aspect is an end-of-input-data-set routine. The standard writer handles output to either a card punch unit or a printer unit, as required. Output to an intermediate device such as a tape unit is considered in light of the ultimate destination (for example, punch or printer). If proper consideration is not given, all records from a given data set may not be available on the output device until the output of records from the next data set is started or until the output data set is closed. When the output data set is closed, the standard writer automatically puts out the last record of its last input data set.

Punch Output: Normally, when the standard writer is using a card punch as the output device, the last three output records are not in the collection pockets of the punch when the input data set is closed. To put out these three records with the rest of the data set and with no intervening pauses, the writer provides for three blank records following the actual data set records.

Printer Output: When the standard writer uses a printer as an output device, the last record of the input data set is not normally put in the output data set when the input data set is closed. To force out this last record, the writer generates a blank record that follows the last record of the actual data set.

The problem of overprinting the last line of one data set by the first line of the following data set must also be considered. Depending on the combination of input record control character and required output record control character, a line of blanks and a spacing control character may be used either individually or in combination to preclude overprinting. (***Note:*** If overprinting is desired for some reason, control characters in the data set records themselves may be used to override the effect (but not the action) of the previously described solutions to overprinting.)

Closing Input Data Set(s): After the standard writer finishes putting out the records of an input data set, it closes the data set before returning control to the system output writer. All input data sets must be closed.

Releasing Main Storage: The storage and buffer areas obtained for the writer must be released to the system before the writer relinquishes control. The FREEMAIN macro instruction should be used for this.

Restoring Register Contents: The original contents of general registers 0 through 12 and 14 must be restored. The RETURN macro instruction is used for this. To inform the operating system of the results of the processing done by the writer, a return code is placed in general register 15 before control is returned. If the writer routine terminates because of an unrecoverable error on the output data set, the return code is 8; otherwise, the return code is 0. Unrecoverable input errors must be handled by the data set writer.

Output Separation

The external writer can be used by a problem program to channel its output to a printer or punch. When this is done; however, the output stream goes uninterruptedly from one job to another, making it difficult to separate the output of one job from that of another unless output separation is provided.

The output separator facility of the system provides a means of identifying and separating the output of various jobs processed by the same output unit. To do this, the separator writes separation records to the system output data set between the writing of each section of a job's output.

For data processed by the external writer, the IBM output separator or the user's own output separator can be used.

The external writer standard separator function operates under control of the external writer. The external writer separator program must reside in the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB). Its name, IEFSD094, must be included as a parameter in the output writer procedure (the second part of the PARM field in the EXEC statement). (The cataloged procedure for the writer is described in the beginning of this section.) The type of separation provided by the separator depends on whether the output is punch-destined or printer-destined.

Punch-Destined Output: The external writer provides three specially punched cards (deposited in stacker 1) prior to the punch card output of each job. Each of these separator cards is punched in the following format:

Columns	1 to 35	blanks
Columns	36 to 43	jobname
Columns	44 to 45	blanks
Column	46	output classname
Columns	47 to 80	blanks

Printer-Destined Output: The external writer provides three specially printed pages prior to printing the output of each job. Each of these three separator pages is printed in the following format:

- Beginning at the channel 1 location (normally near the top of the page), the jobname is printed in block character format over 12 consecutive lines. The first block character of the 8-character jobname begins in column 11. Each block character is separated by 2 blank columns.
- The next 2 lines are blank.
- The output classname is printed in block character format covering the next 12 lines. This is a 1-character name, and the block character begins in column 55.
- The remaining lines to the bottom of the page are blank.

In addition to the above, a full line of asterisks (*) is printed twice (overprinted) across the folds of the paper. These lines are printed on the fold preceding each of the three separator pages, and on the fold following the third page. This feature provides easy separation of job output in a stack of printed pages.

For *printer-destined output with the IBM-supplied separator, a channel 9 punch should be included in addition to the channel 1 punch on the carriage control tape or in the forms control buffer* (FCB). The channel 9 punch controls the location of the line of asterisks and should correspond to the bottom of the page. To print the line of asterisks on the fold of the pages, the printer registration should be offset.

Because the IBM-supplied separator routine makes no provision for a 3800 Printing Subsystem, the FCB must locate a channel 9 punch at least one-half inch from the paper perforation.

Writing an Output Separator Program

You can write your own separator program for your installation. However, the program should conform to the requirements explained in the following topics. The separator program, when added to the link library (SYS1.LINKLIB) or the LPA library (SYS1.LPALIB), is invoked by specifying its name as a parameter in the EXEC statement of the output writer cataloged procedure.

Parameter List

The external writer provides the separator program with a 4-word parameter list of needed information. When the program receives control, register 1 contains the address of a 4-word parameter list, which contains the following:

	Byte 0	This word contains switches that indicate the type of output unit, as follows:
011.....		2520 or 2540 punch device
001.....		1403, 1443, 3211, or 3800 printer device
010.....		tape device with punch-destined output
000.....		tape device with printer-destined output
Bytes 1-3	Reserved	
Bytes 4-7	This word is the address of the output DCB (data control block).	
Bytes 8-11	This word is the address of an 8-character field containing the jobname.	
Bytes 12-15	This word is the address of a 1-character field containing the output classname.	

The DCB pointed to by the parameter list is established for the queued sequential access method (QSAM) and is already open when the separator program receives control.

The address of the jobname and the address of the output classname are provided in the parameter list so that this information may be used in the separation records written by the separator program.

Programming Conventions

The separator program, if specified in the external writer cataloged procedure, is brought into storage by a LINK macro instruction issued by the external writer. The separator program can be any size, but a program over 8K might affect the region requirement of the external writer.

Caution: Since the separator program operates with a privileged protection key, but in problem program mode, the separator program must insure data protection during its execution.

When writing a separator program, the following programming conventions must be observed:

- The program must conform to standard linkage conventions.
- The program must use the PUT macro instruction in locate mode to write separation records on the output data set. (This method is required by the QSAM DCB that is open for the output data set.)
- The program must establish its own synchronous error exit routine. The address of this routine must be placed into the DCBSYNAD field of the output DCB. This gives control to the error exit routine in case an uncorrectable I/O error occurs while writing the program's output.
- The program should use the RETURN macro instruction to return control to the external writer. Before returning, the program must free any main storage it obtained during its operation and must place a return code (binary) in register 15. The return codes signify:

  0 – Successful operation

  8 – Unrecoverable output error (should be set if the error exit routine is entered)

Output from the Separator Program

The separator program can write any kind of separation identification. The jobname and the output classname for each job are available through the parameter list for inclusion in the output, if desired. An IBM-supplied routine can be used that constructs block characters (explained below). As many separator cards can be punched or as many separator pages can be printed as necessary.

The output from the separator program must conform to the attributes of the output data set. These attributes, which can be determined from the open output DCB pointed to by the parameter list, are:

- Record format (fixed, variable, or undefined length)
- Record length
- Type of carriage control characters (machine, ANSI, or none)

For printer-destined output, the separation records must be begin on the same page as the previous job output, or, if there is no previous output, the next page available. However, the separator program should skip at least one line before writing any records, because in some cases the printer is still positioned on the line last printed.

After completing the output of the separation records, the separator program should write sufficient blank records to force out the last separation record. This also allows the error exit routine to obtain control if an uncorrectable output error occurs while writing the last record. The requirements are:

- One blank record for printer-destined output
- Three blank records for punch-destined output

Using the Block Character Routine

For printer-destined output, the separator program can use an IBM-supplied routine to construct separation records in a block character format. This routine is a reenterable module named IEFSD095 and resides in the module library SYS1.AOSB0.

The block character routine constructs block letters (A to Z), block numbers (0 to 9), and a blank. The program furnishes the desired character string and the construction area. The block characters are constructed one line position at a time. Each complete character is contained in 12 lines and 12 columns; therefore, a block character area consists of 144 print positions. For each position, the routine provides either a space or the character itself.

The routine spaces 2 columns between each block character in the string. However, the routine does not enter blanks between or within the block characters. The program must prepage the construction area with blanks or other desired background before entering the block character routine.

To use the IBM-supplied block character routine, the separator program executes the CALL macro instruction with the entry point name of IEFSD095. Since the block characters are constructed one line position at a time, complete construction of a block character string requires 12 entries to the routine. Each time, the address of a 4-word parameter list should be provided in register 1. The parameter list must contain the following:

- Bytes 0-3 — This word is the address of a field containing the desired character string in EBCDIC format.
- Bytes 4-7 — This word is the address of a full word field containing the line count as a binary integer from 1 to 12. This represents the line position to be constructed on this call.
- Bytes 8-11 — This word is the address of a construction area in main storage where the routine will construct a line of the block character string. The required length in bytes of this construction area is 14n-2, where n represents the number of characters in the string.
- Bytes 12-15 — This word is the address of a fullword field containing, in binary, the number of characters in the string.

Index

OS/VS2 MVS System Programming Library:
Job Management
GC28-0627-2

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If comments apply to a Selectable Unit, please provide the name of the Selectable Unit _____.

If you wish a reply, give your name and mailing address:

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

Please circle the description that most closely describes your occupation.

Customer	(Q) Install Mgr.	(U) System Consult.	(X) System Analyst	(Y) System Prog.	(Z) Applica. Prog.	(F) System Oper.	(I) I/O Oper.	(L) Term. Oper.					(O) Other
IBM	(S) System Eng.	(P) Prog. Sys. Rep.	(A) System Analyst	(B) System Prog.	(C) Applica. Prog.	(D) Dev. Prog.	(R) Comp. Prog.	(G) System Oper.	(J) I/O Oper.	(E) Ed. Dev. Rep.	(N) Cust. Eng.	(T) Tech. Staff Rep.	

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC28-0627-2

Reader's Comment Form

Cut or Fold Along Line

OS/VS2 MVS System Programming Library: Job Management (S370-36) Printed in U.S.A. GC28-0627-2

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601