IBM Data Language/I Disk Operating System/
Virtual Storage (DL/I DOS/VS)

Recovery/Restart Guide

Version 1 Release 7

SH24-5030-00

IBM Data Language/I Disk Operating System/
Virtual Storage (DL/I DOS/VS)

Recovery/Restart Guide

Version 1 Release 7

SH24-5030-00

Note !

Before using this information and the product it supports, be sure to read the general information under “Notices” on page vii.

First Edition (December 2002)

This edition applies to Version 1 Release 7 of IBM Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), Program Number 5746-XX1, and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Deutschland Informationssysteme GmbH
Department 0215
Pascal Str. 100
70569 Stuttgart
Germany
```

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks and Service Marks

The following terms are trademarks of International Business Machines Corporation in the Unites States, or other countries, or both:

```
CICS
IBM
```

Preface

Purpose

This manual helps you to plan and develop a recovery and restart procedure in the event your DL/I DOS/VS system fails. It also discusses the various facilities available for recovering and restarting your system after a failure, including a description of the steps that can be be used for a normal recovery procedure. Since the DL/I utilities are used to recover and restart your system, this manual also provides a summary of the job control statements necessary to run them.

Audience

This manual is intended for persons responsible for developing a recovery and restart procedure, recovering DL/I data bases and restarting DL/I in the event of a failure.

Contents

The chapters within this guide should be read sequentially for the first time to familiarize you with the recovery and restart process. Once you are familiar with the contents, you may use this book as a reference document. The information contained in this manual includes:

- Chapter One introduces you to recovery/restart, discussing its necessity and concepts.
- Chapter Two helps you to decide the kind of recovery/restart capability you believe you should have in your installation and those things necessary to consider when developing a recovery/restart plan.
- Chapter Three introduces you to the various facilities available for you to plan and carry out a recovery/restart process.
- Chapter Four provides recovery guide information, including the steps that can be used in a normal recovery process. Also discussed in this chapter are the possible actions you can take if a failure occurs during the recovery process itself.
- Chapter Five provides a summary of the job control statements necessary to run the DL/I utilities you may use in your recovery/restart process.

A Glossary of DL/I terms is provided at the back of this manual for your convenience.

Related Publications:

DL/I VSE Release Guide, SC33-6211-05

DL/I DOS/VS Release Guide, SC33-6211-04

DL/I DOS/VS Data Base Administration, SH24-5011

DL/I DOS/VS Resource Definition and Utilities, SH24-5021

DL/I DOS/VS Interactive Resource Definition and Utilities, SH24-5029.

DL/I DOS/VS Recovery/Restart Guide, SH24-5030.

Other DL/I publications:

DL/I DOS/VS General Information, GH20-1246

DL/I DOS/VS Library Guide and Master Index, GH24-5008

DL/I DOS/VS Application Programming: High Level Programming Interface, SH24-5009

DL/I DOS/VS Application Programming: CALL and RQDLI Interface, SH12-5411

DL/I DOS/VS Guide for New Users, SH24-5001

DL/I DOS/VS Messages and Codes, SH12-5414

DL/I DOS/VS Diagnostic Guide, SH24-5002

Chapter 1. Introduction

In the course of day-to-day data processing, your system and applications are subject to the possibility of failure in many forms. There may be errors in data transmission, a hardware component may fail, or there may even be a program error in the operating system or in an application program.

When these errors occur, you will want to do two things:

- Correct any damage to your data caused by the failure
- Restart and complete the work in progress at the time.

Recovery is the process by which damage to your data is repaired. Restart is the process by which work in progress at the time of failure is completed, keeping the amount of work that has to be redone to a minimum.

Why Are Recovery and Restart Necessary?

When a failure occurs, your data is usually left in an inconsistent state that would lead to additional errors should it be used. For instance, a failure in an application doing order processing might result in an order being sent, but the inventory record not being updated. A failure during processing of an insert or delete of a data segment could make the data base's internal pointers incorrect, leading to the loss of additional valid data.

If you do not do recovery following a failure, you run the risk of lost data. As additional applications use this erroneous or inconsistent information, the quantity of invalid data increases.

After you have repaired the damage caused by a failure, restart is needed if you are to complete the work in progress at the time of failure. The ability to minimize the amount of work to be redone is not really necessary; however, you could save considerable time and money if you could rerun part of a job instead of having to rerun the entire job from the beginning. For example, if a three hour batch job fails at a point ten minutes before completion, restart allows you to run the remaining ten minutes instead of rerunning the whole three hour job.

DL/I Recovery Concepts

DL/I recovery involves re-establishing your data to its last consistent state, that is, to a point at which DL/I is assured of the integrity of the data base. DL/I provides different mechanisms for doing this, depending on the nature of the failure and the steps you have taken to prepare for it.

Logical Unit of Work (LUW)

The first requirement for recovery is for DL/I to be able to recognize points where the data is consistent. During normal processing, it may temporarily become inconsistent. For example, a program might add a student segment to a class data base record and then update a total count of students in the class segment. Adding the student segment and updating the count constitute a single *logical unit*

of work; the data is consistent before either order action is taken and again after both have been taken, but not when one action has been taken without the other.

The ends of logical units of work, or LUWs, are points to which DL/I can recover and leave the data base in a consistent state. DL/I assumes that the start of a job step or the beginning of a CICS/VS transaction is the beginning of a LUW, and that the end of a job step or a CICS/VS transaction is the end of a LUW. The maximum size for a LUW is, therefore, a batch job step or a CICS/VS transaction. DL/I provides facilities for applications to define smaller LUWs if they wish; this makes it possible to restart a batch job in the middle instead of the beginning.

Backout

If the data medium is still accessible (that is, the disk pack has not been damaged, no read or write errors have occurred, etc.), recovery may be accomplished by undoing the changes made by the LUWs in progress at the point of failure. This process is called *backout*.

Forward Recovery

If the data medium is no longer readable, you will not be able to back out any changes that were made. Instead, you must recreate the data base as it was at some point in the past and reapply all transactions up through the last completed LUW. The process of reapplying transactions is called *forward recovery*.

If you are running CICS/VS at the time of failure, or if your batch job defines LUWs that are smaller than a job step, DL/I forward recovery will even reapply incomplete LUWs. In such cases, forward recovery is followed by backout to complete the recovery process.

Logging

In order to back out incomplete LUWs, the system must have a record of modified data as it looked before it was modified. In order to reapply transactions in forward recovery, it must have a record of modified data as it looked after it was modified. The process of making a copy of before-modification and after-modification records is called *logging*, and the copy is called a *log*. The log is marked to indicate where the LUWs begin and end.

When you are running online sessions, logging is done on a file called the CICS/VS journal. This book will call the journals “logs.”

Backup and Restore

In addition to the log of after-modification records, forward recovery requires a copy of the data base as it was before any of the modifications recorded in the log were made. The process of making a copy of the data base is called *backup*. The backup copy may be in a form in which it cannot be used directly, but can be used to recreate the data base in its original form. The process of recreating the data base is called *restore*.

Planning for Recovery

You will not be able to recover from errors or restart processing unless you plan and take action before failures occur. If you do not make backup copies of your data base, then you will never be able to recover if you have a media failure. If you do not instruct the system to log data base updates as they are made, you will not be able to use backout or forward recovery; instead, you will have to do all your work over again if you have a failure.

Planning for recovery involves trade-offs of computer time and your effort, so it is not a simple matter of deciding to use all the DL/I facilities provided. Logging, for instance, slows down your jobs. If your system is not used heavily at night, you may prefer to back up the data base daily and risk having to repeat up to a day's work after a failure, rather than pay the price of logging. Chapter 2 of this manual provides information that you should find helpful in deciding what type of recovery preparation you should make. Chapter 3 discusses the DL/I facilities provided for implementing your plan. Chapter 4 is a recovery guide which will lead you through the steps you must take after a failure, depending on the kind of preparation you have made.

Chapter 2. Planning Considerations

This chapter will help you decide what kind of recovery/restart capability you want for your system. It discusses the basic strategies you can use and the major choices you must make for each of the steps in those strategies. It does not describe specific DL/I facilities. Once you have decided which capabilities you want to have, you will want to read the sections in Chapter 3 which describe the DL/I facilities for implementing those capabilities.

Basic Choices

The first step you must take in planning for recovery from failure is to select a general strategy. The appropriate strategy for you will depend on the size of your data base, the amount of online activity in your system, the skills of your computer operators, and the demands on computer time in your operation. DL/I offers four basic strategies:

- A bare-bones option that meets the absolute minimum requirements for recovery
- A full-featured option that uses the computer to prepare for maximally efficient recovery
- Two intermediate options that compromise between ease of preparation and ease of recovery.

Restore from Backup and Repeat Work

The conceptually simplest option is to depend on periodic backups of the data base as the only preparation for failure. All failures must be recovered by the procedure illustrated in Figure 1.

- Restore the data base from a backup copy; and
- Repeat all work done since the backup was taken.

The primary advantage of this strategy is its relative simplicity for the computer operations personnel. They need to learn only backup and restoration procedures. Logging, checkpointing, and the techniques for backout and forward recovery are unnecessary.

Because logging and checkpointing are avoided, there is no computer overhead devoted to preparation for recovery during normal applications runs. However, backups should be taken fairly frequently, especially if your work includes online sessions, in order to minimize the work that needs to be repeated.

The large amount of work required during recovery, especially of online users, is the most obvious disadvantage of this strategy of preparing for failure. Furthermore, the data base is unavailable during the frequent backups, and also during the comparatively long recovery process of restoring the data base and resuming batch jobs.

Successful recovery requires you to keep input for all your batch jobs since the last backup (preferably since the last two or three, since backup copies can become

unreadable), as well as the backups themselves. Your online users must also keep records of all their activity between backups.

Because of its disadvantages, this strategy is most appropriate for small data bases that can be quickly backed up, in systems that experience few failures. It is better if there is little online activity. It cannot be used if initial data cannot be recovered (for example, online activity generated directly by customers over phone lines, with no paper records). It is also not applicable to systems in which jobs must be run in certain order, unless that order can be recreated (including any online activity) during recovery.

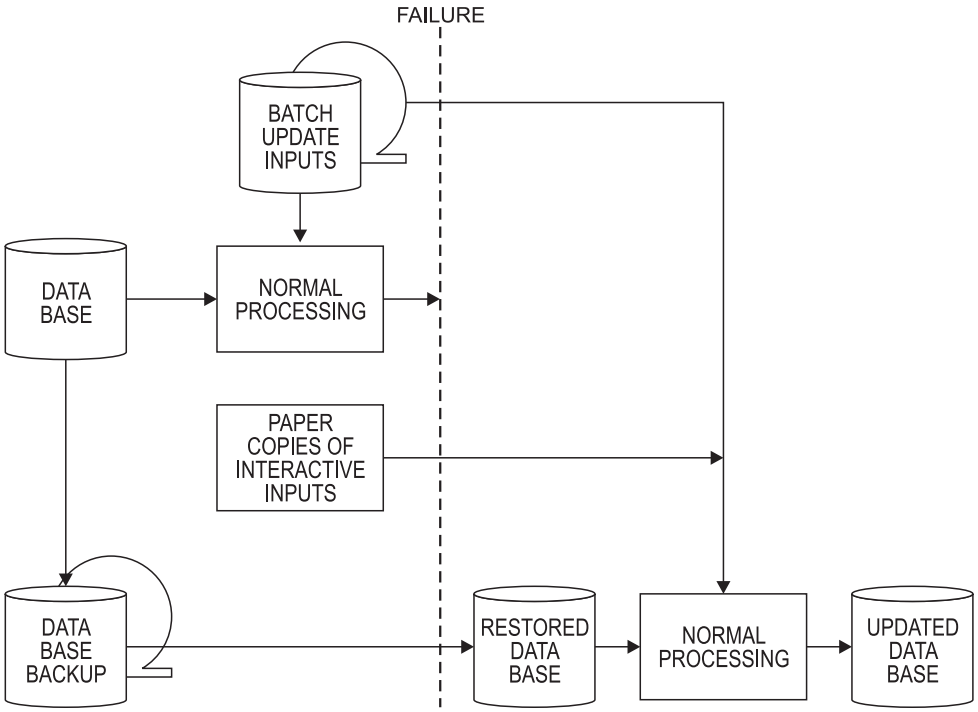

Figure 1. Restoring a Data Base from a Backup Copy

Backout and Forward Recovery Both Available

At the opposite extreme from the option described above, this strategy makes use of the full facilities of DL/I. If the data base is readable after a system failure, any incomplete LUWs are backed out of the data base and the system is restarted from that point, as shown in Figure 2. If the data base is unreadable, it is restored from a backup copy, and the changes made since the backup was taken are reapplied up through the last completed LUW, as illustrated in Figure 3.

This strategy requires operations personnel to be familiar with the full set of DL/I recovery utilities. It provides the fastest recovery from all types of failure, and makes no demands on the users other than to repeat work actually in progress at the time of failure. Recovery is especially fast and easy for online activity when the failure leaves the files readable, as the most common types of failures do.

This strategy incurs overhead during normal applications runs for maintaining the log required for backout and forward recovery. The overhead may be compensated for by reduced backup frequency. Since recovery from a backup copy will be rare and can be performed without repeating all the work, there is generally no need for

the daily backups that are usually required for a strategy that depends exclusively on backup and restore for its recovery procedure.

Successful recovery requires that you ensure that all jobs maintain logs, and that all logs since the last backup (preferably two or three backups) are kept. You will have to know the sequence of the logs if forward recovery becomes necessary.

Since all activity is stored on the logs, this strategy can support a system in which paper copies of online activity are unavailable. It is also appropriate for systems with large data bases that are costly to backup, large amounts of online activity that would be difficult to repeat, or complex applications that may be prone to failures.

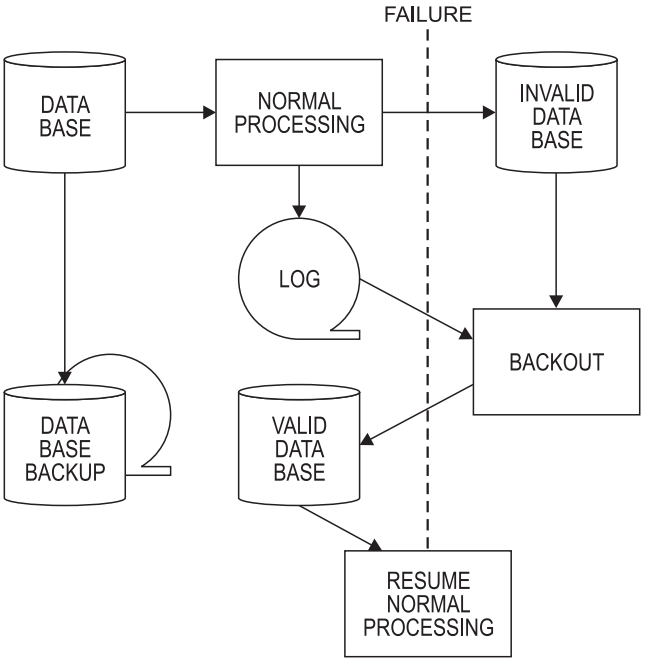

Figure 2. Backing Out Incomplete LUWs

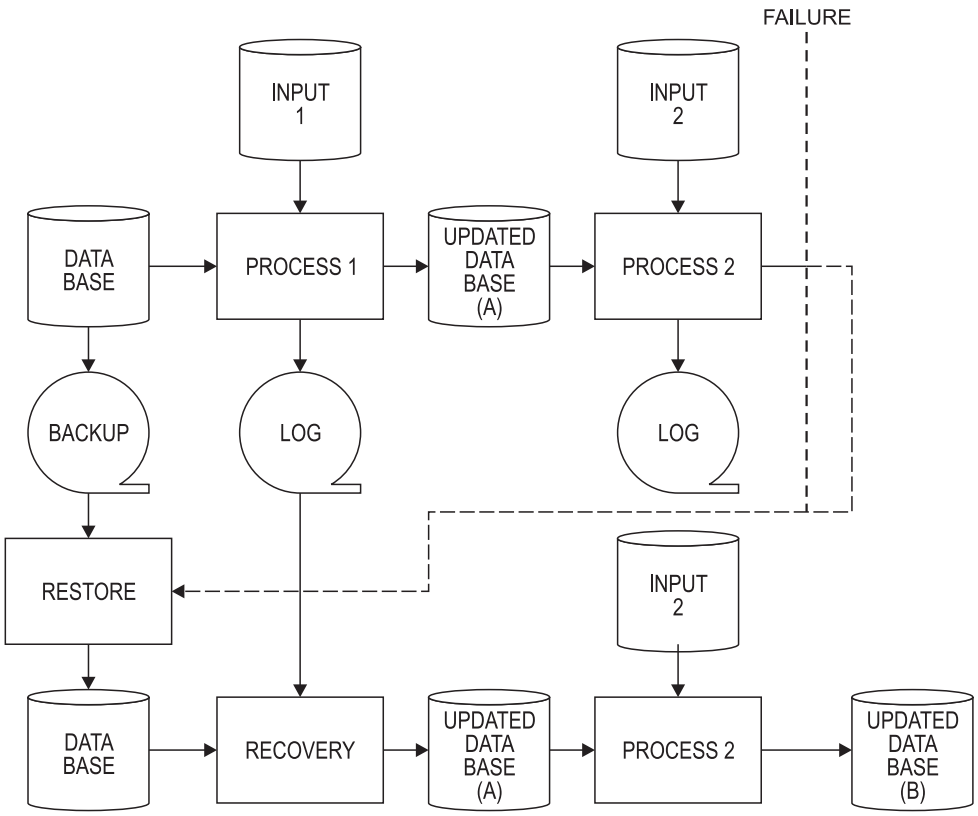

Figure 3. Forward Recovery After Restoring the Data Base

Backout Available, No Forward Recovery

A strategy often applied to online systems is to use the DL/I backout facilities, but to depend on restoring the data base and repeating work in case of a failure that leaves files unreadable. Backout provides very fast recovery of the vast majority of failures, and the backout procedure for online runs is transparent to the computer operator. Thus it has the advantage of efficient recovery from most failures, without requiring appreciably more operator knowledge than is needed for a restore-and-repeat-work strategy. However, this strategy incurs just as much computer time overhead for logging as if forward recovery were also made available. You may also want to take backups as frequently as you would under the restore-and-repeat-work option, since users would have to keep paper copies and repeat work in the case of a failure that left files unreadable.

Successful recovery will usually require only that you be sure each job in an online session keeps a log as it runs. It is not necessary to keep a library of logs since the last backup, but you should keep input for batch jobs, and your users should keep paper copies of their online work, in case of an unreadable-file failure.

Forward Recovery Available, No Backout

For batch jobs only, DL/I provides an especially efficient logging option (asynchronous logging) that produces a log usable for forward recovery but not for backout. This has the advantage of reduced computer overhead during applications runs; but it requires you to restore the data base and reapply changes from the log for any failure, not just for failures that leave files unreadable. If you choose this for your batch jobs, you should use the backout-and-forward-recovery

options for your online sessions. Successful recovery will require you to have logs from all sessions since the data base was backed up, including logs from any online sessions, and to know the sequence in which they were run.

Forward recovery is fairly costly compared to backout, so this strategy is most appropriate for well-tested batch jobs that are not subject to failure. Since forward recovery logs are available, backups do not need to be especially frequent, so this option is appropriate when computer resources required for preparing for failure are at a premium.

Mixed Strategies

It is possible to use one plan of recovery for some jobs, and a different plan for others. For example, installations that use the backout option, with no forward recovery available for their online work, sometimes do not make even backout available for their batch jobs. This kind of strategy is appropriate if it is possible to back up the system just before a series of batch runs, so that a failure during a batch job can be recovered by restoring and repeating the jobs without calling on online users to re-enter all their work.

Another mixed strategy is to make forward recovery as well as backout available for online sessions, but to make neither available for batch jobs. The only difference from the other mixed strategy just described is that you would have to retain logs from the online sessions. If there is a failure during a batch job, you would then have to restore the data base. All batch jobs would have to be rerun, but online sessions could be recovered via forward recovery procedures. Since online work would never have to be repeated, you may be able to take less frequent backups than if forward recovery were not available. Successful recovery using this strategy requires you to know the order in which batch jobs and online sessions were run. You would have to repeat all jobs done before the online session, in order; then recover the online session; then repeat batch jobs that followed the session, etc.

Backup/Restore Options

There are several options available for backing up a data base. The various options address the logical structures of the data base, the physical file organization of the data base, and the physical device on which the data base resides.

A DL/I data base occupies one or two VSAM data sets, which in turn consist of one (data) or two (data and index) components. A VSAM data set component can reside on one or more extents and may also span multiple volumes. Furthermore, the volumes may be fully dedicated to the DL/I data base or may contain other files. The catalog for the data base does not necessarily reside on the same volume as the data base and may refer to non-DL/I data base files. Figure 4 illustrates one way in which a HISAM data base could be assigned to different disk volumes.

The choice of a backup strategy is best determined through analysis of the individual system's requirements regarding speed, logical organization and data base complexity. The three options discussed in this section are data-oriented backup, file-oriented backup, and volume-oriented backup.

Data-Oriented

Data-oriented backup/restore involves copying the logical structures of the data base into a backup copy, and recopying them into empty data base files if a restore is needed. DL/I reorganizes the data when it is being copied back into the files, leaving a more efficient file structure that may improve DL/I performance.

Segment-by-segment reading and writing of the data base is a very slow process. Furthermore, you must both read and rewrite the data base in the course of backing it up if you want to perform forward recovery from logs. The logs refer to the physical location of data within files, and they will be incorrect after a reload unless the same reload, creating the same physical organization, immediately precedes the beginning of logging.

Data-oriented backup is primarily of use when you want to combine backup with other operations. If you want to read all the data segments to validate them, you may want to make a copy while you are reading them to serve as a backup. Similarly, if you want to reorganize the data base, you may want to keep the unloaded tape after your reorganization reload, rather than take the time for a separate backup.

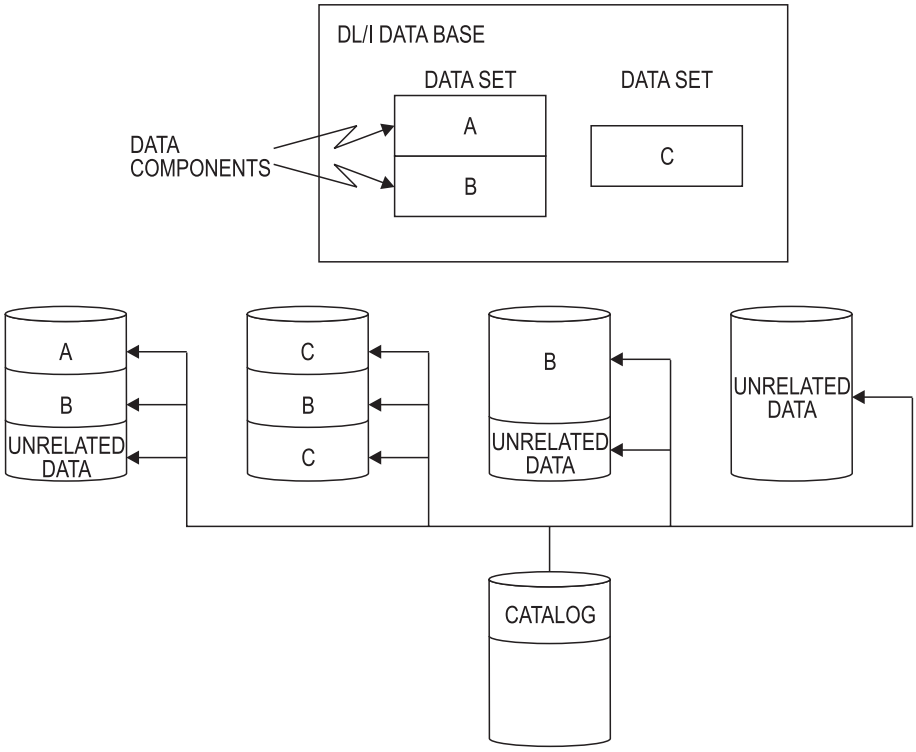

Figure 4. Possible Allocation of Data Base to Multiple Volumes Containing Unrelated Data

File-Oriented

The most common approach to data base backup and restore is to copy the data base files without reference to the internal logical structure of the data within those files. Like data-oriented backup/restore, it leaves unrelated data alone and ensures that catalogs are left in a valid state. It does not reorganize data within files, so it is compatible with forward recovery from a log. Its only disadvantage is that, for very large data bases, it is slower than volume-oriented techniques.

Volume-Oriented

A volume-oriented approach to backup and restore involves copying the entire contents of a disk or other device on which the data base resides. Since the backup can ignore file-oriented concerns like blocking factors, the process can proceed very rapidly. The restore process can also be very quick. If a spare disk pack is available, restoring can consist simply of mounting the backup pack, without any copying at all. Since the backup copy is physically identical to the original, it is compatible with forward recovery from a log.

Volume-oriented backup and restore have a number of potential disadvantages. If unrelated data is maintained on the same disk as the data base, any changes to the unrelated data will be lost when the data base is restored. Also, if the unrelated data has been moved, or if the data base files themselves have acquired additional extents since the backup was taken, any catalog referring to the restored volume would not be valid. Such unrelated data and catalogs would have to be separately restored at the same time the data base is restored.

Because of the disadvantages cited above, volume-oriented backup and restore is most appropriate for large data bases, or sets of related data bases, that fully occupy one or more volumes, preferably with the catalogs maintained on the same volume as the data base itself.

Logging

Logging is the process of making a record of all changes to the data base as they are made. A synchronous log guarantees that changes are written to the log before they are written to the data base, which is a necessary condition if the log is to be used for backout. An asynchronous log is written independently of the data base. Since the system does not have to wait until the log record is written before writing to the data base, the result is improved system throughput, especially if a tape log is used. The price is that the log may not reflect all changes to the data base, in which case it can be used only for forward recovery. In DL/I, asynchronous logging is available only for batch jobs.

If you plan to do forward recovery, you must save all the logs from all jobs and online sessions since the last backup. Backout requires only the logs from those jobs and transactions active when the failure occurred. If your strategy calls for depending entirely on restoring the data base from a backup copy and repeating the work, you should not use logging, as it provides no benefit and consumes computer resources.

Defining Logical Units of Work (LUWs)

A logical unit of work (LUW) is a series of modifications to the data base which, together, change the data from one internally consistent state to another. To protect you from invalid data, the backout and forward recovery procedures will leave the data base in the state it was in at the completion of the most recent LUW. All work done since that LUW must be repeated.

DL/I normally treats a job step or a CICS/VS transaction as a single LUW. For long-running jobs, this may entail considerable loss of work in case of system failure, even when backout and forward recovery are available. To minimize the amount of work to be done, you can specify checkpoints (batch, MPS batch) or

syncpoints (online) that mark the ends of LUWs within job steps or transactions. These points are written to the log, so they can be used for backout or forward recovery.

Checkpoints and syncpoints offer advantages in terms of reduced repetition of work after a failure. In MPS batch and online work, restarting from a checkpoint or syncpoint is about as easy as running a job over. For batch applications, unless the program is written in such a manner as to be able to detect where it is after restart, it is necessary to modify the input file to delete records already processed at the time of system failure.

Checkpoints and syncpoints must be coded into your application programs, so a decision whether to use them must take into account the cost of modifying programs to include them, or the cost of including them in new programs, as well as the additional procedural complexity required for restarting batch jobs. Checkpoints and syncpoints also require some computer overhead. MPS batch checkpoints, which include a partition dump (VSE checkpoint as well as DL/I checkpoint) require the most overhead, as well as extra disk space in which to hold the partition.

Backout

Backout consists of reversing changes made to the data base, beginning with the most recent and working backwards until the beginning of the incomplete LUW is found. It is by far the fastest recovery option. You can make backout available for any critical job, even if your overall strategy does not call for backout to be routinely available, simply by specifying that the job will keep a synchronous log. The only other preparation required is that your operations personnel be familiar with proper backout procedures.

Since backout depends on reading the data base as it was left as the point of failure, it cannot be used for failures such as disk head crashes that physically destroy the data base files.

Forward Recovery

Forward recovery is the process of bringing a restored copy of the data base up to date by reapplying all changes since the last backup was taken, up to the end of the last complete LUW. You can use forward recovery for any type of failure. However, you must have logs for all jobs run since the last backup was taken. Attempting to run a forward recovery with some of the logs missing will destroy the integrity of your data base.

If a log is missing, you can still run forward recovery, but only up to the point of the missing log. If you are then able to repeat the work for which the log is missing (as by rerunning a batch job), you can consider the data base to have been restored to the end of the batch job, and you can run another forward recovery from that point up to the point of failure (or to the next missing log).

DL/I provides an option (change accumulation) for speeding up forward recovery by preprocessing the logs to omit intermediate changes when a segment has been repeatedly updated and to put the changes in sequential order by data base record. Also, because change accumulation merges multiple log volumes into a single

accumulated volume, there are fewer volumes that have to be dedicated. The change accumulation capability depends on the use of specific utilities for data base backup; see Chapter 3 for details.

Preparation for forward recovery requires you to have all jobs keep logs, and to establish procedures for saving those logs. Your personnel will have to be able to use forward recovery techniques. They should also know backout techniques, since DL/I forward recovery of an online session or of a checkpointed batch job requires recovering all the way to the point of failure, then backing out to the end of the last completed LUW.

Restart

Restart consists of beginning your applications again after the data base has been recovered. No special preparation is required except for batch jobs that use DL/I checkpointing. For those jobs, it may be necessary to examine the DL/I log to determine the point of failure, and then to delete from the job's input file all records that have already been processed. Whether that is necessary before restart depends on whether the job is designed to recognize it is being restarted and to bypass completed processing.

Chapter 3. Facilities

This chapter introduces the various facilities that are used in planning and effecting recovery and restart. Where more than one facility is provided for the same purpose, their characteristics are compared. The chapter is divided into 6 major areas:

- Backing up and restoring the data base using data-, file- and volume-oriented utilities
- Logging DL/I transactions on the DL/I log (batch only), the CICS/VS dynamic transaction log (online and MPS batch), and the forward recovery log generated by the backout utilities
- Defining logical units of work through checkpointing and syncpointing
- Backout using the DL/I Data Base Backout utility (with the Log Print utility) or the CICS/VS dynamic transaction backout facility
- Forward recovery using the Change Accumulation utility and the Data Set Recovery utility
- Restart.

Backup/Restore

DL/I provides utilities for data-oriented and file-oriented backup. Volume-oriented utilities are also available under VSE. Utilities for DL/I are discussed in “Data-Oriented” and in “File-Oriented.” Volume-oriented utilities provided by VSE are discussed in “Volume-Oriented.” Note that all related data bases, e.g., an HDAM data base and its associated INDEX data bases, must be copied at once to ensure consistency when they are restored.

Data-Oriented

The two utilities that are data-oriented are the HD Reorganization Unload/Reload and the HISAM Reorganization Unload/Reload.

The HD Reorganization Unload/Reload may be used for all DL/I data bases subject to recovery, except INDEX data bases.

The HISAM Reorganization Unload/Reload may be used only for HISAM, SHISAM, and INDEX data bases. These HISAM utilities are faster than the HD utilities.

You may want to write your own application programs to validate segment data while backing up your data base(s).

If you use one of the unload utilities to back up your data base, you must use the corresponding reload utility to restore it. If you intend to use the forward recovery process with this form of backup, the you must reload the data bases immediately after unload. Otherwise, your forward recovery logs will be keyed to your unreorganized data base and be invalid with reference to your reorganized data base backup. You cannot use the change accumulation facility (See “Change Accumulation Utility”) for forward recovery if you use this type of backup.

HD Reorganization Unload/Reload Utilities

The HD Reorganization utilities are designed to provide a means of reorganizing HD-organized data bases. They may also be used for SHISAM or HISAM data bases, but the HISAM Reorganization utilities are more efficient.

These utilities copy HD, HDAM, HIDAM, HISAM, and SHISAM data bases by segment onto a sequential tape or disk file, output messages and descriptive statistics, reorganize the data base(s) and reload the data base(s). Reorganization of a data base using these utilities will also require other utilities if your data bases contain logical relationships or have related INDEX data bases:

- The Data Base Prereorganization utility generates a control file for use by the remaining utilities. It must be run once whenever data base definitions are changed; it does not have to be run for every reorganization job.
- The Data Base Scan utility generates a data file of logical relationships that will have to be recreated after reorganization.
- The Data Base Prefix Resolution utility creates a work file of recreated logical relationships.
- The Data Base Prefix Update utility applies the recreated logical relationships to the reorganized data base and/or regenerated INDEX data bases.

A more complete description of these utilities may be found in *DL/I DOS/VS Resource Definition and Utilities* or *DL/I DOS/VS Interactive Resource Definition and Utilities*.

HD Reorganization Unload Utility

The HD Reorganization Unload utility uses DL/I logic to access segments of the specified data base. It writes a sequential output file containing each active segment, together with DL/I utility prefix information, arranged in hierarchical sequence.

Several features are incorporated in this program.

1. If two output units are specified, two copies of the data base can be created.
2. A checkpoint facility is included to allow you to recover from a failure without having to rerun the entire unload. This facility takes checkpoints at intervals either specified by you or calculated by the utility (approximately every 5000 segments).
3. A sequence check verifies the sequence of the data base.

Input to the HD Reorganization Unload utility consists of:

1. A DMB and PSB loaded from a core image library. The DMB and PSB names are obtained by DL/I by expanding the DBD name specified in a parameter statement.
2. One of the following data bases to be unloaded:
   - HD randomized data base
   - HD indexed data base
   - HDAM data base
   - HIDAM data base
   - HISAM data base
   - SHISAM data base.

3. Primary and secondary index data bases related to the data base to be unloaded. The index data bases are not unloaded.
4. When restarting from a checkpoint, a copy of the partially unloaded data base from the previously interrupted HD Reorganization Unload execution must be included. The number of the last successful checkpoint record is entered via SYSLOG.

Output from the HD Reorganization Unload utility consists of:

1. One copy (HDUNLD1) or two copies (HDUNLD1 and HDUNLD2) of the unloaded data base, including checkpoint records.
2. Checkpoint data written to SYSLOG.
3. Messages and statistics written to the SYSLST device.

HD Reorganization Reload Process

The HD Reorganization Reload utility is designed to accept a sequential file created by the HD Reorganization Unload utility and reload the data base from it. You must execute the VSAM Access Method Services utility command DELETE to remove the name of the file from the VSAM catalog to release the space allocated for it. You must then execute the VSAM Access Method Services utility command DEFINE to define the new file to VSAM to cause its definition to be recorded on the VSAM catalog.

The general operation consists of reading an input record and generating a DL/I insert call (ISRT) to cause the segment to be inserted. The result is a data base organized in an efficient accessing format with all fragmented free space consolidated.

If any data bases are logically related to the data base being reorganized, running HD Reorganization Reload is only one of several steps involved in restoring/reorganizing the data base. The steps are:

- Run the Data Base Prereorganization utility. This step may be omitted if the utility's output file is available from an earlier run, and the data bases have not been redefined (no new segments, logical relationships, or indexes) since the earlier run.
- Run the Data Base Scan utility. This must be done for any data bases identified in DBS messages from the Prereorganization utility.
- Run the HD Reorganization Unload utility for all data bases being reorganized.
- Delete the data base files to release space, and define the files to receive the reorganized data bases.
- Run the HD Reorganization Reload utility for all data bases being reorganized.
- Run the Data Base Prefix Resolution utility.
- Run the Data Base Prefix Update utility.

Since you will normally use a Reorganization Unload output as a backup only if you were planning to reorganize the data bases anyway, these steps will normally be performed at the same time that the HD Reorganization Unload utility is run. In the event of failure, restoring the data base will consist of rerunning all reorganization steps after the unload.

Data Base Prereorganization Utility

Input to the Data Base Prereorganization utility consists of:

1. One or more control statements read from the SYSIPT device. They contain the names of the data bases that are being reorganized.
2. One or more DMBs and PSBs loaded from a core image library. These names are obtained by expanding the DBD name(s) specified in the control statement(s).

Output from the Data Base Prereorganization utility consists of:

1. The output control file to be used by the Prefix Resolution utility, the data base work file generator module during HD reload, and the Data Base Scan utility.
2. Data Base Scan control statements on SYSPCH, to be used as input to the Data Base Scan utility program (optional).
3. Messages and statistics on the SYSLST device.

Data Base Scan Utility

You must run the Data Base Scan utility for any data bases identified in DBS= output messages from the Prereorganization utility.

Input to the Data Base Scan utility consists of:

1. One or more control statements read from the SYSIPT device containing the data base scan control statements.
2. The input control file created by the Data Base Prereorganization utility.
3. One or more data bases to be scanned: valid data bases are HD randomized, HDAM, HD indexed, and HIDAM. If the data base being scanned contains logical relationships or indexes, the related data base(s) must also be online. Secondary index data bases are not accessed by this utility, but job control language statements for them are required.
4. If restarting from an incomplete run of the Scan utility, a copy of the partially created work file from the previously interrupted run must be included. The number of the last successful checkpoint record is entered from the SYSIPT device.

Output from the Data Base Scan utility consists of:

1. An output work file to be used as input to the Prefix Resolution utility.
2. Checkpoint data written to SYSLOG (if requested).
3. Messages and statistics on the SYSLST device.

HD Reorganization Reload Utility

If there are no data bases logically related to the data base being reorganized, or after the Prereorganization and Scan utilities have been run, the HD Reorganization Reload utility may be run. Input consists of:

1. A DMB and PSB loaded from a core image library. The DMB and PSB names are obtained by DL/I by expanding the DBD generation name obtained from the parameter statement.
2. A copy of the unloaded data base.

3. A control file, if logical relationships exist. This file has been previously created by the Prereorganization utility.

Output from the HD Reorganization Reload utility consists of:

1. One of the following reorganized data bases:
   - HD randomized data base
   - HD indexed data base
   - HDAM data base
   - HIDAM data base with its newly created INDEX
   - HISAM data base
   - SHISAM data base.
2. A work file, if logical relationships or secondary indexes exist. This file is used as input to the prefix resolution program.
3. Messages and statistics on the SYSLST device.

Data Base Prefix Resolution Utility

Input to the Data Base Prefix Resolution utility consists of:

1. The control file created by the Data Base Prereorganization utility.
2. All work files generated by Data Base Reorganization Reload, and Data Base Scan utilities.
3. One control statement read from the SYSIPT device.

Output from the Data Base Prefix Resolution utility consists of:

1. A work file that contains all the sorted input work file records for logical relationships and/or a similar file for secondary index relationships.
2. Statistics on the SYSLST device.
3. Messages on SYSLST and SYSLOG.

Data Base Prefix Update Utility

Input to the Data Base Prefix Update utility consists of:

1. One or two input work files created by the Data Base Prefix Resolution utility.
2. One or more data bases that were reorganized and any logically related data bases: valid data bases are HD randomized, HDAM, HD indexed, HIDAM, and primary INDEX.
3. One control statement read from the SYSIPT device.

Output from the Data Base Prefix Update utility consists of:

1. The updated data bases that were input to the Data Base Prefix Update (see 2 above) and, optionally, secondary index data bases created by this utility.
2. Messages and statistics on the SYSLST device.

HISAM Reorganization Unload/Reload Utilities

The HISAM Reorganization Unload and Reload utilities reorganize the data base by:

1. Copy HISAM, SHISAM, and INDEX data bases to sequential tape or disk file
2. Reload the data bases, and
3. Generate statistics about the data bases.

More complete information may be obtained by referring to *DL/I DOS/VS Resource Definition and Utilities*. Information on interactive generation of these utility job streams is contained in the *DL/I DOS/VS Interactive Resource Definition and Utilities* manual.

The HISAM utilities are faster than the HD utilities.

HISAM Reorganization Unload Utility

The HISAM Reorganization Unload utility program is primarily designed to provide an efficient means of reorganizing SHISAM, HISAM, and INDEX data bases. Additionally, the reorganized output can be used as input to the Data Base Data Set Recovery utility, allowing you to skip the data base restore step needed before forward recovery. If the output of the HISAM Reorganization Unload utility is to be used for forward recovery, you must reload the data base with the HISAM Reorganization Reload utility prior to applying changes to the data base. The reload is necessary so that log tapes created after this unload refer to the correct data, which may have moved as a result of the reorganization. HISAM Reorganization Unload is normally used as a backup procedure only if you are planning to reorganize the data base at the same time as backing it up.

The general operation consists of reading KSDS records, picking up any overflow dependent segments from the ESDS, dropping deleted root and dependent segments, and writing a sequential file arranged in a physically related sequence. Two copies of the data base may be produced.

Input to the HISAM Reorganization Unload utility consists of:

1. A control statement read from the SYSIPT device containing the DBD name of the SHISAM, HISAM, or INDEX data base to be reorganized, the KSDS filename, and output filename(s).
2. A DMD loaded from a core image library.
3. The input KSDS and ESDS which constitute the HISAM data base (KSDS only if SHISAM or INDEX).

Output from the HISAM Reorganization Unload utility consists of:

1. One or two copies of the reorganized data base. Tape or DASD may be used and multiple copies may be produced on mixed device types.
2. Message and statistics on the SYSLST device.

HISAM Reload Procedures

A HISAM Reorganization Unload backup may be restored by means of either the HISAM Reorganization Reload utility or the Data Set Recovery utility. Since you will normally use HISAM Reorganization Unload as a backup only if you wanted to

reorganize the data base at the same time, you will normally run HISAM Reorganization Reload at the same time your backup is taken. If the data base needs to be restored, you can simply resubmit the reload portion of you reorganization job. The Data Set Recovery utility is primarily intended for forward recovery using logs (see “Data Set Recovery Utility”); but if no logs are available, it will restore the data base without doing forward recovery.

HISAM Reorganization Reload Utility

The HISAM Reorganization Reload utility is a means of reloading SHISAM, HISAM, or INDEX data bases which have been unloaded and reorganized by the HISAM Reorganization Unload utility. The general operation consists of reading an input file created by the HISAM Reorganization Unload utility and writing the records to the appropriate KSDS or ESDS.

Prior to executing the HISAM Reorganization Reload utility, you must execute the VSAM Access Method Services utility command DELETE to remove the name of the file from the VSAM catalog and release the space allocated for it. You must then define the new file to VSAM to cause its definition to be recorded on the VSAM catalog.

Input to the HISAM Reorganization Reload utility consists of:

1. A control statement read from the SYSIPT device containing the symbolic filename of the input file.
2. The input file containing the unloaded SHISAM, HISAM, or INDEX data base.

Output from the HISAM Reorganization Reload utility consists of:

1. The reloaded KSDS and ESDS which constitute the HISAM data base (KSDS only if SHISAM or INDEX).
2. Messages and statistics on the SYSLST device.

The HISAM Reorganization Reload utility provides statistics for the data base reloaded. Following the page heading are the various messages generated if errors occurred for this execution. An explanation of all numbered messages may be found in *DL/I DOS/VS Messages and Codes*.

Data Set Recovery Utility

If you use the Data Set Recovery utility for restoring from a HISAM Reorganization Unload Backup, input consists of:

1. Control statements read from the SYSIPT device containing the DBD name and the name of the file to be recovered.
2. A DMB and utility PSB loaded from a core image library. The DMB and PSB names are obtained by DL/I by expanding the DBD name specified in the parameter statement.
3. An input file consisting of the output of the HISAM Reorganization Unload utility.

Before restoring the data base, the user must execute the VSAM Access Method Services utility command DELETE to remove the data base data set name from the VSAM catalog and release the space allocated for it. He must then DEFINE the

new file to VSAM to cause a definition of the new file to be recorded on the VSAM catalog.

Output from the Data Base Data Set Recovery utility consists of:

1. A recovered file.
2. Informational messages written to the SYSLST device.
3. Error messages written to the SYSLST and SYSLOG devices.

File-Oriented

DL/I provides file-oriented backup facilities for copying data base files without reorganizing their contents. The utility for direct copying of data base(s) is the Data Set Image Copy. This utility is tailored for DL/I backup use when recovering using forward recovery. The backup file created may be used directly for input to the Data Set Recovery utility, bypassing the need to restore the data base before forward recovery. You can also use the change accumulation facility (See “Change Accumulation Utility”) during forward recovery from this backup. You will need to know how many files are in your data base and their names to run this facility.

More complete information about this facility may be found in *DL/I DOS/VS Resource Definition and Utilities* or *DL/I DOS/VS Interactive Resource Definition and Utilities*.

Backup Using Data Set Image Copy

Data Set Image Copy is designed to back up data sets rather than data bases. In most cases, a data base is synonymous with a data set, but HSAM data bases consist of two data sets. It is necessary to specify each data set to be backed up to the Data Set Image Copy utility, even for HISAM data bases. As with any backup method, you must back up all logically related data bases at one time; therefore, you may specify many data sets (all data sets in all logically related data bases) for one run of Data Set Image Copy.

The Data Set Image Copy utility achieves a high rate of throughput by handling files in a physical sequential fashion, without concerning itself with physical or logical data relationships. VSAM is used to process the input files. SAM is used to process the output files.

Input to the Data Base Data Set Image Copy utility consists of:

1. A control statement read from the SYSIPT device containing the DBD name and the name of the file to be copied. One or more control statements may be processed in one execution.
2. A DMB loaded from a core image library. The DMB name was obtained by expanding the DBD name specified in the control statement.
3. The input file(s) to be copied and referenced by the control statement(s).

Output from the Data Base Data Set Image Copy utility consists of:

1. One or two copies of the dumped file. Tape or DASD may be used and multiple copies may be produced on mixed device types. REWIND=UNLOAD is specified in the DTF for tape files, but may be changed to REWIND=NORWD by using the UPSI job control statement.

2. Informational messages and statistics on the SYSLST device.
3. Error messages on the SYSLST and SYSLOG devices.

Restoring from a Data Set Image Copy Backup

In order to restore a data base which was backed up with Data Set Image Copy, you must use the Data Set Recovery utility. This utility can be used for forward recovery, but it can also simply restore a data base. The utility must be run for each VSAM file in the data base to be restored.

When used for data base restore, input to the Data Set Recovery utility consists of:

1. Control statements read from the SYSIPT device containing the DBD name and the name of the file to be recovered.
2. A DMB and utility PSB loaded from a core image library. The DMB and PSB names are obtained by DL/I by expanding the DBD name specified in the parameter statement.
3. An input file consisting of the output of the Data Set Image Copy utility.

Before restoring the data base, the user must execute the VSAM Access Method Services utility command DELETE to remove the data base data set name from the VSAM catalog and release the space allocated for it. He must then DEFINE the new file to VSAM to cause a definition of the new file to be recorded on the VSAM catalog.

Output from the Data Base Data Set Recovery utility consists of:

1. A recovered file.
2. Informational messages written to the SYSLST device.
3. Error messages written to the SYSLST and SYSLOG devices.

Volume-Oriented

VSE provides physical input/output control system (PIOCS)-oriented I/O for fast backup copy to another disk or a dump tape. There are two utilities available:

- Fast Copy Disk
- Copy Disk Volume (Fast Copy Data Set Program).

Copy Disk Volume is an expanded version of the Fast Copy Disk utility. One advantage that Copy Disk Volume has is its ability to restore a single file from a full volume dump.

These utilities are most useful when your data base occupies full volumes, rather than being distributed over volumes shared with unrelated data. They are also useful if files other than your data base files need to be backed up during your scheduled backups. Care must be taken with respect to unrelated data when using these utilities, as such data may interfere with the extents on the target device.

Backups taken with these utilities are not compatible with the use of the change accumulation facility (See “Change Accumulation Utility”) during forward recovery.

The features of Fast VSE/Copy Disk and VSE/Copy Disk Volume are very similar. Therefore, in the following discussion, each applicable, specific, volume-oriented

feature of VSE/Fast Copy Disk will be addressed first and the differences of that feature in VSE/Copy Disk Volume will be noted.

More complete information may be obtained from *VSE/Fast Copy Data Set Installation Reference*. *VSE/FAST COPY DISK/COPY DISK VOLUME*

The purpose of the Fast Copy Disk and Copy Disk Volume utilities is to copy a file from one disk device to another of the same type in a short time. Data may be copied directly from one disk device to another, or it may be written on a dump data set (usually tape) to be restored at a later time.

The Fast Copy Disk and Copy Disk Volume utilities operate on any disk device supported by VSE. The target volume must be of the same type as the source volume; that is, it must have the same blocksize if it is an fixed block architecture (FBA) device or the same track geometry if it is a count-key-data (CKD) device. The Fast Copy Disk (FBA) utility operates on FBA disks as source and target disks; the Fast Copy Disk (CKD) utility operates on CKD disks.

Fast Copy Disk and Copy Disk Volume operate either on a:

- Volume, that is, volume-specific entities (IPL record, volume label, VTOC and the set of files stored on the volume, or on a
- Single file at a time, that is, the labels of one file (out of a VTOC) and all the data extents described by these labels.

In addition, Copy Disk Volume operates on:

- A partial volume; that is, individual files such as all expired files may be excluded from the copy or dump process, or on
- Multivolume files; that is, you may copy a single multivolume file to one or several volumes of the same device type, or you can dump this file to tape or disk and later on restore it to one or several volumes of the same device type as the source device.

A special volume function is provided for exceptional situations when the VTOC is no longer valid; this function processes an entire volume physically rather than being VTOC-driven.

The dump data set is a sequential file. For Fast Copy Disk (CKD) it must be on magnetic tape; for Fast Copy Disk (FBA), it can be on magnetic tape, on any CKD disk, or on any FBA disk.

The following functional support is provided by Fast Copy Disk and Copy Disk Volume.

Volume Dump

This function copies all data from the input volume to a dump data set from which the data can later be restored to the same or a different device (of the same type). The dump data set can only be used as input to the Volume Restore function of this utility to restore the dumped data to a disk device.

Volume Restore

This function restores all data from a dump data set produced by the Volume Dump function to a disk device, which has the same blocksize (FBA) or track geometry (CKD) as the source device from which the dump was taken. Also, this target device must have equal or larger capacity relative to the source device. Before any data is written, the target device is checked for the presence of data to be preserved, that is, unexpired and/or data secured files. For any such file, the file identification is issued on SYSLOG at the operator's request. The operator may then either cancel the job or proceed with the Restore operation, thereby deleting those files. Data protected files cannot be deleted.

Volume Copy

This function copies all data from the source device to a target device directly, that is, without using a dump data set. The target device has to meet the same criteria as a target device in a Volume Restore operation.

With Copy Disk Volume you may process *all or no* VSAM data sets of a volume in volume dump, restore, or copy operations. This ensures that the catalog or the catalog recovery area is processed together with the data spaces, and that catalog recovery (an Access Method Services function) can be applied after a volume dump has been restored.

Copy Disk Volume has the following additional features not available with Fast Disk Copy.

Selective Restore

You can restore individual non-VSE/VSAM files from one volume dump to disk devices which are of the same type as the source device. The selective restore function allows you to restore an individual file from a tape which contains the data of a whole volume.

Note: VSE/VSAM spaces cannot be restored via selective restore.

Partial Volume Dump/Copy

You can copy or dump part of a volume; for example, you can

- Copy all of a volume with the exception of file A, file B,....
- Exclude all expired files from dumping
- Exclude all VSAM data sets from dumping.

This function offers a performance improvement by avoiding the processing of unimportant files. It also assists you in avoiding unnecessary backup/restore of data unrelated to your data base.

Note that for *partial volume copy* the volume identification written on the output disk *must* be different from the volume identification of the input disk.

NOREWIND option

You can have several dump data sets (volume or file) on one tape using the NOREWIND parameter for the dump or for the restore function. When one dump

data set is written to or restored from tape and the NOREWIND parameter is specified, the VSE/Fast Copy Data Set program positions the tape at the end of this dump data set and does not perform a REWIND. If NOREWIND is not specified, rewind is performed before and after execution of the particular function.

Multivolume File Dump/Copy/Restore

With this function a single file (specified via DLBL/EXTENT statements) which spans several volumes (multivolume file) may be copied to one or several volumes of the same device type or may be dumped to one “file dump data set” on tape(s) or disk(s). This “file dump data set” may be restored to one or several volumes of the same device type.

Procedures for running Fast Copy Disk are presented in *VSE/Advanced Functions Utilities*. Procedures for Copy Disk Volume are in the *VSE/Fast Copy Data Set Installation Reference*.

Logging

Separate logs are provided by DL/I (recommended for batch jobs only) and CICS/VS (online or MPS batch). Logging is performed by putting a record on tape or disk of before and after images of all changes to your data base. The before images are used for backout. The after images are used for forward recovery.

If you want logging you must specify it for each job you run, whether batch, online, or MPS batch. If you use the system default, you will not get a log.

DL/I Log

The DL/I Log is used for batch jobs. It may also be selected for online or MPS batch jobs; however, it may be used for backout and forward recovery only if it was created by a batch job.

You must select synchronous logging if you have planned to do backout. Faster asynchronous logging may be selected if you intend to use the DL/I Log for only forward recovery.

The UPSI byte setup, JCL requirements, and parameter statement are shown below for specifying logging and indicating whether it is to be synchronous or asynchronous.

To ask for the log, you must set UPSI bit 0 to 1 and bit 6 to 0:

```
// UPSI 1xxxxx0x          (x may be either 0 or 1)
```

The following ASSGN statement is applicable to tape:

```
// ASSGN      SYS011,cuu
// TLBL       LOGOUT
```

The following ASSGN statement is applicable to disk:

```
// ASSGN      SYS011,cuu
// DLBL       DSKLOG1
// DLBL       DSKLOG2
```

DL/I application programs use the following parameter statement:

```
{DLI}, progname,psbname[,{.buf}]
{DLR}                    {1  }
{,HDBFR=({bufno} [,dbdname1,dbdname2,...])][,...]
         {32   }
[,HSBFR=({indno} ,{ksdsbuf},[{esdsbuf}],dbdname3)][,...]
         {3    }  {2      }  {2      }
[,TRACE=modname][,ASLOG=YES][,LOG=({TAPE },{PAUSE  })]
                                  {DISK1} {NOPAUSE}
                                  {DISK2}
```

Use ASLOG=YES if you want an asynchronous log, good only for forward recovery. If you want synchronous logging, omit the ASLOG parameter.

You must include the LOG parameter if your log is to go to disk. DISK1 indicates a single disk log file; DISK2 indicates two files. If there are two files specified, the system will switch to the second when the first is full; if there is only one file, or if the second log becomes full, the system will reuse the first log. Note that a log which has been reused cannot be used for forward recovery, nor can it be used for backout unless program checkpoints call for LUWs small enough to ensure that the beginning of an incomplete LUW can never be lost. If there is only one disk file, any LUW that is in progress at the time the log begins to be reused cannot be backed out, so you should specify a single disk log only when you are sure that the space assigned is sufficient.

Unless NOPAUSE is specified, the system will wait for an operator response before switching files or reusing the only file. The wait allows the operator to protect your recovery capability by cancelling a job that would otherwise overwrite the log. You should specify NOPAUSE only when using two disk logs with a program that takes checkpoints, when backout capability is part of your failure protection strategy but forward recovery capability is not.

More information about DL/I Log may be obtained from *DL/I DOS/VS Resource Definition and Utilities*.

CICS/VS System Journal File

The CICS/VS System Journal is the log used for online and MPS batch processing. It can be used for backout and forward recovery of online and MPS batch application programs.

If DTB (Dynamic Transaction Backout) is selected, a log record is also written to the CICS/VS Dynamic Transaction Log that is located in CPU storage. This is not a permanent log since it will be lost if the CICS/VS environment is lost. It is only used for DTB.

Both before and after images are recorded on the CICS/VS System Journal file. This file may be used for input to the DL/I Data Set Recovery utility and the DL/I Data Set Backout utility.

To use the CICS/VS System Journal File with either of these DL/I utilities, the UPSI bit 7 must be set to zero, the System Initialization Table (SIT) must have been coded with DL/I=YES and JCT=01, and the following Journal Control Table (JCT) coding example should be used:

```
DFHJCT TYPE=INITIAL,SUFFIX=01
DFHJCT TYPE=ENTRY,JFILEID=SYSTEM,...

DFHJCT TYPE=FINAL
```

More information about setting up the CICS/VS System Journal File may be obtained from *DL/I DOS/VS Resource Definition and Utilities* and *CICS/DOS/VS Installation and Operations Guide*.

Forward Recovery Log From Backout Utility

A separate log for forward recovery is automatically generated by the DL/I Data Set Backout utility. This log is needed if forward recovery is used for a failure which occurs after backout is used for an earlier failure, and before a new data base backup is taken.

Since this log is automatically generated, you do not have to encode any JCL instructions for it.

Defining the Last Valid State

Checkpoints in batch mode and syncpoints in online mode define the ends of LUWs (Logical Units of Work). The goal of recovery is to remove all incomplete LUWs from the data base(s). Checkpoints are used for batch processing. Syncpoints are issued by online applications. MPS batch uses a combination of a VSE checkpoint call to save the processing environment and a DL/I checkpoint call to trigger the DL/I checkpoint processing and CICS/VS syncpointing. Checkpoints and syncpoints make a record of the last valid state for a data base.

If your program is a long-running one, you can reduce the amount of backout that might be necessary by taking checkpoints. When a logically complete set of updates has been completed, and your program is about to begin another set of updates, take a checkpoint by issuing the CHECKPOINT command. This signals the Backout utility to stop at this point.

The formats of the DL/I CHECKPOINT command are:

```
EXECUTE DLI CHECKPOINT ID(CHKPID);
```

and

```
EXECUTE DLI CHECKPOINT ID('CHKP0007');
```

where CHKPID is the name of an eight-byte character string that uniquely identifies this checkpoint. Alternatively, the CHECKPOINT ID can be coded in the commands, enclosed in single quotes, as in the second format.

Batch Checkpointing

Checkpointing of batch programs requires only that DL/I logging is active and that DL/I checkpoint commands are coded in your programs.

Online Syncpointing

For syncpointing of online transactions, CICS/VS journaling must be active. The Dynamic Transaction Backout (DTB) program must be in the system and DTB=YES must be specified for CSDB (DLZMPC00) and CSDC (DLZBPC00) in the Program Control Table (PCT). DL/I checkpoint commands coded in the transactions will automatically generate CICS/VS syncpoints. CICS/VS syncpoint commands or macros may also be used, but they disconnect the transaction from DL/I and require rescheduling of the PSB.

MPS Batch Checkpointing

Checkpointing should be used in a MPS batch program in conjunction with the restart capability. The restart facility is initiated by using DLR instead of DLI as the first parameter in the parameter statement when the job is started or restarted.

In addition to specifying the DLR function code, a VSE CHKPT macro, or high-level programming language equivalent, must be coded before each DL/I CHKP in the application program. Either the DL/I call interface or High-Level Programming Interface (HLPI) may be used to invoke the DL/I CHKP. No other DL/I commands may be issued between the VSE CHKPT and DL/I CHKP command or an error condition will result. It is recommended that the VSE CHKPT be coded before the DL/I CHKP command. Certain resources in the CICS/VS online partition are required:

- CICS/VS journaling must be active. JOUROPT=(CRUCIAL,INPUT) must also be specified for the system log in the Journal Control Table (JCT) in order to support emergency restart.
- The Keypoint Program (KPP) must be in the system and START=WARM or TSP=(YES/xx,WARM) must be specified in the System Initialization Table (SIT) in order to allow warm shut-downs and bring-ups of Temporary Storage.
- The Dynamic Transaction Backout Program (DTB) must be in the system and DTB=YES must be specified for CSDB (DLZMPC00) and CSDC (DLZBPC00) in the Program Control Table (PCT).
- The Temporary Storage Program (TSP) must be in the system and the temporary storage queue name “DLZTSQ00” must be specified as TYPE=RECOVERY in the Temporary Storage Table (TST).
- Sufficient space must exist in the Temporary Storage data set (DFHTEMP) for the temporary storage queue (TSQ) used by MPS Restart. Note that temporary storage space is also required by Dynamic Transaction Backout (DTB) for each active BPC in the online partition. If sufficient temporary storage space is not defined, PCBs and other tasks may go into a wait state. Care should be taken to avoid this situation by defining enough space.
- In order to use transaction CSDP to purge the TSQ used by MPS Restart, the following entries are required in CICS/VS tables:

```
PCT:  DFHPCT TYPE=ENTRY, PROGRAM=DLZMPUR0, TRANSID=CSDP, CLASS=LONG
PPT:  DFHPPT TYPE=ENTRY, PROGRAM=DLZMPUR0, RES=PGOUT
```

An example of the job control statements necessary for an MPS batch job using MPS Restart is given below. Included in the example are statements which assign a tape to contain checkpoint records written by VSE CHKPT.

```
// JOB MPSRSTRT
// MTC REW,280
// ASSGN SYS100,280
// EXEC DLZMP100,SIZE=250K
DLR,RSTRTPGM,RSTRTPSB
/*
/&
```

The invocation of VSE CHKPT will be different depending on the programming language being used. Assembler language programs use the VSE CHKPT macro directly, as shown in the following example. Note that the example follows the recommended procedure of using the VSE checkpoint ID (returned by the CHKPT macro) as the DL/I checkpoint ID.

```
CHPPT1   DS    0H                    COMBINED CHECKPOINT #1
         LA    R5,RESUME1            GET RESUME ADDRESS
         CHKPT SYS100,RSTRTR         ISSUE A VSE CHKPT - SYS100
*                                    IS TAPE, RSTRTR IS ADDRESS
*                                    OF RESTART ROUTINE
RESUME1  DS    0H                    RESUME PROCESSING AFTER RESTART
         LTR   R0,R0                 CHECK RETURN CODE
         BZ    ERROR                 BRANCH IF ERROR
         ST    R0,CHKPTID            USE VSE CHKPT ID ON DL/I CHKP
         OI    CHKPTID+3,X'F0'       MAKE LAST DIGIT PRINTABLE
         LA    R1,CHKPPARM           GET ADDRESS OF CHKP PARM LIST
         CALL  ASMTDLI               ISSUE A DL/I CHKP
         BAL   R5,VERSTAT            VERIFY DL/I STATUS CODE
         LA    R1,GUPARM             GET ADDRESS OF GU PARM LIST
         CALL  ASMTDLI               REPOSITION DATA BASE
         .                           CONTINUE PROCESSING
         .
RSTRTR   DS    0H                    RESTART ROUTINE
         STXIT AB...                 REISSUE STXIT AND OTHER
         .                           RESTART PROCESSING
         .
         BR    R5                    RESUME PROCESSING
VERSTAT  DS    0H                    VERIFY DL/I STATUS CODE
         CLC   DBPCBSTC,BLANKS       IS STATUS CODE BLANKS?
         BER   R5                    YES - RETURN TO CALLER
         .                           NO - CHECK FOR POSSIBLE ERROR
ERROR    DS    0H                    VSE CHECKPOINT ERROR ROUTINE
         .
CHKPPARM DS    0F
         DC    A(CHKP)               FUNCTION
CHKPPCB  DS    A                     PCB ADDRESS
         DC    XL1'80'               LAST PARM INDICATOR
         DC    AL3(CHKPTID)          I/O AREA
GUPARM   DS    0F
         DC    A(GU)                 FUNCTION
GUPCB    DS    A                     PCB ADDRESS
         DC    A(IOAREA)             I/O AREA
         DC    XL1'80'               LAST PARM INDICATOR
         DC    AL3(SSA)              SSA
BLANKS   DC    CL2'  '               BLANKS FOR STATUS CODE CHECK
CHKP     DC    CL4'CHKP'             CHECKPOINT FUNCTION
GU       DC    CL4'GU  '             GET UNIQUE FUNCTION
CKPALIGN DS    0F                    CHKPTID FULLWORD ALIGNED FOR
*                                    STORE INSTRUCTION
```

```
CHKPTID  DC    CL8' '                  DL/I CHECKPOINT ID
IOAREA   DS    CL80                    I/O BUFFER
SSA      DS    0F                      SSA
```

The normal checkpoint messages are issued to SYSLOG as the result of taking a VSE CHKPT and a DL/I CHKP. However, this is not necessary for the function of MPS Restart.

The PL/I interface to VSE CHKPT consists of a call to the built-in function PLICKPT. An example is given below. Again, the VSE checkpoint ID is used as the checkpoint ID for DL/I.

```
DECLARE CHKPTID CHAR(8);
DECLARE RETCODE FIXED BIN(31);

CALL PLICKPT ('',CHKPTID,'SYS100,2400',RETCODE); /*ISSUE A VSE CHKPT*/
IF RETCODE = 4 THEN DO;'          /*VSE CHKPT ERROR*/
     PUT EDIT('ERROR DURING CHECKPOINT. RETCODE=',RETCODE)(A,F(2));
     STOP;                        /*ABNORMAL END*/
     END;
IF RETCODE = 4 THEN               /*VSE RESTART OCCURRED*/
     PUT EDIT('RESTART AT CHECKPOINT #',CHKPTID)(A);
EXEC DLI CHECKPOINT ID(CHKPTID); /*ISSUE A DL/I CHKP WITH VSE CHKPT ID*/
EXEC DLI GET UNIQUE SEGMENT(ROOT) /*REPOSITION DATA BASE*/
             INTO(IOAREA)
             WHERE(KEYFIELD = LASTKEY);
```

In COBOL programs, a VSE CHKPT is issued implicitly each time a file is processed a predetermined number of times. The number of times may be specified by using the RERUN statement. In order to invoke a VSE CHKPT as part of a combined checkpoint in COBOL, the following format is suggested:

```
ENVIRONMENT DIVISION
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CHKPT-MSGS ASSIGN TO SYS101-UR-3203-S.
I-O-CONTROL.
*TAKE A VSE CHKPT ON EVERY WRITE TO CHKPT-MSGS
    RERUN ON SYS100-UT-2400-S.
        EVERY 1 RECORDS OF CHKPT-MSGS.
DATA DIVISION.
FILE SECTION.
FD  CHKPT-MSGS
    LABEL RECORD OMITTED
    RECORDING IS F
    RECORD CONTAINS 72 CHARACTERS
    DATA RECORD IS MSG-LINE.
01  MSG-LINE.
    02  FILLER     PIC X.
    02  ALL-71     PICX(71).
WORKING STORAGE SECTION.
01  INITMSG        PIC X(72) VALUE ' READY FOR VSE CHECKPOINTS '
01  CHKPTMSG       PIC X(72) VALUE ' TAKING A VSE CHECKPOINT '
01  CHKPTID        PIC X(8)  VALUE 'DL/ICHKP'.
PROCEDURE DIVISION.
    OPEN OUTPUT CHKPT-MSGS.
* INITIALIZE CHECKPOINT MESSAGE FILE
    WRITE MSG-LINE FROM INITMSG AFTER POSITIONING 2.
```

```
    .
    .
* TAKE AN IMPLICIT VSE CHECKPOINT
    WRITE MSG-LINE FROM CHKPTMSG AFTER POSITIONING 2.
* TAKE A DL/I CHECKPOINT
    EXEC DLI CHECKPOINT
        ID(CHKPTID)
    END EXEC.
* REPOSITION THE DATA BASE
    EXEC DLI GET UNIQUE
        SEGMENT(ROOT)
        INTO(IOAREA)
        WHERE(KEYFIELD = LASTKEY)
        FIELDLENGTH(KEYLEN)
    END EXEC.
```

In this example, a print file is defined so that a VSE CHKPT is invoked each time a message is written to it. The initial message is necessary because COBOL does not start issuing VSE checkpoints until the write statement after the first write occurs. COBOL will also issue a VSE checkpoint when the print file is closed at the end of the program, but this additional checkpoint will not affect the function of MPS Restart.

Since the COBOL-VSE CHKPT interface is implicit and the VSE checkpoint ID is not available to the application program, it cannot be used as the checkpoint ID on the DL/I CHKP command. To avoid this problem, programmers may want to write their own assembler language subroutine to interface directly with the COBOL library subroutine ILBDCKP0, which actually issues the checkpoints. The linkage interface for this subroutine is described in the *DOS/VS COBOL Subroutine Library Logic*.

RPG II does not support an interface to VSE CHKPT. Users will have to write their own interface in another programming language (for example, assembler language) if the MPS Restart facility is to be used with RPG programs.

Backout

Backout procedures require you to either:

- Close your batch application log file and run backout with the DL/I Data Base Backout utility, or
- Restart CICS/VS in the online and MPS Batch environments.

Backout occurs automatically when an online transaction or MPS batch job fails, or when you restart online work after a system failure via CICS/VS emergency restart. Batch jobs must be backed out by running a backout utility. All backouts must be performed before other applications are run against the same files. A printout of the log file may be helpful in preparing for backout operations.

Data Base Backout Utility

The Data Base Backout utility is the DL/I facility for backing out batch jobs. It requires you to close your tape log (you must write EOF if the file did not). Disk logs must have been formatted for reading backward.

The Data Base Backout utility removes (backs out) the effects of any abnormally terminated user program that accessed data bases using DL/I calls. If the program was operating in a DL/I batch partition, all data base changes made by the program from the time it began processing until it terminated abnormally are backed out. If the program issued checkpoint calls, only the changes made since the last checkpoint are backed out. If the program was operating in a CICS/VS-DL/I teleprocessing partition, only the changes made from the last synchronization point or scheduling call until the loss of the online system are backed out. If the program terminated (either normally or abnormally) before the CICS/VS-DL/I online system terminated, its changes are not backed out by this utility. CICS/VS dynamic backout may have backed them out, however.

If the data bases were not closed by DL/I prior to the abnormal termination, the user must execute the VSAM access method services command VERIFY to update the VSAM catalog for each file. If this is not done, an error will occur when the DL/I system attempts to open the data bases.

A log file is created to reflect the backout history. The log file must be included in any subsequent forward recovery attempted for the data bases involved in the backout.

Input to the Data Base Backout utility consists of:

1. One PSB and one or more DMBs loaded from a core image library during DL/I initialization.
2. The input data base(s) with which the Data Base Backout utility is to execute.
3. The DL/I log file(s) for the DL/I job execution which abnormally terminated.

Output from the Data Base Backout utility consists of:

1. The data bases reflecting the status prior to the DL/I job execution which abnormally terminated.
2. The output DL/I log file reflecting the changes made to the data base during the Data Base Backout utility execution. This log file as well as the the input log must be used as input for any future recovery utility execution against the above data bases, unless the data base has been backed up since the backout was done.
3. Messages on the SYSLST and SYSLOG devices.

Dynamic Transaction Backout Utility

The Dynamic Transaction Backout utility is the CICS/VS facility for backout from online and MPS batch DL/I jobs.

If no user STXIT routine was specified, or if the STXIT routine determined that the ABEND is to continue, CICS/VS will invoke Dynamic Transaction Backout (DTB) as part of the abnormal task termination process, if specified for that task in the Program Control Table (PCT)

DTB is the backing out of the effects of a transaction that has abnormally terminated. The resources that have been specified as protected are restored to the state they would have been in had the transaction terminated at its last user-defined synchronization point, or had not run at all. The resources are thus restored to a well-defined self-consistent state.

A similar facility, called Transaction Backout, is invoked during emergency restart following a CICS/VS failure. Transaction Backout operates on all transactions that were active at the time of system failure, whereas DTB backs out only the failing transaction while CICS/VS continues to run normally.

You must define the CICS/VS tables and the control statements to startup CICS/VS to use the combined DL/I and CICS/VS transaction backout facility and dynamic transaction backout (DTB). Complete information on these tables and statements can be found in the *CICS/DOS/VS Installation and Operations Guide*.

Log Print Utility

If a printout of the log is desired before you attempt backout, the Log Print utility can provide it. This utility will print the DL/I log either formatted by data fields or as a hexadecimal and character dump. It will also print the CICS/VS log in dump format only. By using this utility, you can also create a copy of the log usable for backout, even if the log is on tape, was never closed, and has been rewound so that an EOF mark cannot be written.

The information printed by the Log Print utility may be used to determine which data bases to back out. The utility can be invoked after an abnormal termination or system incident occurs. At such times, it can be used to determine which PSBs have been left open and should be backed out.

The Log Print utility program runs as a VSE program and does not make any DL/I calls.

The Log Print utility can print the following types of log files:

- Standard labeled DL/I tape log files
- Unlabeled CICS/VS tape log files
- Standard labeled CICS/VS tape log files
- VSAM DL/I disk log files
- SAM CICS/VS disk log files.

Through an input control statement, you can request that all DL/I log records be printed, or that only those records associated with a specific PSB be printed. By specifying a start date and/or an end date, you can additionally request that only records within a given time range be printed. You can select one or two output formats, keyword or dump. The keyword format individually identifies each field within the record. The dump format shows the record contents in both hexadecimal and character format, 32 bytes per line.

You can optionally request the utility copy feature. If specified, the utility creates a new DL/I log tape by copying all log records from the old log file up to the first invalid record. The new log tape can then be used as input to the Backout utility.

You can also request that CICS/VS journal records be printed in dump format. Records may also be selectively printed by DBD name, by CICS/VS TASK ID, or by relative block number.

Input to the Log Print utility consists of:

1. The log files to be printed.
2. Control statements that specify the type of log files and which log records are to be printed. These control statements are read from SYSIPT.

Output of the Log Print utility consists of:

1. Log records on the SYSLST device.
2. Informational messages on the SYSLST device.
3. Error messages on the SYSLST and SYSLOG devices.
4. Optionally, a new DL/I log tape suitable for use by the Backout utility.

For more complete information refer to *DL/I DOS/VS Resource Definition and Utilities* or *DL/I DOS/VS Interactive Definition and Utilities*.

Forward Recovery

Forward recovery operates on all recovery logs in time sequence. It does this either directly, log-by-log, or by accumulating changes before applying them.

Two utilities are used in the forward recovery process:

- Change Accumulation utility
- Data Set Recovery utility.

Change Accumulation Utility

The Change Accumulation utility processes and accumulates DL/I and CICS/VS log files for more efficient input to the Forward Recovery utility. The output can be used by forward recovery only in conjunction with Data Set Image Copy backups.

The Data Base Change Accumulation utility provides the Data Base Data Set Recovery utility with a sequential file that contains only the information from the log files necessary for recovery. This is done by:

- Eliminating all log records which are not data base change records
- Eliminating all log records before a user-specified purge date
- Combining all log records that update the same data base segment
- Putting the log records in sequence.

The resulting records are grouped by data base and are sequenced by data base, data set, and relative byte address. This utility may be executed several times over a period of time in order to incorporate additional data base changes and delete changes which are no longer significant. The input to this utility may be one or more DL/I system log files and a previous data base accumulation change file, if one exists.

The use of a purge date is optional. You may specify one purge date for all data base log records that you are processing or for all data base log records that update a particular data base. This allows you to eliminate log records for activity predating the last backup of a data base.

The Data Base Data Set Recovery utility does not support a change accumulation file for recovery of a SHISAM data base. However, the Data Base Change Accumulation utility may be used to merge log files containing updates to SHISAM data bases. This is done by specifying the SHISAM data base name in a DB1 control card.

The input to the Data Base Change Accumulation utility consists of:

1. All log files since the last Data Base Data Set Image Copy utility execution or the last run of this utility. This includes log files created after execution of the Data Base Backout utility.
2. The previous data base change accumulation file (if any). This is the output from the last execution of this utility.
3. Control statements that specify any purge dates, how the data base records are to be processed, and rewind options for the log and change accumulation tapes.

Output from the Data Base Change Accumulation utility consists of some or all of the following:

1. A new change accumulation file, a sequential file containing the combined data base records for the data base (DB0 on a control card).
2. A data base log file that contains data base log records identified on a DB1 control statement. The purpose of the data base log file is to avoid reprocessing the original log files. The new data base log may be used in place of the original log for another run of this utility or as the log input to the Data Base Data Set Recovery utility. The log contains records that were neither purged nor accumulated, as specified by input control statements.
3. Informational messages on the SYSLST device.
4. Error messages on the SYSLST and SYSLOG devices.

Note: Different record formats are used for log (variable-blocked) and accumulation (spanned) files.

More information on the Change Accumulation utility may be obtained in *DL/I DOS/VS Resource Definition and Utilities* or *DL/I DOS/VS Interactive Resource Definition and Utilities*.

Data Set Recovery Utility

The Data Set Recovery utility accomplishes forward recovery by re-applying changes from the forward recovery log to a restored copy of the data base. The forward recovery logs may be used directly, or the change accumulation log may be used. This utility may also use the Data Set Image Copy backup output instead of the restored data base.

The Data Base Data Set Recovery utility achieves a high throughput by manipulating an individual file on a physical data replacement basis in a physical sequential fashion. The general operation consists of taking an image copy input,

from data set image copy or HISAM reorganization unload, merging the application data from the accumulated change input (not applicable to SHISAM data bases), and creating a new data base file. Finally, any DL/I system logs which are not included in the accumulated change input are applied to the data base. This last operation is accomplished in a random access fashion using facilities provided by the DL/I system. It is considerably faster to apply the changes in a sequential order from the accumulated change input, and therefore to your advantage to make use of this facility.

However, it is possible to restore the data base from a backup copy and use only the logs as input.

Note: HD indexed and HIDAM data bases must have their primary INDEX data bases recovered at the same time as the main data base to ensure integrity.

Input to the Data Base Data Set Recovery utility consists of:

1. Control statements read from the SYSIPT device.
2. A DMB and utility PSB loaded from a core image library. The DMB and PSB names are obtained by DL/I by expanding the DBD name specified in the parameter statement.
3. Input files in any one of the following combinations:
    a. Image copy input only (from Data Base Data Set Image Copy utility or HISAM Reorganization Unload utility). This option restores the data base without forward recovery.
    b. Image copy input and change accumulation input (from Data Base Change Accumulation utility - not supported for simple HISAM).
    c. Image copy input and log file.
    d. Image copy input, change accumulation input, and log file (not supported for simple HISAM).
    e. Log file only (may be used after the data base has been restored from a backup copy).

If image copy input is used, the user must execute the VSAM Access Method Services utility command DELETE to remove the data base data set name from the VSAM catalog and release the space allocated for it. He must then DEFINE the new file to VSAM to cause a definition of the new file to be recorded on the VSAM catalog.

Output from the Data Base Data Set Recovery utility consists of:

1. A recovered file.
2. Informational messages written to the SYSLST device.
3. Error messages written to the SYSLST and SYSLOG devices.

Note that if the HISAM unload output file is to be used as the input file to the Data Base Data Set Recovery utility, the data base must be reloaded immediately following the unload. This procedure ensures that the recovery program ignores any records on the DL/I log which were created prior to the unload/reload operation. This is necessary because the HISAM Reorganization Unload utility reorganizes the data base, and any log records created before the data base is

reloaded do not reflect an image of the data base as it appears after the data base has been reloaded by the Data Base Data Set Recovery utility.

For more information about the Data Set Recovery utility see *DL/I DOS/VS Resource Definition and Utilities* or *DL/I DOS/VS Interactive Resource Definition and Utilities*.

Restart

Once the data base has been recovered, your application jobs are restarted from the point of recovery. If you have not taken checkpoints, you will have to rerun your applications from the beginning. If you have taken checkpoints and have recovered to the last checkpoint, you will be able to restart from there. Online users will have to re-enter their transactions from the recovery point after the operator restarts CICS/VS.

MPS Batch

In order to restart an MPS batch job after a failure, the VSE checkpoint ID of the last successful combined checkpoint must first be obtained from the SYSLOG message issued by MPS Restart. If the MPS batch job itself failed, the message containing the correct checkpoint ID is issued at the time of failure. If the MPS batch job did not take a successful combined checkpoint before the failure occurred, a message will be issued indicating that the job should be started over from the beginning of the job step. If there was a system failure, a message containing the necessary information is issued when MPS is started again in the online partition, along with messages for all other MPS batch jobs which had issued a combined checkpoint before the failure occurred.

After the VSE checkpoint ID is obtained, it is specified as a parameter on the VSE RSTRT job control statement. The RSTRT statement is used instead of the EXEC statement when the job is resubmitted for execution.

Assume that the following job control statements were used to run an MPS batch job:

```
// JOB MPSRSTRT
// MTC REW,280
// ASSGN SYS100,280
// EXEC DLZMPI00,SIZE=250K
DLR,RSTRTPGM,RSTRTPSB
/*
/&
```

Then the statements in the example below will restart the job from checkpoint 0010:

```
// JOB MPSRSTRT
// MTC REW,280
// ASSGN SYS100,280
// RSTRT SYS100,0010
DLR,RSTSTPGM,RSTRTPSB
/*
/&
```

On a restart, parameter input is ignored by DL/I, since the parameters were already read and saved when the job first started. However, if the parameter input

statement was included on SYSIPT (vs. being entered from SYSLOG) when the job first started, it is important that one also be included when the job is restarted. This is because DL/I will attempt to position SYSIPT past the parameter input statement when the job is restarted.

Batch

In batch jobs (not MPS), restart consists of running the job over from the beginning. If the application program takes DL/I checkpoints, then input file records processed before the last successful checkpoint must be bypassed. Programs may be written to automatically bypass already-processed records. If not, you must use a utility or your own program to delete the records from the file. The Log Print utility (“Log Print Utility”) may be used to determine which records have been processed, if that information is not available from program output.

Online

CICS/VS will automatically restart failed transactions if dynamic transaction backout has occurred and RESTART=YES was coded in the Program Control Table for the transaction. Otherwise, transactions must be manually restarted from the beginning in the same manner as they were originally restarted. All considerations applicable to batch program restart (above) apply to manually restarted CICS/VS transactions.

Chapter 4. Recovery Guide

This chapter consists of two sections. The first provides a description of the steps used in normal recovery, with flowcharts as a procedural guide for when to use each step. The second discusses possible actions for recovering from a failure which occurs during the recovery process itself.

Normal Recovery

Normal recovery procedures will depend upon the computing environment and the type of preparation that has been made for recovery from failure. For each of the three basic environments - batch, MPS batch, and online - the corresponding chart should be referenced. The flowcharts direct the recovery by determining, at a number of decision points, the preparations that have been made, and by providing appropriate action steps.

Consistent with flowcharting standards, decision points are identified by diamond-shaped blocks, and subsequent action items are identified within rectangles.

After a failure, use the chart decision points to plot a recovery path based on logs available, the type of backups made, and other preparation procedures implemented, as well as whether the data base files are physically intact (readable).

Follow the prescribed actions in the rectangular boxes to enact recovery.

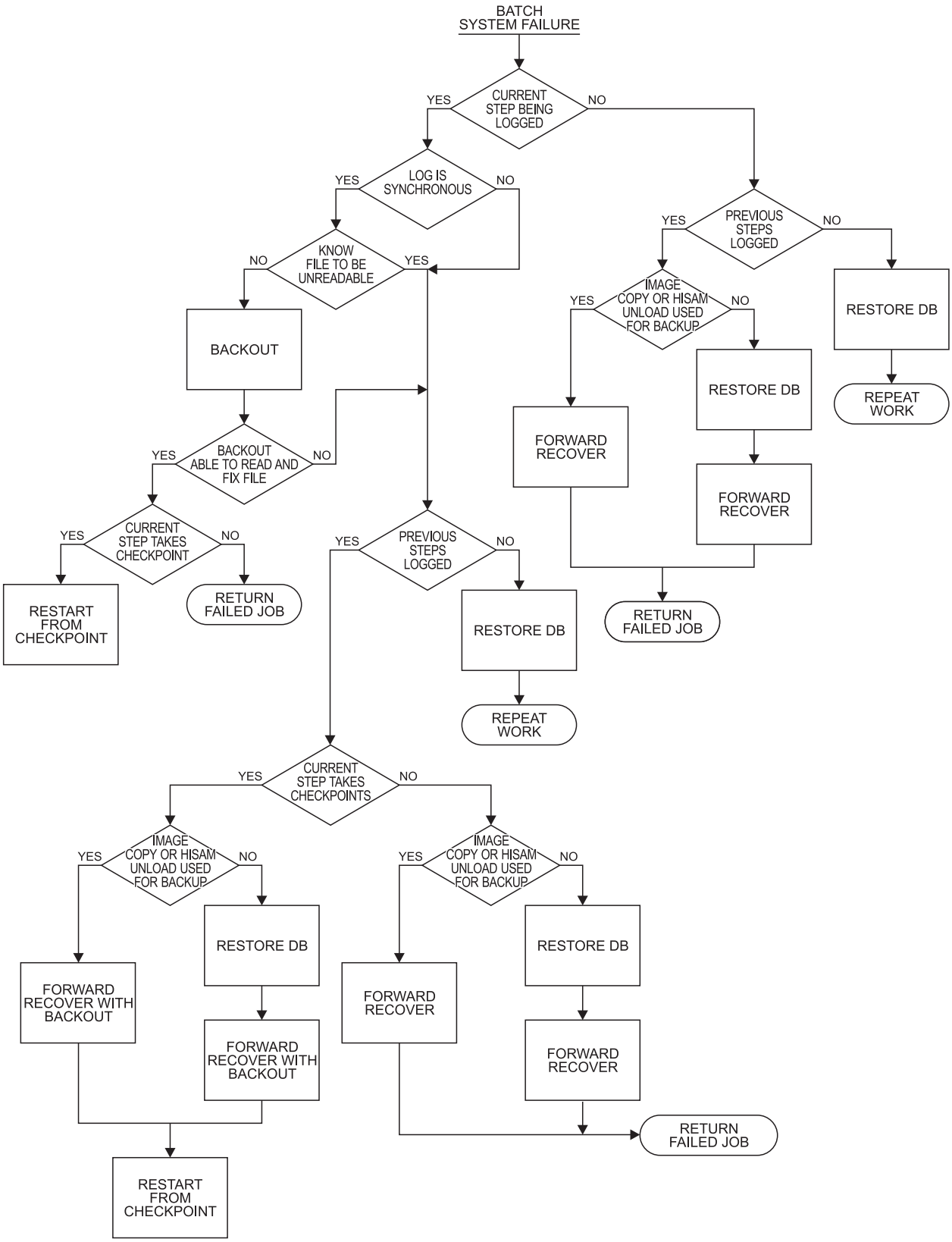

Figure 5. Recovery Decision Chart for Batch System Failure

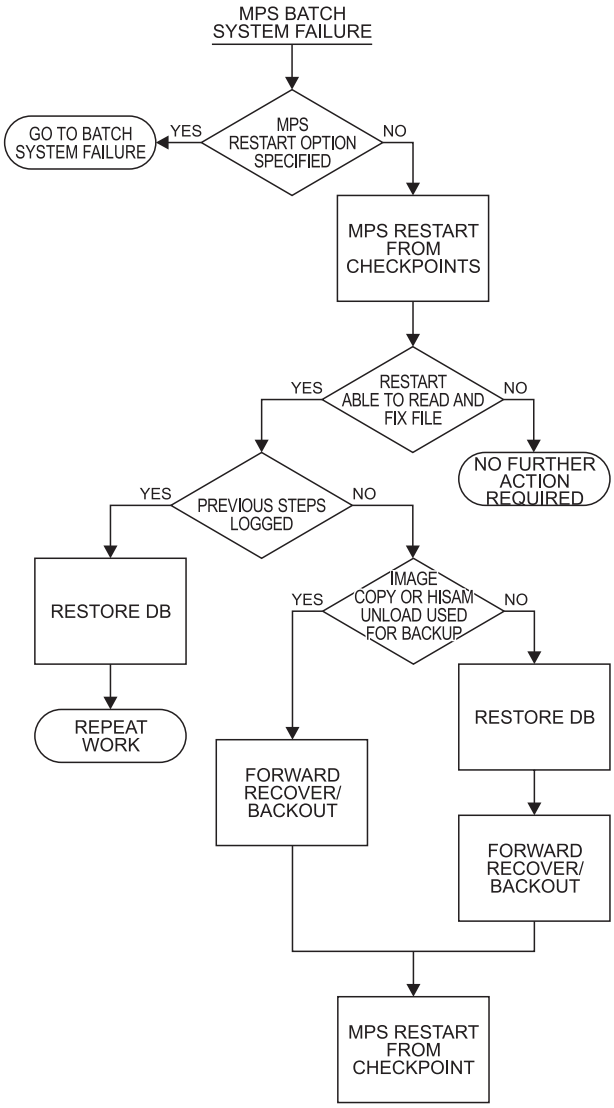

Figure 6. Recovery Decision Chart for MPS Batch System Failure

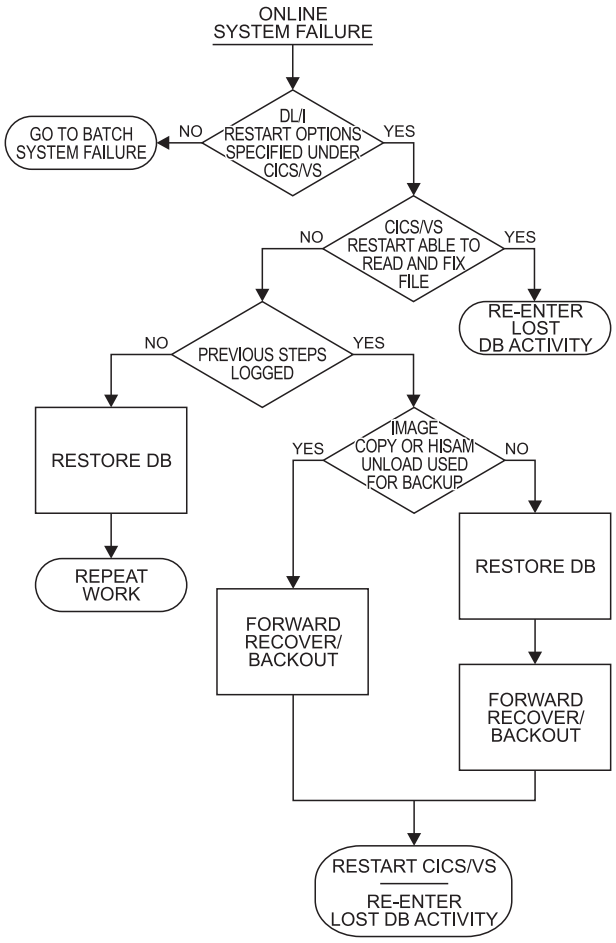

Figure 7. Recovery Decision Chart for Online System Failures

Restore Data Base From Backup Copy

When you restore your data base from a backup copy or copies:

- You must restore all related data bases
- You must use the restore utility which corresponds to the backup utility; that is, if you backed up your data base with HISAM Reorganization Unload then you must restore with HISAM Reorganization Reload; or if you used Data Set Image Copy for backup then you must use the Data Set Recovery utility.

HD Reorganization Unload Backup

If your backup consists of the output of HD Reorganization Unload, you must repeat all the steps you took to reload your data base during reorganization. Rerun the job or jobs you ran for reorganization, omitting the unload step.

HISAM Reorganization Unload Backup

If your backup was taken with HISAM Reorganization Unload, you must use the HISAM Reorganization Reload to restore it. Rerun the job or jobs you ran when you reorganized the data base, omitting the unload step.

Data Set Image Copy Backup

Restoring a data base from a Data Set Image Copy backup requires using the Data Set Recovery utility. The following steps are required:

1. Execute the VSAM Access Methods Services utility command DELETE to remove the names of the data base data set(s) and release allocated space.
2. Execute the VSAM Access Methods Services utility command DEFINE to record the name(s) of data sets to be recovered.
3. Run the Data Set Recovery utility, using the Data Set Image Copy backup copy as input.

Fast Copy Disk/Copy Disk Volume Backup

If you backed up your data base to a removable disk pack using Fast Copy Disk or Copy Disk Volume, you can simply mount the backup copy. If you dumped to a tape, you must use the same utility to reload your disk. Procedures may be found in *VSE Advanced Functions: Utilities*, for Fast Copy Disk, and in *VSE/Fast Copy Data Set Installation Reference*, for Copy Disk Volume.

Backout Incomplete LUWs

The Data Base Backout utility removes (backs out) the effects of any abnormally terminated program that accessed data bases using DL/I calls. Since programs may be divided into logical units of work (LUWs), a backout is performed only on those LUWs which have not completed. The following paragraphs describe the procedures necessary to prepare and execute a backout.

Step 1: If the data bases were not closed by DL/I prior to the abnormal termination, the VSAM Access Method Services command VERIFY must be executed to update the VSAM catalog for each file. If this is not done, an error will occur when the DL/I system attempts to open the data bases.

Step 2: If the log files are on tape and still open, an end-of-file (EOF) mark must be appended. This is done by closing the tape if not already rewound; or otherwise by using the Log Print utility to copy the log. The Log Print utility appends an EOF when no additional valid records can be copied.

Step 3: Now the Backout utility can be executed. Input for the utility includes the current Dl/I log file; the input data base(s), with which the utility is to execute; one PSB, which must be the one used by the failed application program; and one or more DMBs which will be loaded from a core image library during DL/I initialization.

Forward Recover Previously Successful Job Steps

A forward recovery procedure is designed to quickly recover data files that have become unreadable. Forward recovery can work from a restored data base or directly from the Data Set Image Copy or HISAM Reorganization Unload/Reload utility output files. If these latter utilities' outputs are not available, the data base files must be restored and used as input to the recovery utility.

Forward recovery operations use the log files or change accumulation file to reapply changes that had been made to the data base by successfully completed jobs. Jobs that were executing when the file failure occurred must be restarted.

From Restored Data Base

This section outlines the steps required to perform a forward recovery after a data base has been restored.

Step 1: Assemble all log files since the last backup. Do not include the log from the job executing at the time of system failure. These will be applied to the restored data base in chronological order. Change accumulation files cannot be used in forward recovery using a restored data base.

Step 2: Execute the VSAM Access Method Services utility command DELETE to remove the data base set name from the VSAM catalog and release the space allocated for it. DEFINE the new file to VSAM to cause a definition of the new file to be recorded on the VSAM catalog.

Step 3: Execute forward recovery using the Data Set Recovery utility. Input to the utility includes: appropriate control statements; a DMB and utility PSB; the restored data base; and the log files.

From Data Set Image Copy or HISAM Reorganization Unload Backup

This section provides a step-by-step procedure for performing a forward recovery by inputting directly to the Data Set Recovery utility without performing a restore of the data base files. The recovery utility will accept a data base backup which has been created by the DB Data Set Image Copy utility. It will also accept a backup created by the HISAM Reorganization Unload utility, but only if the data base was reorganized at the time of the backup by HISAM Reorganization Reload. Forward recovery using a HISAM Reorganization Unload backup will not work with change accumulation.

Step 1: Assemble all log files since the last backup. Do not include the log from the job executing at the time of system failure. Log files must be applied in the chronological order in which they were made.

Step 2: (Data Set Image Copy Backup Only):

Optionally, the log files can be grouped together by executing the DB Change Accumulation utility. It is considerably faster to apply the changes in a sequential order from accumulated change input, and therefore it is advantageous to make use of this facility. Input to the DB Change Accumulation utility consists of: all log files since the last backup or last run of the Accumulation utility (whichever is later); the previous change accumulation file (if applicable); and control statements that specify any purge dates, how the data base records are to be processed, and rewind options for the log and accumulation tapes. A new change accumulation file is output, along with an updated log file to be used as the log input for the DB Data Set Recovery utility.

Step 3: Execute the VSAM Access Method Services utility command DELETE to remove the data base set name from the VSAM catalog and release the space allocated for it. DEFINE the new file to VSAM to cause a definition of the new file to be recorded on the VSAM catalog.

Step 4: Implement a forward recovery on the backup copy by merging the data from the accumulated change input (if any), logs not input to the Change Accumulation utility (if any), and the backup copy of the data base into a new data base file using the DB Data Set Recovery utility. Input to the Data Set Recovery utility includes: appropriate control statements from the SYSIPT device; a DMB and utility PSB loaded from a core image

library; an image copy file (from Data Set Image Copy utility only); the change accumulation file; and optionally, any logs not included in the change accumulation. Do this for each data base file to be recovered.

Forward Recover to Failure Point and Back Out

If a file failure occurs during application processing, both forward recovery and backout can be used to properly recover the data base. Forward recovery can work from a restored data base or directly from the Data Set Image Copy or HISAM Reorganization Unload utility output files. If these latter utilities' outputs are not available, the data base files must be restored and used as input to the Data Set Recovery utility.

Forward recovery operations use the log files or change accumulation files to reapply all changes that had been made to the data base. (Change accumulation cannot be used with restored data bases or with a HISAM Reorganization Unload output file.) The Backout utility is then run against the current log to reverse changes made by LUWs that were interrupted by the failure.

From Restored Data Base

This section describes the steps necessary to perform a forward recovery and backout after a data base has been restored.

Step 1: If the data bases were not closed by DL/I prior to the abnormal termination, the VSAM access method services command, VERIFY, must be executed to update the VSAM catalog for each file. If this is not done, an error will occur when the DL/I system attempts to open the data bases.

Step 2: If the log files are on tape and still open, an end-of-file (EOF) mark must be appended. This is done by closing the tape if not already rewound; or otherwise by using the Log Print utility to copy the log. The Log Print utility appends an EOF when no additional valid records can be copied.

Step 3: Assemble all log files since the last backup. These will be individually applied to the restored data base in chronological order. Include the log for the current job step that just failed.

Step 4: Execute the VSAM Access Method Services utility command DELETE to remove the data base set name from the VSAM catalog and release the space allocated for it. DEFINE the new file to VSAM to cause a definition of the new file to be recorded on the VSAM catalog.

Step 5: Execute forward recovery using the Data Set Recovery utility. Input to the utility includes: appropriate control statements; a DMB and utility PSB; the restored data base; and the log files.

Step 6: Now the Backout utility can be executed. Input for the utility includes the current DL/I log file, the input data base(s), with which the utility is to execute; one PSB, which must be the one used by the failed application program; and one or more DMBs which will be loaded from a core image library during DL/I initialization.

From Data Set Image Copy or HISAM Reorganization Unload Backup

This section provides a step-by-step procedure for performing a forward recovery by inputting a backup directly to the recovery utility, and for following with a backout. The Data Set Recovery and Backout utilities will be used for these respective tasks. The recovery utility will accept directly the output of the data set image copy or HISAM Reorganization Unload utility. The HISAM output, however, can only be used directly if the data base was reloaded with the HISAM Reorganization Reload utility at the time the backup was made. Forward recovery with a HISAM Reorganization Unload backup will not work with change accumulation.

Step 1: If the data bases were not closed by DL/I prior to the abnormal termination, the VSAM access method services command, VERIFY, must be executed to update the VSAM catalog for each file. If this is not done, an error will occur when the DL/I system attempts to open the data bases.

Step 2: If the log files are on tape and still open, an end-of-file (EOF) mark must be appended. This is done by closing the tape if not already rewound; or otherwise by using the Log Print utility to copy the log. The Log Print utility appends an EOF when no additional valid records can be copied.

Step 3: Assemble all log files since the last backup. Log files must be applied in the chronological order in which they were made. Include the log from the job step which just failed.

Step 4: (Data Set Image Copy Backup Only):

Optionally, the log files can be grouped together by executing the DB Change Accumulation utility. It is considerably faster to apply the changes in a sequential order from accumulated change input, and therefore it is advantageous to make use of this facility.

Input to the DB Change Accumulation utility consists of: all log files since the last backup or last run of the Accumulation utility (whichever is later); the previous change accumulation file (if applicable); and control statements that specify any purge dates, how the data base records are to be processed, and rewind options for the log and accumulation tapes. A new change accumulation file is output, along with an updated log file to be used as the log input for the DB Data Set Recovery utility.

Step 5: Execute the VSAM Access Method Services utility command DELETE to remove the data base set name from the VSAM catalog and release the space allocated for it. DEFINE the new file to VSAM and cause a definition of the new file to be recorded on the VSAM catalog.

Step 6: Implement a forward recovery on the backup copy by merging the data from the accumulated change input (if any), logs not input to the Change Accumulation utility (if any), and the backup copy of the data base into a new data base file using the DB Data Set Recovery utility.

Input to the Data Set Recovery utility includes: appropriate control statements from the SYSIPT device; a DMB and utility PSB loaded from a core image library; an image copy file (from Data Set Image Copy utility); the change accumulation file; and optionally, any logs not included in the change accumulation. Do this for each data base file to be recovered.

Step 7: Now the Backout utility can be executed. Input for the utility includes the DL/I log file for the job that just failed, the input data base(s) with which the utility is to execute; one PSB, which must be the one used by the failed application program; and one or more DMBs which will be loaded from a core image library during DL/I initialization.

Restart Program From Checkpoint

If logging is used, batch applications programs may issue checkpoint calls to establish restart points in the even of abnormal termination. The purpose of the checkpoint call is to limit the amount of work that will be backed out and that must be rerun in the event of a failure, which can be very valuable for long running applications.

Restarting a batch processing applications program, conceptually, is a very simple process. It may be restarted without change to the program or the JCL. The difficulty, however, is in determining which records on your input files were correctly processed prior to failure and in revising (removing records already processed) those input files before rerunning the program.

MPS batch programs can take advantage of the MPS Restart facility to minimize your effort. With a few changes in your MPS Restart JCL, you can reinitiate your job and continue processing.

Batch Processing

Restarting at a checkpoint in a batch process involves identification of records in your input files which have not been processed and initiating your program to process those records. It is assumed that the batch applications program was maintaining a log and issuing DL/I checkpoint calls, and backout (or forward recovery/backout) has been performed. In other words, your DL/I data bases are in a valid state up to the last checkpoint and you have saved information which allows you to identify the records in your input files which have already been processed.

There are three options for restarting a batch program from a checkpoint, depending upon the capabilities built into the applications program:

- The applications program has built-in routines to access log files and determine the starting point in the input files for continued processing. Run your applications program, indicating to it that this is a program restart rather than a normal run. (Refer to individual applications program documentation to do this.)
- The applications program has saved information regarding how many and which records have been processed.
    1. Use this information to remove processed records from the input files.
    2. Initiate program normally.
- The applications program has no build-in mechanisms and has saved no information.
    1. Run the Log Print utility to determine which input records have been applied to your DL/I data bases.
    2. Recreate input files, removing already processed records.
    3. Initiate program normally.

MPS Batch

Restarting at a checkpoint in an MPS batch process is a very simple and straight forward process which involves reinitialization of your job with some changes to JCL. It is assumed that CICS/VS is maintaining a journal with DL/I records. The MPS batch applications program has issued VSE and DL/I checkpoint calls, and the MPS Restart option has been specified.

The steps for restarting a MPS batch program from a checkpoint are:

1. Identify the checkpoint from messages issued by DL/I before the failure or at system bring-up time.
2. Execute the MPS Restart procedure for your job, specifying the checkpoint ID.

The MPS Restart procedure will automatically back out your DL/I data bases to the last checkpoint if it is able to read and fix the data bases. If it cannot, then you must perform forward recovery/backout before using MPS Restart. If you specify the wrong checkpoint, error message DLZ126I will be issued. It will indicate the correct checkpoint ID and will allow restart with this checkpoint. You will be provided an option to cancel the restart job or to continue it using the correct checkpoint ID.

When Recovery Fails

The preceding sections discuss recovery from failure of applications or failure of the system while applications are in progress. This section addresses what you should do in the event of a recovery failure.

Failure During Data Base Restore

There are three options for recovery from a data base restore failure which depend upon system conditions at the time of failure:

- The HDAM Reorganization Reload utility can be treated as an applications program with checkpoints.
- If the DL/I data base and backup are readable, then repeat the restore job.
- If the backup is unreadable, then
    1. Use an older backup or another copy of the backup if it is available and repeat the restore job.
    2. If an older backup was used, it will be necessary to forward recover work performed since that backup was taken, or to repeat the work if logs are not available. Repeated work must be exact (all activities in the original order) if you want to use forward recovery on the data base after you restore it to the state of the original backup.

Failure During Backout

There are three backout methods which must be treated separately in describing how to recover from a backout failure:

- DL/I Backout
- CICS/VS Dynamic Transaction Backout
- CICS/VS Transaction Backout.

DL/I Backout Failure

- If backout failed because of a system failure during backout processing, then redo DL/I backout.
- If backout failed because of a damaged log or data base, then refer to the batch recovery procedures described in “Batch Procedures for Recovery From a Backout Failure.”

CICS/VS Dynamic Transaction Backout Failure

- If backout failed because of a software or system failure during backout processing, then
  1. Bring CICS/VS down if necessary
  2. Perform a CICS/VS Emergency Restart.
- If backout failed because of a damaged log file or data base, then
  1. Bring CICS/VS down
  2. Refer to the batch recovery procedures described in “Batch Procedures for Recovery From a Backout Failure”
  3. Cold Start CICS/VS.

CICS/VS Transaction Backout Failure During Emergency Restart

- If backout failed because of a system failure during backout processing, then redo Transaction Backout (repeat Emergency Restart).
- If backout failed because of a damaged log file or data base, then
  1. Bring CICS/VS down
  2. Refer to the batch recovery procedures described in “Batch Procedures for Recovery From a Backout Failure”
  3. Cold Start CICS/VS.

Batch Procedures for Recovery From a Backout Failure

- If your DL/I data base is unreadable or unfixable and recovery logs are available, then
  1. Restore data base from backup file
  2. Do forward recovery/backout.
- If your DL/I data base is unreadable or unfixable and forward recovery logs are not available for all or some sessions, then
  1. Restore your data base from a backup
  2. Forward recover up to the log that is unavailable
  3. Repeat work for the session for which the log is unavailable
  4. Continue with forward recovery or forward recovery/backout.
- If your backout log is unreadable but forward recovery logs are available, then
  1. Restore your data base from a backup
  2. Forward recover up to the failed session

3. If your program takes checkpoints, use log print to copy the log from the failed session up to the point it is unreadable, forward recover as far as possible, then back out to last completed checkpoint
4. Restart failed application.

- If your backout log is unreadable and forward recovery logs are unavailable, then
  1. Restore your data base from a backup
  2. Redo all work.

Failure During Forward Recovery

Forward recovery can fail because of a system failure during recovery which interrupts recovery processing; because of a faulty backup file; because of a damaged, restored data base; or because of a damaged log file.

- If recovery fails because of a forward recovery process interruption or because the restored data base is damaged, then
  1. Re-restore the data base.
  2. Repeat the forward recovery job.
- If recovery fails due to a faulty backup file, then
  1. Restore the data base from an earlier backup or from a second copy of the backup file, if available.
  2. Assuming recovery logs are available, run forward recovery from that point, or
  3. If a complete set of recovery logs is not available, then forward recover up to the missing log, repeat work for the session which has no log, and continue forward recovery.
- If recovery failed because of a damaged log file, then
  1. Re-restore the data base.
  2. Forward recover up to the log that is damaged
  3. Repeat work for the session which has the bad log
  4. Continue forward recovery.

In order to continue to forward recovery after repeating work for a missing or damaged log, you must repeat the work exactly (as by rerunning a batch job). Changing the order (as when reconstructing online activity) will prevent subsequent forward recovery from working.

Chapter 5. Summary of Job Control Language Statements for DL/I Utilities

Most users will use VSE's interactive facility to generate the job streams required for the utilities discussed in this manual. Interactive procedures are described in the *DL/I DOS/VS Interactive Resource Definition and Utilities* manual. For those users who wish to refer directly to JCL requirements, this appendix provides a summary of JCL for DL/I utilities referred to in this manual, including representative examples. More complete information on utility options available may be found in the *DL/I DOS/VS Resource Definition and Utilities* manual. Information on additional JCL parameter options may be found in *VSE/Advanced Functions System Control Statements*.

HD Reorganization Unload

The following job control statements are required when using the HD Reorganization Unload utility:

`// DLBL`	`filename`	This statement defines the symbolic name of the data base file to be unloaded. One statement must be present for each file that appears in the DBD describing this data base. One statement must also be present for each file that appears in DBDs of all other data bases that are related by logical relationships or secondary indexes. For HD randomized and indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements (if any). For HDAM, HIDAM (and primary INDEX), and SHISAM data bases, use the filename specified in the DD1 operand of DATASET statements. For HISAM data bases use the filenames specified in the DD1 and OVFLW operands of DATASET statements.

`// ASSGN`
 	
`[// TLBL(Tape)`
`or`
`[// DLBL(DASD)`
`[// EXTENT`
	`SYS010, {cuu}`
	`{IGN}`
	
	`RESTART ]`
	
	`RESTART ]`
	`extent data ]`
		These statements define the input dump file if a utility restart is to be attempted. The file that defines the input dump file contains the partially dumped data base and is the same file as the HDUNLD1 (or HDUNLD2) from the previously interrupted unload utility. It may be DASD, or tape with standard labels. RESTART is the name of the input file. If no restart is to be performed, IGN must be specified, but the other statements in this group may then be omitted.

`// ASSGN`
 	
`// TLBL`
`or`
`// DLBL`
`// EXTENT`
	`SYS011,cuu`
	
	`HDUNLD1`
	
	`HDUNLD1`
	`extent data`
		These statements define the first copy of the output file which contains the dumped data base. It may be DASD, or tape with standard labels. HDUNLD1 is the symbolic name of the output file(DTF).

// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS012, {cuu} {IGN} HDUNLD2] HDUNLD2] extent data]	These statements define the second copy of the output file if one is desired. The file may be DASD, or tape with standard labels. HDUNLD2 is the symbolic name of the output file. If a second copy is not desired, IGN must be specified but the other statements in this group may then be omitted.
// EXEC	DLZRRC00, SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

The HD Reorganization Unload utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered either via SYSIPT or SYSLOG:

```
{ULU},DLZURGU0, dbdname
{ULR}        [,{buf}][,HDBFR=][,HSBFR=][,TRACE=]
               {  1.}
```

dbdname
is the name of the DBD that contains the data base to be unloaded.

The positional parameter 'buf' and the keyword parameters HDBFR, HSBFR, and TRACE are optional parameters that may also be entered in the parameter statement. These parameters have the same usage as for application programs.

The following control statements are used with the HD Reorganization Unload utility:

```
[REW=option]
```

The REW control statement invokes tape rewind options. Its use is optional.

If used, columns 1-4 must contain REW= and column 5 must contain one of these letters to indicate the tape option you want:

N - means no rewind,
R - means rewind only, and
U - means rewind and unload.

No other columns are used. Put the completed statement after the DL/I parameter statement in your job stream.

If this control statement is not used, tapes will automatically rewind and unload at end-of-job.

```
  [CHKPT={NO    }]
         {xxxxxx}
```

The CHKPT control statement invokes the checkpoint option. Its use is optional. If used, columns 1-6 must contain CHKPT= and must be immediately followed by one of these options. Either,

- NO must be specified in columns 7-8 to indicate that no checkpoints are to be taken, or
- A 1-6 digit decimal number must be specified in columns 7-12 to indicate the number of segments that are to be processed before a checkpoint is taken.

No other columns are used. Put the completed statement after the DL/I parameter statement in your job stream.

If this control statement is not used, checkpoints are automatically taken at intervals of approximately 5000 segments.

Example 1: This example shows the unloading of an HD indexed data base. Two DLBL statements are provided. The first one is for the filename defined in the DATASET statement DD1 operand and the second is for the filename defined in the ACCESS statement REF operand. A restart is not requested. Two tape output copies are requested. Checkpoint records are taken at intervals of 3000 segments.

```
// JOB    HDUNLD1
// DLBL   HDDATAB,'HD DATA BASE',,VSAM
// DLBL   HDPRINX,'HD PRIMARY INDEX',,VSAM
// ASSGN  SYS010,IGN
// ASSGN  SYS011,180
// TLBL   HDUNLD1,'HDUNLD1'
// ASSGN  SYS012,181
// TLBL   HDUNLD2,'HDUNLD2'
// EXEC   DLZRRC00,SIZE=300K
ULU,DLZURGU0,DLZDBD3
CHKPT=3000
/*
/&
```

Example 2: This example shows the unload of an HDAM data base using one output copy. A restart is not requested. One DLBL statement is provided for the HDAM data base. The output tape does not unload and no checkpoints are taken.

```
// JOB    HDUNLD2
// DLBL   HDAMDB,'HDAM DATA BASE',,VSAM
// ASSGN  SYS010,IGN
// ASSGN  SYS011,180
// TLBL   HDUNLD1,'HDUNLOAD'
// ASSGN  SYS012,IGN
// EXEC   DLZRRC00,SIZE=300K
ULU,DLZURGU0,DLZDBD4
REW=R
CHKPT=NO
/*
/&
```

Example 3: This example shows a restart after an abnormal termination of the execution of Example 1. The console operator must enter the number of the last successful checkpoint record.

```
// JOB    HDUNLD3
// DLBL   HDDATAB,'HD DATA BASE',,VSAM
// DLBL   HDPRINX,'HD PRIMARY INDEX',,VSAM
// ASSGN  SYS010,180
// TLBL   RESTART,'HDUNLD1'
// ASSGN  SYS011,181
// TLBL   HDUNLD1,'HDUNLD1RST'
// ASSGN  SYS012,182
// TLBL   HDUNLD2,'HDUNLD2RST'
// EXEC   DLZRRC00,SIZE=300K
ULU,DLZURGU0,DLZDBD3
CHKPT=3000
/*
/&
```

HD Reorganization Reload

The following job control statements are required when using the HD Reorganization Reload utility.

// ASSGN
 	
// TLBL
or
// DLBL
// EXTENT
	SYS011,cuu
	
	HDUNLD1
	
	HDUNLD1
	extent data
		These statements define the input file containing the data base to be reloaded. This file was created by the HD Reorganization Unload utility and may be DASD, or tape with standard labels. HDUNLD1 is the symbolic name of the input file.

[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS013,cuu] WORKFIL] WORKFIL] extent data]	These statements define the output work file to be created if logical relationships or secondary indexes exist. It may be DASD, or tape with standard labels. WORKFIL is the symbolic name of the output file. These statements are not required if logical relationships do not exist.
[// ASSGN [// DLBL [// EXTENT	SYS012,cuu] CONTROL] extent data]	These statements define the input control file created by the Data Base Prereorganization utility. It must be DASD. CONTROL is the symbolic name of the input file. These statements are not required if logical relationships do not exist.
[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS010, {cuu}] {IGN} {UA } RSTFILE] RSTFILE] extent data]	These statements are used when doing a “reload restart.” They define the partially created output work file (WORKFIL), if any, that was being created during the initial reload. This work file is now used as input during restart. Assign it as SYS010 with a filename of RSTFILE. If no work file was created during initial reload, SYS010 must be assigned IGN or UA when doing the restart.

// DLBL	filename	This statement defines the symbolic name of the data base file to be reloaded. One statement must be present for each file that appears in the DBD describing this data base. For HD randomized and indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements (if any). For HDAM, HIDAM, (and primary INDEX), and SHISAM data bases, use the filename specified in the DD1 operand of DATASET statements. For HISAM data bases, use the filenames specified in the DD1 and OVFLW operands of DATASET statements. If secondary index relationships exist, DLBL statements for the secondary index data bases are also required when the secondary indexes are to be created automatically.
// EXEC	DLZRRC00 SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

The HD Reorganization Reload utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered via SYSIPT or SYSLOG.

```
ULU,DLZURGL0,dbdname
           [,{buf}][,HDBFR=][,HSBFR=][,TRACE=]
             {1  }
```

dbdname
is the name of the data base description that contains the data base to be reloaded.

The positional parameter ‘buf’ and the keyword parameters HDBFR, HSBFR, and TRACE are optional parameters that may also be entered in the parameter statement. These parameters have the same usage as for application programs.

The following control statements are used with the HD Reorganization Reload utility:

```
[REW=option]
```

This control statement invokes tape rewind options. Its use is optional. If used, columns 1-4 must contain REW= and column 5 must contain one of these letters to indicate the tape option you want:

N - means no rewind
R - means rewind only, and
U - means rewind and unload.

No other columns are used. Put the completed statement after the DL/I parameter statement in your job stream.

If this control statement is not used, tapes will automatically rewind and unload at end-of-job.

HD Reorganization Reload can use the checkpoint records created by the Reorganization Unload utility to restart the reload if it fails before completion. If a work file (for logical relationships or secondary indexes) was being created during the initial reload, the partially created work file should be submitted as input to the restarted job. It should be assigned as SYS010 with the filename of RSTFILE. The utility uses this file to create a complete work file on SYS013 as is normally done.

```
[BLDINDEX=NO]
```

This control statement indicates that secondary indexes are not to be created during reload. Instead, they will be created later using the logical relationship utilities. Use of this control statement is optional. If used, it must begin in column 1.

If the BLDINDEX=NO statement is not provided, the secondary indexes will automatically be created during reload, as long as DLBL statements are provided for them. Secondary indexes will not be created if no DLBL statements are provided.

Example 1: This example shows an HDAM reorganization reload operation.

```
// JOB    HDLOAD1
// ASSGN  SYS011,180
// TLBL   HDUNLD1,'HDAM REORG'
// ASSGN  SYS013,181                ]
// TLBL   WORKFIL,'HDAM WORK FILE'  ]  THESE STATEMENTS ARE
// ASSGN  SYS012,231                ]  REQUIRED FOR LOGICAL
// DLBL   CONTROL,'CONTROL FILE'    ]  RELATIONSHIPS
// EXTENT SYS012,232232             ]
// DLBL   HDAMDB,'HDAM DATA BASE',,VSAM
// EXEC   DLZRRC00,SIZE=300K
ULU,DLZURGL0,DLZDBD4
/*
/&
```

Example 2: This example shows a HIDAM reorganization reload operation. Note that in this example the HIDAM INDEX data base data set is also described.

```
// JOB    HDLOAD2
// ASSGN  SYS011,180
// TLBL   HDUNLD1,'HDUNLDIRST'
// ASSGN  SYS013,231
// DLBL   WORKFIL,'HDAM WORK FILE'  ]
// EXTENT SYS013,232232,1,0,400,5   ]  THESE STATEMENTS ARE
// ASSGN  SYS012,231                ]  REQUIRED FOR LOGICAL
// DLBL   CONTROL,'CONTROL FILE'    ]  RELATIONSHIPS
// EXTENT SYS012,232232             ]
// DLBL   HIDAMDB,'HIDAM DATA BASE',,VSAM
// DLBL   HIDAMDX,'HIDAM INDEX'
// EXEC   DLZRRC00,SIZE=300K
ULU,DLZURGL0,DLZDBD3
/*
/&
```

HISAM Reorganization Unload

The HISAM Reorganization Unload utility is executed as a standard VSE program and requires the following job control statements:

// DLBL	filename	This statement defines the symbolic name of the data base file to be unloaded. One statement must be present for each file that appears in the DBD describing this data base. For SHISAM data bases, use the filename specified in the DD1 operand of DATASET statements. For INDEX data bases, use the filename specified in either the DD1 operand of DATASET statements or the REF operand of ACCESS statements. For HISAM data bases, use the filenames specified in the DD1 and OVFLW operands of DATASET statements.

// ASSGN
 	
// TLBL
or
// DLBL
// EXTENT
	SYS011,cuu
	
	filename
	
	filename
	extent data
		These statements define the first copy of the reorganized output file. One is required for each data base (KSDS/ESDS pair) to be reorganized. The filename may be any name, but must appear in the associated utility control statement. The file may be on tape or DASD.

[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS012,cuu] filename] filename] extent data]	These statements are optional and are required only if the associated utility control statement requests two copies of the dump. The name must appear in the control statement. It may be tape or DASD.
// EXEC	DLZURUL0, SIZE=xxxK	HISAM Reorganization Unload utility module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

One control statement is required for the HISAM Reorganization Unload utility. Its form and content are described below. It is read from the SYSIPT device.

```
1234         13        22           31           40
```

```
  N
RxRdbdname   infile   outfile1   [outfile2]   [comments]
  U
```

Column	Description
1	This must be the character R, which defines this as a HISAM Reorganization Unload utility control statement.
2	This must be 1 or 2, depending on the number of copies required.
3	This invokes the tape rewind option: N means no rewind, R means rewind only, and U (or blank) means rewind and unload.
4	This must be the name of the DBD that includes the symbolic filenames of the KSDS/ESDS pair (only KSDS if SHISAM or INDEX) to be reorganized.
13	This must be the symbolic filename of the KSDS of the SHISAM, HISAM or INDEX data base to be reorganized. It must appear as the DD1 parameter of the DATASET statement or the REF parameter of ACCESS statements in the referenced DBD, and corresponding VSE DLBL statements must be provided.
22	This is the filename of the TLBL or DLBL statement for the primary output file (SYS011).
31	This is the filename of the TLBL or DLBL statement for the second copy of the reorganized output file (SYS012). This field must be blank if column 2 contains a 1.
40	Comments may be placed in columns 40 through 80.

Example 1: This example unloads a HISAM data base, creating one output copy. The KSDS input filename is DBH13A, and the ESDS is DBH13B. These names appear in the DBD named DLZDBD1. The output filename is DBOUT1.

```
// JOB     HISAMUN1
// DLBL    DBHI3A,'DBHI3A',,VSAM
// DLBL    DBHI3B,'DBHI3B',,VSAM
// ASSGN   SYS011,180
// TLBL    DBOUT1,'DLZDBD1REORG'
// EXEC    DLZURUL0,SIZE=100K
R1 DLZDBD1  DBHI3A   DBOUT1
/*
/&
```

Example 2: This example unloads SHISAM or INDEX data bases which consist of only a KSDS. Two copies of the data base are created.

```
// JOB     HISAMUN2
// DLBL    DBHI4A,'DBHI4A',,VSAM
// ASSGN   SYS012,181
// TLBL    DBOUT2,'DLZDBD2REORG'
// ASSGN   SYS011,180
// TLBL    DBOUT1,'DLZDBD2REORG'
// EXEC    DLZURUL0,SIZE=100K
R2 DLZDBD2  DBHI4A   DBOUT1   DBOUT2
/*
/&
```

HISAM Reorganization Reload

The HISAM Reorganization Reload utility is executed as a standard VSE program and requires the following job control statements:

// ASSGN
 	
// TLBL
or
// DLBL
// EXTENT
	SYS011,cuu
	
	filename
	
	filename
	extent data
		These statements define the input file that was created by the HISAM Reorganization Unload utility. The filename parameter may be any name but must appear in the associated utility control statement.

// DLBL	filename	This statement defines the symbolic name of the data base file to be reloaded. One statement must be present for each file that appears in the DBD describing this data base. If multiple data bases are to be reloaded during a single job execution, all files must be defined. For SHISAM data bases, use the filename specified in the DD1 operand of DATASET statements. For INDEX data bases, use the filename specified in either the DD1 operand of DATASET statements or the REF operand of ACCESS statements. For HISAM data bases, use the filenames specified in the DD1 and OVFLW operands of DATASET statements.
// EXEC	DLZURUL0, SIZE=xxxK	HISAM Reorganization Reload utility module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

One control statement is required for the HISAM Reorganization Reload utility. Its form and content are described below. It is read from the SYSIPT device.

```
  1 3                 22                      40
```

```
L N                 filename              [comments]
  R
  U
```

Column	Description
1	This must be the character L, which defines this as a HISAM Reorganization Reload utility control statement.
3	This invokes the tape rewind option: N means no rewind, R means rewind only, and U (or blank) means rewind and unload.
22	This is the filename of the TLBL or DLBL statement for the input file (SYS011) that contains the data base to be reloaded.
40	Comments may be placed in columns 40 through 80.

Example 1: This example reloads a HISAM data base consisting of a KSDS and an ESDS.

```
// JOB     HISAMLD1
// ASSGN   SYS011,180
// TLBL    DBRLDIN,'DLZDBDIREORG'
// DLBL    DBHI3A,'DBHI3A',,VSAM
// DLBL    DBHI3B,'DBHI3B',,VSAM
// EXEC    DLZURRL0,SIZE=100K
L                   DBRLDIN
/*
/&
```

Example 2: This example reloads SHISAM or INDEX data bases.

```
// JOB     HISAMLD2
// ASSGN   SYS011,180
// TLBL    HISAMIN,'DLZDBD2REORG'
// DLBL    DBHI4A,'DBHI4A',,VSAM
// EXEC    DLZURRL0,SIZE=100K
L                   HISAMIN
/*
/&
```

Data Base Prereorganization

The following job control statements are required for the Data Base Prereorganization utility.

// ASSGN
// DLBL
// EXTENT
	SYS012,cuu
	CONTROL
	extent data
		These statements define the output control file. This file is used as input to the Prefix Resolution utility, work file generator module, and Data Base Scan utility. It must be DASD. CONTROL is the symbolic name of the output file.

[// ASSGN	SYSPCH,cuu]	This statement is optional. SYSPCH is required only if an OPTIONS=PUNCH control statement is provided (see below). A data base scan list is written on SYSPCH.

// EXEC	DLZRRC00, SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

The Data Base Prereorganization utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered either via SYSIPT or SYSLOG:

```
ULU,DLZURPR0
```

The following control statements follow the parameter statement:

```
DBR=databasename1,databasename2,...
```

The DBR control statement names data bases that are being reorganized. One or more of these control statements may be provided. Each data base name must be right-padded with necessary blanks (that is, left-justified) to provide a total length of eight characters. A blank must follow the last entry on each control statement. Note that if a HIDAM or HD indexed data base is to be reorganized, only its DBD name should be listed on a DBR control statement and not the INDEX DBD name. Likewise, secondary indexes must not be listed.

```
[OPTIONS=(  {NOPUNCH}  [,STAT][,SUMM])]
            { PUNCH }
```

The OPTIONS control statement indicates whether any optional information is to be provided during the reorganization of the data base. Information specified on this statement affects output in the execution of the Prereorganization utility, the Prefix Resolution utility and the Prefix Update utility. The parentheses are required even if only one option is specified.

PUNCH
Specifying this option will cause the Data Base Scan list to be written to the file defined by the SYSPCH ASSGN statement. Refer to the description of the Data Base Scan utility for use of the SYSPCH output. If this option is not specified, the default is the NOPUNCH option.

STAT
Specifying this option causes the Data Base Prefix Resolution utility to print statistics of segments that are updated.

SUMM
Specifying this option causes the Data Base Prefix Resolution utility to print counts of logical relationships rather than to print individual warning messages for each unmatched logical relationship.

Example: In this example two data bases are to be reloaded. The DBD names for the two data bases are PAYRLDB and SKILINV.

```
// JOB    PREREORG
// ASSGN  SYS012,232
// DLBL   CONTROL,'CONTROL FILE'
// EXTENT SYS012,232232,1,0,1800,5
// EXEC   DLZRRC00,SIZE=250K
ULU,DLZURPR0
DBR=PAYRLDB ,SKILINV
/*
/&
```

Data Base Scan

The following job control statements are required for running the Data Base Scan utility:

// DLBL	filename	This statement defines the symbolic name of the data base file to be scanned. One statement must be present for each file that appears in the DBD describing the data base. One statement must also be present for each file that appears in DBDs of all other data bases that are related by logical relationships or indexes (primary or secondary). For HD randomized and indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements (if any). For HDAM, HIDAM, and INDEX (primary or secondary) data bases, use the filename specified in the DD1 operand of DATASET statements.
[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS010,cuu] RESTART] RESTART] extent data]	These statements are optional and are required only if a utility restart is to be attempted. The file which defines the input work file contains the partially created work file and is the same file as the WORKFIL from the previously interrupted Scan utility. The symbolic name of the input file is RESTART.

// ASSGN
 	
// TLBL
or
// DLBL
// EXTENT
	SYS013,cuu
	
	WORKFIL
	
	WORKFIL
	extent data
		These statements define the output work file. This file is used as input to the Prefix Resolution utility. It may be DASD, or tape with standard labels. WORKFIL is the symbolic name of the output file.

// ASSGN
// DLBL
// EXTENT
	SYS012,cuu
	CONTROL
	extent data
		These statements define the input file created by the Data Base Prereorganization utility. It must be DASD. CONTROL is the symbolic name of the input file.

// EXEC	DLZRRC00, SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for requirements.

The Data Base Scan utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered either via SYSIPT or SYSLOG:

```
ULU,DLZURGS0
```

The following control statements are placed after the parameter statement:

```
DBS=dbdname,segmentname
```

The DBS control statement names a data base segment that is to be scanned. One or more of these control statements may be provided. The dbdname and segmentname must be padded with blanks to provide a total length of eight characters each. If no DBS control statements are provided to this program, the control file is used. The SYSPCH output of the Prereorganization utility may be used as input to this program. The value of using the SYSPCH output as input is that, if multiple data bases need to be scanned, the SYSPCH output may be separated by data base name. Then the set of scan control statements for each data base may be submitted to different executions of the Data Base Scan utility.

Note that if any DBS control statements are provided to this program, the entire scan list contained on the control file is ignored, and work file records are generated only for those segments named on DBS control statements. Even if a scan list is provided through DBS control statements, a control file must be provided to this program as the control file contains information (other than the scan list) that the Scan utility requires. The user must ensure that all data bases indicated on the scan list provided by the Data Base Prereorganization utility are scanned before attempting to execute the Prefix Resolution utility.

```
CHKPT=  {NO   }
        {nnnnn}
```

The CHKPT control statement indicates whether checkpoints are to be taken during operation of this program. A checkpoint record is written onto the work file named WORKFIL every time the number of records specified by ‘nnnnn’ is generated. The parameter ‘nnnnn’ must be a five-digit decimal number. As each checkpoint record is generated, a checkpoint message is issued to the SYSLOG device. The message indicates the name of the program and the checkpoint number of the checkpoint record written. The checkpoint messages should be saved in case a restart operation is required. If this statement is omitted or CHKPT=NO is specified, no checkpoints will be taken.

```
RSTRT=  {NO   }
        {nnnnn}
```

The RSTRT control statement indicates whether a restart operation is to be performed by the program. If this statement is omitted or RSTRT=NO is specified, no restart operations occur. The parameter nnnnn is a five-digit decimal number and its value may be obtained from the checkpoint messages that were written to the SYSLOG device. The restart module reads records on the RESTART file until the checkpoint record numbered nnnnn is encountered. After each record is read,

it is written onto the output WORKFIL file. When the correct checkpoint record is located, a message is written onto the SYSLOG device indicating the program name and the checkpoint number, and the checkpoint record will be written onto the output WORKFIL file. Processing continues from that restart point. The volume containing the specified checkpoint record, and any succeeding volumes that were written during a previous execution of this program, must not be supplied to the Prefix Resolution utility.

Example: This example shows the scan of an HDAM data base which is logically related to another data base which the user indicated in DBR control statements supplied to the Data Base Prereorganization utility. The second data base is HIDAM and therefore the index is also assigned.

```
// JOB    SCAN
// DLBL   HDAMDB,'HDAM DATA BASE',,VSAM
// DLBL   HIDAMDB,'HIDAM DB LR',,VSAM
// DLBL   INDEXDB,'HIDAM INDEX',,VSAM
// ASSGN  SYS012,232
// DLBL   CONTROL,'CONTROL FILE'
// EXTENT SYS012,232232
// ASSGN  SYS013,181
// TLBL   WORKFIL,'SCAN WORK FILE'
// EXEC   DLZRRC00,SIZE=300K
ULU,DLZURGS0
              .
              .
      CONTROL STATEMENTS
              .
              .
/*
/&
```

Data Base Prefix Resolution

The Data Base Prefix Resolution utility is executed as if it were a VSE application program and requires the following job control statements:

// ASSGN
// DLBL
// EXTENT
	SYS00n,cuu
	SORTWKn
	extent data
		These statements define the intermediate storage files for the sort/merge. Refer to DOS/VS Sort/Merge Reference regarding specification of number and size of intermediate storage files. Sort work files are numbered consecutively from SYS001 to SYS008. Filename assignments are from SORTWK1 to SORTWK8.

[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS011,cuu] WORKFIL] WORKFIL] extent data]	These statements define the sorted output work file for logical relationships. The file is used as input to the Data Base Prefix Update utility. It may be DASD, or tape with standard labels. WORKFIL is the symbolic name of the output file. These statements must be provided if the utility control statement indicates 1 or blank in column 10 (the input work files contain logical relationship information).

[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS014,cuu] INDXWRK] INDXWRK] extent data]	These statements define the sorted output work file for secondary index relationships. The file will be used as input to the Data Base Prefix Update utility. It may be DASD, or tape with standard labels. INDXWRK is the symbolic name of the output file. These statements must be provided if the utility control statements indicates 1 or blank in column 10 (the input work files contain index relationship information).

// ASSGN
 	
// TLBL
or
// DLBL
// EXTENT
	SYS010,cuu
	
	INTRMED
	
	INTRMED
	extent data
		These statements define the intermediate sort work file. It may be tape or DASD. INTRMED is the symbolic name of the intermediate work file. This file is used to hold data between executions of the SORT utility by prefix resolution.
		Note: If message 4450A (NO MORE AVAILABLE EXTENTS) occurs, and if SYSLOG is assigned to a keyboard, ensure that the system operator does not type new extent limits for INTRMED work file. To do so results in the loss of data because, due to the way VSE works, data written on the extended part in the first sort is not read when INTRMED is used as input to the second sort. If message 4450A occurs, rerun the Prefix Resolution utility with a larger INTRMED sort work file.

// ASSGN
 	
// TLBL
or
// DLBL
	SYS013,cuu
	
	WRKINnn
	
	WRKINnn
		These statements define the input work files generated by Data Base Reload, and Data Base Scan utilities. They may be DASD or tape with standard labels. WRKINnn is the symbolic name of the input file(s), where nn is a numeric value from 01 to 99. Work files are numbered consecutively from WRKIN01 to WRKIN99.

// ASSGN
// DLBL
// EXTENT
	SYS012,cuu
	CONTROL
	extent data
		These statements define the input control file created by the Data Base Prereorganization utility. It must be DASD. CONTROL is the symbolic name of the input file (DTF).

// EXEC	DLZURG10, SIZE=xxxK	Data Base Prefix Resolution utility module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

The following control statement is required:

```
 1        10        15          21         25
```

```
R        {L    }  number of    number of    [DUMP]
         {I    }  sort work    input files
         {blank}  files
```

Column	Description
1	This must be the character R, which defines this as a Prefix Resolution utility control statement.
10	This must be one of the characters L, I, or blank. L - specifies that the input files contain only logical relationships to be resolved. I - specifies that the input files contain only secondary index relationships to be resolved.
15	This must be a numeric value and specifies the number of work files supplied to the Sort/Merge program. The number of files can be 1 through 8.

Column	Description
21-22	This must be a numeric value from 01 to 99, or blank and specifies the number of input work files supplied to the utility. Blank defaults to 01.
25-28	The DUMP parameter is optional. If specified, a storage dump will be provided. The DUMP parameter must be specified to ensure that all the steps in the job are cancelled in case of a severe error. If it is not specified and such an error occurs, the step will abnormally end, but any other steps in the job will be allowed to execute.

Example: The following example illustrates the creation of a sorted work file from two input work files, one from the HD Reorganization Reload utility and one from the Data Base Scan utility. Three intermediate storage files are used.

```
// JOB     PREFIX RESOLUTION
// ASSGN   SYS011,231
// DLBL    SORTWK1
// EXTENT  SYS001,231231,,,1800,20
// ASSGN   SYS002,232
// DLBL    SORTWK2
// EXTENT  SYS002,232232,,,1200,40
// ASSGN   SYS003,232
// DLBL    SORTWK3
// EXTENT  SYS003,232232,,,1220,700
// ASSGN   SYS010,233
// DLBL    INTRMED
// EXTENT  SYS010,233233,,,1000,800
// ASSGN   SYS011,180
// TLBL    WORKFIL,'PREFIX WORK FILE'
// ASSGN   SYS012,232
// DLBL    CONTROL,'CONTROL FILE'
// EXTENT  SYS012,232232
// ASSGN   SYS013,182
// TLBL    WRKIN01,'HDAM WORK FILE'
// TLBL    WRKIN02,'SCAN WORK FILE'
// EXEC    DLZURG10,SIZE=100K
R        L    3     02
/*
/&
```

Data Base Prefix Update

The following job control statements are required:

// DLBL	filename	This statement defines the data bases that were reorganized and any logically related data bases. One statement must be present for each data base and all related index data bases. For HD randomized and indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements (if any). For HDAM, HIDAM, and INDEX (primary and secondary) data bases, use the filename specified in the DD1 operand of DATASET statements.
[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS011,cuu] WORKFIL] WORKFIL] extent data]	These statements define the input work file for logical relationships created by the Data Base Prefix Resolution utility. It may be DASD or tape with standard labels. WORKFIL is the symbolic name of the input file. These statements must be provided if the utility control statement indicates 1 or blank in column 10 (a logical relationship work file has been created by the Prefix Resolution utility).
[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS014,cuu] INDXWRK] INDXWRK] extent data]	These statements define the input work file for secondary index relationships created by the Data Base Prefix Resolution utility. It may be DASD or tape with standard labels. INDXWRK is the symbolic name of the input file. These statements must be provided if the utility control statement indicates 1 or blank in column 10 (an index relationship workfile has been created by the Prefix Resolution utility).
// EXEC	DLZRRC00, SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

The Data Base Prefix Update utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered either via SYSIPT or SYSLOG:

```
ULU,DLZURGP0
```

The following control statement is required:

```
 1                  10

U                   {L    }
                    {I    }
                    {blank}
```

Column	Description
1	This must be the character U, which defines this as a Prefix Update utility control statement.

Column	Description
10	This must be one of the characters L, I, or blank. L - specifies that the logical relationship resolution output of the Prefix Resolution utility is to be used as input to this program. I - specifies that the secondary index relationship resolution output of the Prefix Resolution utility is to be used as input to this program. blank - Specifies that both output files of the Prefix Resolution utility are to be used as input to this program.

Example: The following example illustrates prefix updating for two data bases (PAYRLDB and SKILINV) that are logically related. The file defined by the SKILINV DD1 operand is SKLDAM. The file defined by the PAYRLDB DD1 operand is PAYROLL. Additionally, since the PAYRLDB data base is HIDAM, the INDEX data base must be defined.

```
// JOB    PREFIX UPDATE
// DLBL   SKLDAM,'HDAM DATA BASE',,VSAM
// DLBL   PAYROLL,'HIDAM DATA BASE',,VSAM
// DLBL   HIDAMDX,'HIDAM INDEX',,VSAM
// ASSGN  SYS011,181
// TLBL   WORKFIL,'PREFIX WORK FILE'
// EXEC   DLZRRC00,SIZE=350K
ULU,DLZURGP0
U        L
/*
/&
```

Data Set Image Copy

The Data Set Image Copy utility is executed as a standard VSE program and requires the following job control statements:

[// UPSI	00000001]	This optional statement changes the output tape rewind option from REWIND=UNLOAD to REWIND=NORWD. This allows multiple output files to be placed on the same volume. You can establish the sequence of dumped output files on a multi-file tape volume by specifying file sequence numbers on your TLBL statements. If the UPSI statement is not used, each output file is placed on a separate volume. This statement is ignored for DASD files.

// DLBL	filename	This statement defines the symbolic name of the data set to be copied. One statement must be present for each file that appears in the DBD describing the data base. For HD indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements. For HD randomized, HDAM, HIDAM (and its primary INDEX), and SHISAM data bases, use the filename defined in the DD1 operand of DATASET statements. For HISAM data bases, use the filenames specified in the DD1 and OVFLW operand of DATASET statements.

// ASSGN
 	
// TLBL
or
// DLBL
// EXTENT
	SYS011,cuu
	
	filename
	
	filename
	extent data
		These statements define the first copy of the dumped output file. One is required for each file to be dumped. The filename may be any name but must appear in the associated utility control statement. The file may be tape or DASD.

[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS012,cuu] filename] filename] extent data]	These statements are optional and are required only if the associated utility control statement requests two copies of the dump. The filename must appear in the control statement. It may be a tape or DASD.
// EXEC	DLZUDMP0, SIZE=xxxxK	Data Base Data Set Image Copy utility module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

One control statement is required for the Data Set Image Copy utility. Its content is described below. It is read from the SYSIPT device.

```
1 2   4-10      13-19   22-28       30  31-37     39  40-80
```

```
D n   dbdname   infile  outfile1    c   outfile2  c   comments
```

Column	Description
1	This must be the character D, which defines the statement as a Data Base Data Set Image Copy utility control statement.
2	This must be a 1 or 2, depending on the number of copies required.
4-10	This must be the name of the DBD that includes the name of the file to be dumped.
13-19	This must be the symbolic filename of the input file to be dumped. It must appear as the DD1 or OVFLW parameter of the DATASET statement or the REF parameter of the ACCESS statement in the referenced DBD. Corresponding DLBL statements must be provided.
22-28	This is the filename of the TLBL or DLBL statement for the primary output data set (SYS011).
30	This is the rewind option for the primary output data set, if stored on tape. This field must be either N, for no rewind; R, for rewind; or U, for rewind and unload. The default value is U.

Column	Description
31-37	This is the filename of the TLBL or DLBL statement for the second copy of the dumped data set (SYS012). This field must be blank if column 2 contains a 1.
39	This is the rewind option for the second copy of the dumped data set, if stored on tape. If column 2 contains a 2, this field must be either N, for no rewind; R, for rewind; or U, for rewind and unload. The default value is U.
40	Comments may be placed in columns 40 through 80.

Example 1: The first example copies one file with the filename DBHI3A from the HISAM data base named DLZDBD1. The name of the output file is DBAOUT1.

```
// JOB     DBDUMP1
// DLBL    DBHI3A,'DBHI3A',,VSAM
// ASSGN   SYS011,180
// TLBL    DBAOUT1,'DBHI3A'
// EXEC    DLZUDMP0,SIZE=100K
D1 DLZDBD1  DBHI3A   DBAOUT1           DUMP SINGLE DATA SET
/*
/&
```

Example 2: The second example copies two files with the filenames DBHI3A and DBHI3B from the data base named DLZDBD1. Two copies of file DBHI3A and one copy of DBHI3B are created. Because the UPSI statement is specified in this example, the first dumped output is a multifile tape volume. It contains the first copy of DBHI3A and the copy of DBHI3B as the first and second files, respectively. The second dumped output, also a tape, contains the second copy of DBHI3A.

```
// JOB     DBDUMP2
// UPSI    00000001
// DLBL    DBHI3A,'DBHI3A',,VSAM
// DLBL    DBHI3B,'DBHI3B',,VSAM
// ASSGN   SYS011,180
// TLBL    DBAOUT1,'DBHI3A'
// TLBL    DBBOUT1,'DBHI3B',,,,2
// EXEC    DLZUDMP0,SIZE=100K
D2 DLZDBD1  DBHI3A   DBAOUT1 UDBAOUT2 U  DATA SET 1-DUMP 1+2
D2 DLZDBD1  DBHI3B   DBBOUT1             DATA SET 2-DUMP 1
/*
/&
```

Data Base Change Accumulation

The Data Base Change Accumulation utility is executed as a standard VSE program and requires the following job control statements.

[// ASSGN [// DLBL [// EXTENT	SYS00n,cuu] SORTWKn] extent data]	These statements define the intermediate storage files for the sort/merge program. Refer to *DOS/VS-VM/System Product Sort Merge: Programmer's Guide* regarding specification and size of intermediate storage files. If there is to be no new accumulation output, the sort program is not used and no work files need be assigned. Sort work files are numbered consecutively from SYS001 to SYS008. Filename assignments are from SORTWK1 to SORTWK8.
// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS010, {cuu} {IGN} CUMOUT] CUMOUT] extent data]	These statements define the new accumulated change output file. It may be either DASD or tape with standard labels. CUMOUT is the symbolic name of the output file (DTF). If there is to be no new accumulation output, specify IGN and omit the other statements in this group or do not assign this file.
// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS011, {cuu} {IGN} CUMIN] CUMIN] extent data]	These statements define the old accumulated change input file. It may be either DASD or tape with standard labels. CUMIN is the symbolic name of the input file (DTF). If no accumulated change input file is used, IGN must be specified and the other statements omitted.
// ASSGN [// TLBL or [// DLBL	SYS012, {cuu} {IGN} LOGOUT] LOGOUT]	These statements define the new output log file. It may be tape with standard labels or a VSAM disk file. LOGOUT is the symbolic name of the output file. If there is to be no new output log data set, specify IGN and omit the TLBL statement or DLBL statement or do not assign this file.
[// ASSGN [// TLBL or [// DLBL [// EXTENT	SYSnnn,X'cuu' LOGINnn] LOGINnn] extent data]	These statements define the old input log file(s) containing the change records to be accumulated. It must be tape with standard labels unless specified differently in an LI control statement. SYSnnn is the logical unit to be assigned where nnn must be 013 unless specified differently in an LI control statement. LOGINnn is the symbolic name of the input file(s) (DTF), where nn is a numeric value from 01 to 99. Log files are numbered consecutively from LOGIN01 to LOGIN99.
// EXEC	DLZUCUM0, SIZE=xxxK	Data Base Change Accumulation utility module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

Eight types of control statements are used by the Change Accumulation utility. All or any combination of statements may be supplied, subject to the limitations described in the following section. Input control statements are read from the SYSIPT device. *ID Statement*

This statement is used to describe the table requirements and sort requirements for this change accumulation execution. This statement is optional. If it is supplied, it must be the first statement. If it is not supplied, default values are assigned as described below. Any number specified must be left-justified or have leading zeros.

```
  1        11-13       31-33        41          51-52

ID       number     maximum key   number of   number of
         of DBDs    length        sort work   input log
                                  files       files
```

AI Statement

This statement is used to control positioning for the input change accumulation file. Only one AI statement is allowed. The statement format is:

```
1-2       11-16        18

AI        SYSnnn       U
                       N
                       R
```

SYSnnn is the logical unit of the input change accumulation file. U (unload), N (no rewind), and R (rewind) specify disposition of the tape after reading.

AO Statement

This statement is used to control positioning for the output change accumulation file. Only one AO statement is allowed. The statement format is:

```
1-2       11-16        18

AO        SYSnnn       U
                       N
                       R
```

SYSnnn is the logical unit of the output change accumulation file. U (unload), N (no rewind), and R (rewind) specify disposition of the tape after writing. *DB0 Statement*

This statement is used to describe which records are to be accumulated from the input log files and the old change accumulation file, if one is provided as input, and written to the new change accumulation file. One or more of these statements may be used, or this statement may be omitted. If it is omitted and at least one DB1 statement is used, there will be no new change accumulation file. The options available are described below.

```
1    4-10     12-20

DB0  dbdname  yydddhhmm
      *ALL
      *OTHER
```

Columns 4-10 may contain a data base name, *ALL or *OTHER. If *ALL or *OTHER, column 4 must be blank, and *ALL or *OTHER specified beginning in column 5.

*ALL - the purge date in columns 12-20, if any, is applied to all input change records. All records not purged are written onto the new change accumulation file. This must be the only DB0 statement.

*OTHER - All change records not matching data base names specified on DB1 statements and not purged, are accumulated. *OTHER can be specified in

conjunction with the data base name option on other DB0 statements, or as the only DB0 statement. In either case, only the data base name option is allowed on the DB1 statements, and at least one must be specified.

dbdname - The purge date is applied to those change records which match the supplied data base name (DBD name), and the resulting records are accumulated. Any number of DB0 statements with this option may be supplied.

If an old change accumulation file is provided as input, all records described by the DB0 statement(s) are checked for purging as specified. Any other records are written to the new change accumulation file.

When not using the *ALL option, all data sets (primary index, secondary indexes, and primary data) must be named.

Columns 12-20 may contain a purge date in the form YYDDDHHMM, where YY=year, DDD=day of year, HH=hour (00-23), MM=minute (00-59). If a purge date is supplied, all records matching a data base name description and dated before the purge date are eliminated. If an old accumulated change input is supplied, any records matching the data base name described by the identification in columns 4-10 and dated before the purge date are not merged into the new change accumulation file. If this field is blank, no purge date is used.

DB1 Statement

This statement is used to describe which records are to be written to the new log file. One or more of these statements may be used, or this statement may be omitted. If it is omitted, there will be no new log file. The options available are described below.

```
  1   4-10    12-20
```

```
DB1 dbdname yydddhhmm
     *ALL
     *OTHER
```

Columns 4-10 may contain a data base name, *ALL or *OTHER. If *ALL or *OTHER, column 4 must be blank, and *ALL or *OTHER specified beginning in column 5.

*ALL - the purge date in columns 12-20, if any, is applied to all input change records. All records not purged are written onto the new log file. This must be the only DB1 statement.

*OTHER - All change records not matching data base names specified on DB0 statements and not purged, are written to the new log. *OTHER can be specified in conjunction with the data base name option on other DB1 statements, or as the only DB1 statement. In either case, only the data base name option is allowed on the DB0 statements, and at least one must be specified.

dbdname - The purge date is applied to those change records which match the supplied data base name (DBD name), and the resulting records are accumulated. Any number of DB1 statements with this option may be supplied.

When not using the *ALL option, all data sets (primary index, secondary indexes, and primary data) must be named.

Columns 12-20 may contain a purge date in the form YYDDDHHMM, where YY=year, DDD=day of year, HH=hour (00-23), MM=minute (00-59). If a purge date is supplied, all records matching a data base name description and dated before the purge date are eliminated. If this field is blank, no purge date is used.

LI Statement

This statement describes the input log files. Up to 99 LI statements may be specified. The first LI statement describes log file LOGIN01; the second LI statement, LOGIN02, etc. The necessary job control statements must also be specified for each log file. The LI control statement must follow any ID or DB control statements.

```
 1-2      11-16     18        21          31

LI       [SYSnnn]  U         L           [nnnnn]
                   N         U
                   R         C
                             V
                             S
```

SYSnnn is the logical unit for this file (SYS013 is the default). U (unload), N (no rewind), and R (rewind) specify disposition of the tape after reading. L (labeled DL/I tape), U (unlabeled CICS/VS tape), C (labeled CICS/VS tape), V (VSAM DL/I disk), and S (SAM CICS/VS disk) indicate the type of log file being input. nnnnn gives the buffer size (default 1100).

LO Statement

This statement is used to control positioning for the output log files. Only one LO statement is allowed. The statement format is:

```
 1-2        11-16      18

LO         SYSnnn     N
                      U
                      R
```

SYSnnn is the logical unit of the output log file. U (unload), N (no rewind), and R (rewind) specify disposition of the tape after writing.

Example 1: This example shows that all records for data base DLZDBD1 are to be accumulated, eliminating all records for the data base before day 285 of year 83 and before 1200 hours. All records for other data bases are to be written to the new log file. There is no old accumulation change file to update.

```
// JOB      DBACCUM1
// ASSGN    SYS011,IGN
// ASSGN    SYS010,180
// TLBL     CUMOUT,'CUMOUT'
// ASSGN    SYS015,181
// TLBL     LOGIN03,'DLI.LOG4'
// ASSGN    SYS001,231
// DLBL     SORTWK1
// EXTENT   SYS001,231231,,,1800,10
// ASSGN    SYS002,232
// DLBL     SORTWK2
// EXTENT   SYS002,232232,,,1200,12
// ASSGN    SYS003,232
// DLBL     SORTWK3
// EXTENT   SYS003,232232,,,1214,700
// ASSGN    SYS012,233
// DLBL     LOGOUT,'DLI.LOG3',,VSAM
// ASSGN    SYS013,234
// DLBL     LOGIN01,'DLI.LOG1',,VSAM
// EXTENT   SYS013,234234
// ASSGN    SYS014,235
// DLBL     LOGIN02,'DLI.LOG2',,VSAM
// EXTENT   SYS014,235235
// EXEC     DLZUCUM0,SIZE=100K
// ASSGN    SYS016,182
// TLBL     ACCUMFI1,'DLI.ACMF'
// EXEC     DLZUCUM0,SIZE=100K
ID        1                  3         2
DB0DLZDBD1 832851200
DB1 *OTHER
LI        SYS013    V
LI        SYS014    V
LI        SYS015 U  L
AI        SYS016 N
AO        SYS016 R
/*
/&
```

Example 2: This example shows the accumulation of all data base change records. The ID statement in this example specifies that no DBD name is supplied. The maximum root-segment field length is specified as 4 bytes, because all log records reflect HD-type organizations. The DB control statement specifies that all records are to be accumulated, with all records before day 295 of year 83 being eliminated. An old change accumulation file is to be merged with the new change accumulation file. The purge date is also applied to the old accumulation file.

```
// JOB      DBACCUM2
// ASSGN    SYS010,180
// TLBL     CUMOUT,'CUMOUT1'
// ASSGN    SYS011,181
// TLBL     CUMIN,'CUMIN'
// ASSGN    SYS001,231
// DLBL     SORTWK1
// EXTENT   SYS001,231231,,,1800,10
// ASSGN    SYS002,232
// DLBL     SORTWK2
// EXTENT   SYS002,232232,,,1200,12
// ASSGN    SYS003,232
// DLBL     SORTWK3
// EXTENT   SYS003,232232,,,1214,700
// ASSGN    SYS012,IGN
// ASSGN    SYS013,182
// TLBL     LOGIN01,'DLZLOG73200'
// EXEC     DLZUCUM0,SIZE=100K
ID        1
DB0 *ALL   832950000
/*
/&
```

Data Set Recovery

The following job control statements are required for the Data Set Recovery utility.

// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS011, {cuu} {IGN} DUMPIN] DUMPIN] extent data]	These statements define the input file. It may be a file created by the Data Base Data Set Image Copy utility or the HISAM Reorganization Unload utility. It may be DASD or tape with standard labels. DUMPIN is the symbolic name of the input file (DTF). If no input is supplied, IGN must be specified but the other statements in this group may then be omitted.
// ASSGN [// TLBL or [// DLBL [// EXTENT	SYS012, {cuu} {IGN} CUMIN] CUMIN] extent data]	These statements define the accumulated change input file. It may be either DASD or tape with standard labels. CUMIN is the symbolic name of the input file (DTF). If no input is supplied, IGN must be specified but the other statments in this group may then be omitted.
// ASSGN [// TLBL or [// DLBL	SYSnnn, {cuu} {IGN} LOGINnn] LOGINnn]	These statements define the input log file. It must be tape with standard labels unless specified differently in an LI control statement. SYSnnn is the logical unit to be assigned where nnn must be 013 unless specified differently in an LI control statement. LOGINnn is the symbolic name of the input file(s) (DTF or ACB) where nn is a numeric value from 01 to 99. Log files are numbered consecutively from LOGIN01 to LOGIN99. If no log input is supplied, IGN must be specified but the TLBL statement or DLBL statements may then be omitted.

// DLBL	filename	This statement defines the symbolic name of the data set to be copied. One statement must be present for each file that appears in the DBD describing the data base. For HD randomized and HD indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements (if any). For HDAM, HIDAM, INDEX, and SHISAM data bases, use the filename defined in the DD1 operand of DATASET statements. For HISAM data bases, use the filenames specified in the DD1 and OVFLW operands of DATASET statements.
// EXEC	DLZRRC00, SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

Note: VSE message OP47A UNX INTERV SYS013=cuu may appear on the SYSLOG device whenever the second and successive input log files are opened and the previous file is still rewinding. This is a normal condition, and processing continues when the correct log file is mounted.

The Data Base Data Set Recovery utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered via either SYSIPT or SYSLOG:

```
UDR,DLZURDB0,dbdname
          [,{buf}][,HDBRF=][,HSBFR=][,TRACE=]
            {1  }
```

dbdname
is the name of the SHISAM, HISAM, HD indexed, HD randomized, HIDAM, HDAM, or INDEX DBD that includes the file to be recovered. The DL/I utility PSB name is generated from this name and the PSB is loaded. When recovering a primary INDEX data base, the dbdname must be that of the HD indexed or HIDAM data base, since only one utility PSB is created for HD indexed or HIDAM.

The positional parameter ‘buf’ and the keyword parameters HDBFR, HSBFR, and TRACE are optional parameters that may also be entered in the parameter statement. These parameters have the same usage as for application programs.

Three types of control statements are used by the Data Set Recovery utility. Input control statements are read from the SYSIPT device.

S Statement

This statement describes the data set to be recovered. It is required and must be the first statement.

```
 1  4         13         22-23         25

S  dbdname   filename   number of     [comments]
                        log files
```

dbdname must be the name of the SHISAM, HD indexed, HD randomized, HISAM, HDAM, INDEX, or HIDAM DBD that describes the file to be recovered.

filename must be the symbolic name of the file to be recovered. It must appear as the DD1 or OVFLW parameter of the DATASET statement or the REF parameter of the ACCESS statement in the referenced DBD, and corresponding DLBL statements must have been provided.

Columns 22 and 23 contain the actual number of input log files (from 1 to 99). This is equal to the maximum value nn used on LOGINnn TLBL or DLBL card(s). If the value is omitted or zero, and no LI statements are supplied, and SYS013 is assigned to tape, one log tape input is assumed. This value is overridden if LI control statements are supplied.

LI Statement

This statement describes the input log files. Up to 99 LI statements may be specified. The first LI statement describes log file LOGIN01; the second LI statement, LOGIN02; etc. The necessary job control statements must also be specified for each log file. If no LI statements are present, all input log files are assumed to be standard DL/I labeled tape on logical unit SYS013 with a buffer size of 1024 bytes. The number of input logs is determined from the S statement.

```
 1   11             18        21          31

LI   [SYSnnn]      U         L           [nnnnn]
                   N         U
                   R         C
                             V
                             S
```

SYSnnn is the logical device for the file. U (unload), N (no rewind), and R (rewind) specify disposition of the tape after reading. L (labeled DL/I tape), U (unlabeled CICS/VS tape), C (labeled CICS/VS tape), V (VSAM DL/I disk), and S (SAM CICS/VS disk) indicate the type of log file being input. nnnnn is the blocksize.

DI Statement

This statement is used to control positioning for the image copy or unload input file. Only one DI statement is allowed. The statement format is:

```
 1-2        11-16     18
```

```
DI          SYSnnn   U
                     N
                     R
```

SYSnnn is the logical unit. U (unload), N (no rewind), and R (rewind) specify disposition of the tape after reading.

Example 1: This example shows recovery of a key sequenced data set (KSDS) with a filename of DBHI3A in a HISAM data base named DLZDBD1. Input is provided from an image copy and change accumulation tape.

```
     // JOB     RECOVER1
     // ASSGN   SYS011,180
     // TLBL    DUMPIN,'DBHI3A'
     // ASSGN   SYS012,181
     // TLBL    CUMIN,'DLZCUM'
     // ASSGN   SYS013,IGN
     // DLBL    DBHI3A,'DBHI3A',,VSAM
     // EXEC    DLZRRC00,SIZE=300K
     UDR,DLZURDB0,DLZDBD1
     S  DLBDBD1  DBHI3A
     /*
     /&
```

Example 2: This example shows recovery of an entry sequenced data set (ESDS) with a filename DBHI3B in a HISAM data base named DLZDBD1. Input is provided from an image copy, a DL/I disk log, and a more recent tape log.

```
// JOB     RECOVER2
// ASSGN   SYS011,180
// TLBL    DUMPIN,'DBHI3B'
// ASSGN   SYS012,IGN
// ASSGN   SYS015,182
// TLBL    LOGIN02,'DLI.LOG2'
// DLBL    LOGIN01,'DLI.LOG1',,VSAM
// DLBL    DBHI3B,'DBHI3B',,VSAM
// EXEC    DLZRRC00,SIZE=300K
UDR,DLZURDB0,DLZDBD1
S  DLZDBD1  DBHI3B
LI        SYS013    V
LI        SYS015 U  L
DI        SYS011 N
/*
/&
```

Example 3: This example shows restoring a key sequenced data set (KSDS) with a filename of DBHI3A in a HISAM data base named DLZDBD1. No forward recovery is performed.

```
// JOB     RESTORE1
// ASSGN   SYS011,180
// TLBL    DUMPIN,'DBHI3A'
// ASSGN   SYS012,IGN
// ASSGN   SYS013,IGN
// DLBL    DBHI3A,'DBHI3A',,VSAM
// EXEC    DLZRRC00,SIZE=300K
UDR,DLZURDB0,DLZDBD1
S  DLZDBD1  DBHI3A
/*
/&
```

Example 4: This example shows restoring an entry sequenced data set (ESDS) with a filename DBHI3B in a HISAM data base named DLZDBD1. No forward recovery is performed.

```
// JOB     RESTORE2
// ASSGN   SYS011,180
// TLBL    DUMPIN,'DBHI3B'
// ASSGN   SYS012,IGN
// ASSGN   SYS013,IGN
// DLBL    DBHI3B,'DBHI3B',,VSAM
// EXEC    DLZRRC00,SIZE=300K
UDR,DLZURDB0,DLZDBD1
S  DLZDBD1  DBHI3B
/*
/&
```

Log Print

The Log Print utility executes as a standard VSE program and requires the following job control statements:

// ASSGN [// TLBL or [// DLBL [// EXTENT	SYSnnn,cuu LOGINnn] LOGINnn] extent data]	These statements define the input log file(s) containing the records to be printed. It must be tape with standard labels unless specified differently in an LI control statement. SYSnnn is the logical unit to be assigned, where nnn must be 013 unless specified differently in an LI control statement. LOGINnn is the symbolic name of the input file(s) where nn is a numeric value from 01 to 99. Log files are numbered consecutively from LOGIN01 to LOGIN99. LI control statements must be specified if more than one log file is to be processed.
[// ASSGN [// TLBL	SYS012,cuu] LOGOUT]	These statements define the new output log file. It must be tape with standard labels. LOGOUT is the symbolic name of the output file. These statements are required only if COPY is specified on the LO control statement.
// EXEC	DLZLOGP0, SIZE=xxxK	Log Print utility module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

This utility uses three types of control statements, LO, LS, and LI. Input control statements are read from the SYSIPT device.

LO Statement

This statement describes which records are to be printed by the utility and which format is desired. The statement is optional, and if not supplied, default values are assigned as described below. Only one LO statement may be specified.

```
1-2 4-10       13-21        23-31        33-39      41-44   46-50      52

LO  [psbname]  [yydddhhmm]  [yydddhhmm]  [KEYWORD]  [COPY]  [DIDSD]    [U]
                                                                       [N]
                                                                       [R]
                                         [DUMP   ]
```

Column	Description
1-2	Required. Must contain the characters LO.
3	Must be blank.
4-10	Optional. This parameter applies to DL/I records only. It identifies a 1- to 7-character PSB name. If specified, only log records containing this PSB name will be printed. Because open records (log code X'2F') contain no PSB name, none will be printed. If omitted, all log records will be printed unless excluded by another parameter on the LO or LS statement.
11-12	Must be blank.
13-21	Optional. This parameter contains a start date of the form yydddhhmm where yy=year (00-99), ddd=day (000-366), hh=hour (00-23), and mm=minute (00-59). If specified, only log records having a date equal to or later than this start date will be printed. Because only DL/I data base records (log codes X'50' and X'51') contain a recognizable date, other type records encountered before the first printable data base record will not be printed. If omitted, all log records will be printed unless excluded by another parameter on the LO or LS statement.
22	Must be blank.
23-31	Optional. This parameter contains an end date of the form yydddhhmm where yy=year (00-99), ddd=day (000-366), hh=hour (00-23), and mm=minute (00-59). If specified, only records having a date equal to or earlier than this end date will be printed. This end date must not be earlier than that specified in columns 13-21. Because only DL/I data base records contain a recognizable date, other type records encountered after the last printable data record will not be printed. If omitted, all log records will be printed unless excluded by another parameter on the LO or LS statement.
32	Must be blank.

Column	Description
33-39	Optional. This parameter applies to DL/I records only. It identifies the printed output format either KEYWORD or DUMP. If KEYWORD is specified, the record contents are printed in keyword format (each field individually identified). If DUMP is specified, the record contents are printed in dump format (both hexadecimal and character format of 32 bytes per line). If this parameter is omitted, KEYWORD is assumed.
40	Must be blank.
41-44	Optional. This parameter is used to request the utility copy feature. If COPY is specified, the utility copies all log records from the input log file(s) to the new output log tape until an invalid record is encountered. At that time, the utility closes the new log tape and terminates processing. The new log tape file may then be used as input to the Backout utility. If no invalid record is encountered, a complete copy of the input log file(s) is created. If this parameter is omitted, no output log tape is created.
45	Must be blank.
46-50	Optional. This parameter is used to request printing of CICS/VS journal records. If CICSD is specified, all CICS/VS journal records (and any other non-DL/I records) are printed in dump format. These records are printed in addition to the DL/I records. Records outside the time range specified in columns 13-21 and 23-31 will not be printed. If this parameter is omitted, only DL/I records will be printed.
51	Must be blank.
52	Column 52 identifies the rewind option for the output log file. U - Rewind and unload N - No rewind R - Rewind only If this parameter is omitted, N is assumed.
53-80	Must be blank.

LS Statement

With this statement you can selectively print only those DL/I records associated with a particular DBD name, CICS/VS task ID, or relative block number. This statement is optional. If it is not supplied, selective log printing is not performed. Only one LS statement may be specified.

```
  1-2   4-10        12-16      18-25

LS   [dbdname]    [nnnnn]     [xxxxxxxx]
```

Column	Description
1-2	Required. Must contain the characters LS.
3	Must be blank.

Column	Description
4-10	Optional. This parameter identifies a 1- to 7-character DBD name. If specified, only open records (log code X'2F') and data base records (log codes X'50' and X'51') containing this DBD name will be printed. If omitted, all log records will be printed unless excluded by another parameter on the LS or LO statement.
11	Must be blank.
12-16	Optional. This parameter identifies a 1- to 5-digit CICS/VS task ID. It must be left-justified and contain decimal digits (0-9). If specified, only scheduling records (log code X'08'), termination records (log code X'07'), and data base records (log codes X'50' and X'51') associated with this ID will be printed. Note: No log records will be printed if the log file is not a CICS/VS journal or if the appropriate scheduling is not present. If omitted, all log records will be printed unless excluded by another parameter on the LS or LO statement.
17	Must be blank.
18-25	Optional. This parameter identifies a 1- to 8-digit RBN (relative block number). It must be left-justified and contain only hexadecimal digits (0-9, A-F). If specified, only data base records (log codes X'50' and X'51') containing this RBN will be printed. If omitted, all log records will be printed unless excluded by another parameter on the LS or LO statement.
26-80	Must be blank.

LI Statement

This statement describes the input log files. Up to 99 LI statements may be specified. The first LI statement describes log file LOGIN01; the second LI statement, LOGIN02; etc. The necessary job control statements must also be specified for each log file. Multiple log files must be in chronological order, that is, LOGIN01 must have been created before LOGIN02, LOGIN02 before LOGIN03, etc. If this statement is omitted, one DL/I standard labeled tape log file on logical unit SYS013 with a buffer size of 1024 bytes is assumed. All columns between parameters must be blank.

```
 1-2      11-16     18    21      31-35

LI      [SYSnnn]   U     L       [nnnnn]
                   N     U
                   R     C
                         V
                         S
```

Column	Description
1-2	Required. Must contain the characters LI.
3-10	Must be blank.

Column	Description
11-16	Optional. This parameter identifies the logical unit for this log file. If specified, it must be of the form SYSnnn, where nnn is a valid logical unit assignment. If omitted, SYS013 is assumed.
17	Must be blank.
18	N means R means - (default for labeled tape) U means - (default for unlabeled tape)
19-20	Must be blank
21	Optional. This parameter identifies the log file type. If specified, it must be L, U, C, V, or S where: L - specifies standard labeled tape log file (DL/I). U - specifies unlabeled tape log file (CICS/VS). C - specifies labeled tape log file (CICS/VS). V - specifies VSAM disk log file (DL/I). S - specifies SAM disk log file (CICS/VS). If omitted, L is assumed.
22-30	Must be blank.
31-35	Optional. This parameter identifies the buffer size (in bytes) used for this log. If specified, the value must be left-justified and must not be less than 1100 nor greater than 32767. Note: This parameter must be specified for a CICS/VS journal with a buffer size greater than 1100 bytes. If omitted, 1100 is assumed.
36-80	Must be blank.

Example 1: This example prints the contents of two DL/I VSAM disk log files and one tape file. Only records associated with PSB name DLZPSB1 occurring on or after February 1, 1983, will be printed. The records will be printed in keyword format.

```
// JOB     LOGP1
// DLBL    LOGIN01,'DLI.LOG1',,VSAM
// DLBL    LOGIN02,'DLI.LOG2',,VSAM
// ASSGN   SYS015,182
// TLBL    LOGIN02,'DLI.LOG2'
// EXEC    DLZLOGP0,SIZE=100K
LO DLZPSB1  830320000
LI                  V
LI        SYS014    V
LI        SYS015 U  L
/*
/&
```

Example 2: This example prints the contents of a standard labeled DL/I tape log file and copies the contents to a new tape log file until an invalid record is encountered. The new tape log may be used as input to the Backout utility. The log records printed will be in keyword format.

```
// JOB   LOGP2
// ASSGN SYS013,180
// TLBL  LOGIN01,'DLZLOG77031'
// ASSGN SYS012,181
// TLBL  LOGOUT,'DLZLOG77031'
// EXEC  DLZLOGP0,SIZE=300K
LO                              KEYWORD COPY
/*
/&
```

Example 3: This example prints the contents of an unlabeled CICS/VS tape journal whose buffer size is 4096. Only data base log records that are associated with CICS/VS task ID 27 and relative block number 3E and occurred before January 1, 1983, will be printed.

```
// JOB   LOGP3
// ASSGN SYS013,180
// TLBL  LOGIN01
// EXEC  DLZLOGP0,SIZE=100K
LO                    823652359
LS         27    3E
LI        SYS013    U         4096
/*
/&
```

Example 4: This example of the Log Print utility prints the contents of a standard labeled DL/I tape log file and copies the contents to a new tape log file until an invalid record is encountered. The new tape log may be used as input to the Backout utility. The log records printed will be in keyword format.

```
// JOB    LOGP4
// ASSGN  SYS013,180
// TLBL   LOGIN01,'DLZLOG77031'
// ASSGN  SYS012,181
// TLBL   LOGOUT,'DLZLOG77031'
// EXEC   DLZLOGP0,SIZE=300K
LO                                      COPY
/*
/&
```

Data Base Backout

The following JCL is required for the Data Base Backout utility.

// ASSGN [// TLBL or [// DLBL	SYSnnn,cuu LOGINnn] LOGINnn]	These statements define the input log file(s). It must be tape with standard labels unless specified differently in an LI control statement. SYSnnn is the logical unit to be assigned, where nnn must be 012 unless specified differently in an LI control statement. LOGINnn is the symbolic name of the input file, where nn is a numeric value from 01 to 99. Log files are numbered consecutively from LOGIN01 to LOGIN99. LI control statements must be specified if more than one input log file is to be processed. Multi-volume files and 7-track tape files with data conversion cannot be read backward because of a VSE restriction. However, multi-volume files can be processed by describing each volume as a separate file.
[// ASSGN [// TLBL or [// DLBL	SYS011,cuu] LOGOUT] {DSKLOG1}] {DSKLOG2}]	These statements define the output log file. It may be tape with standard labels or a VSAM disk file.
// DLBL	filename	This statement defines the symbolic name of the data set to be backed out. One statement must be present for each file that appears in the DBD describing the data base. For HD randomized and HD indexed data bases, use the filenames defined in the DD1 operand of DATASET statements and the REF operand of ACCESS statements (if any). For HDAM, HIDAM, INDEX, and SHISAM data bases, use the filename defined in the DD1 operand of DATASET statements. For HISAM data bases, use the filenames specified in the DD1 and OVFLW operand of DATASET statements.
// EXEC	DLZRRC00, SIZE=xxxK	DL/I initialization module. Refer to *DL/I DOS/VS Data Base Administration* for storage requirements.

The Data Base Backout utility is executed as if it were an application program under the control of DL/I. The following parameter data, beginning in column 1, must be entered either via SYSIPT or SYSLOG:

```
DLI,DLZBACK0,psbname
           [,{buf}][,HDBFR=][,HSBFR=][,TRACE=]
             {1  }
           [,ASLOG=][,LOG=]
```

psbname
is the name of the PSB that references the data bases to be backed out. This must be the same PSB name specified for the DL/I application program which terminated abnormally.

The positional parameter buf and the keyword parameters HDBFR, HSBFR, TRACE, ASLOG, and LOG are optional parameters that may also be entered in the parameter statement. These parameters have the same usage as for application programs. The LOG parameter must be specified if the output log is to be a VSAM disk file.

One type of control statement is used by the Backout utility. It is read from the SYSIPT device.

LI Statement

This statement describes the input log files. Up to 99 LI statements may be specified. The first LI statement describes log file LOGIN01; the second statement, LOGIN02; etc. The necessary job control statements must also be specified for each log file. Multiple log files must be in reverse chronological order. That is, LOGIN01 must have been created after LOGIN02, LOGIN02 after LOGIN03, etc. If this statement is omitted, one DL/I standard labeled tape log file on logical unit SYS012 with a buffer size of 1100 bytes is assumed.

```
 1          11          18     21           31

LI         [SYSnnn]    U      L          [nnnnn]
                       N      U
                       R      C
                              V
```

Column	Desciption
1	Columns 1 and 2 must contain the characters LI.
11	Columns 11-16 identify the logical unit for this log file. It must be of the form SYSnnn, where nnn is a valid logical unit assignment. If this parameter is omitted, SYS012 is assumed.
18	Column 18 identifies the rewind option for the input log file. U - Rewind and unload; N - No rewind (default for labeled tape); R - Rewind only (default of unlabeled tape).
21	Column 21 identifies the log file type and may be L, U, C, or V. L - Specifies standard labeled tape log file (DL/I). U - Specifies unlabeled tape log file (CICS/VS). C - Specifies standard labeled tape log file (CICS/VS). V - Specifies VSAM disk log file (DL/I). If this parameter is omitted, L is assumed.
31	Columns 31-35 identify the buffer size in bytes used for this log file. This value must be specified for a CICS/VS journal with a buffer size greater than 1100 bytes. The value must be left-justified and must not be less than 1100 nor greater than 32767. If this parameter is omitted, 1100 is assumed.

Example 1: This example backs out the data base changes made by the program which uses PSB DLZPSB1. Both the input log and the output log are standard labeled DL/I tape files.

```
// JOB    BACKOUT
// ASSGN  SYS012,180
// TLBL   LOGIN01,'LOGIN'
// MTC    REW,SYS012
// MTC    FSF,SYS012,3
// ASSGN  SYS011,181
// TLBL   LOGOUT,'LOGOUT'
// DLBL   DBHI3A,'DBHI3A',,VSAM
// DLBL   DBHI3B,'DBHI3B',,VSAM
// EXEC   DLZRRC00,SIZE=300K
DLI,DLZBACK0,DLZPSB1,1
LI        SYS012 U  L
/*
/&
```

Example 2: This example backs out the data base changes made by the program that uses PSB DLZPSB2. The log input consists of two DL/I VSAM disk log files. DLI.LOG2 was created after DLI.LOG1 and must be processed first by the utility. The new output log is also a VSAM disk file.

```
// JOB     BACKOUT2
// DLBL    LOGIN01,'DLI.LOG2',,VSAM
// DLBL    LOGIN02,'DLI.LOG1',,VSAM
// DLBL    DSKLOG1,'DLI.LOG3',,VSAM
// DLBL    DBHI3C,'DBHI3C',,VSAM
// EXEC    DLZRRC00,SIZE=300K
DLI,DLZBACK0,DLZPSB2,1,LOG=(DISK1)
LI        SYS014    V
LI        SYS015    V
/*
/&
```

Glossary

A number of terms and phrases used in describing DL/I DOS/VS are either new to most readers, or have new meanings. To improve the readability and your understanding of this and other DL/I DOS/VS publications, the significant and important terms are defined in the following Glossary. Some of the definitions refer to the representative DL/I hierarchical structure shown in Figure 8.

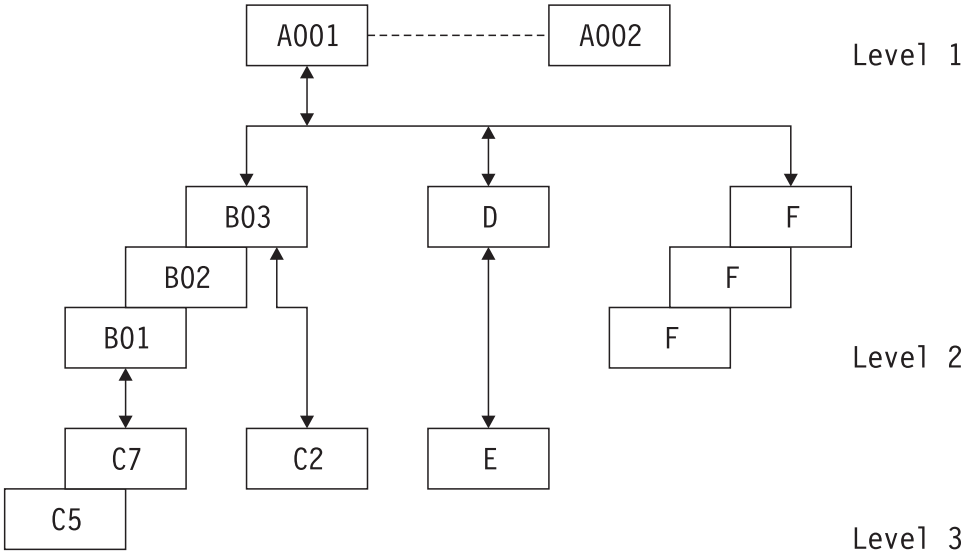

Figure 8. Representative DL/I Hierarchical Structure

- Each block represents a segment.
- The segment names are A through F.
- The numbers represent the sequence fields (keys).
  If no number is present, the segment has no key.
- The lines connecting the segment blocks show the hierarchical paths.

ACB. Application control block.

ACBGEN. Application control block generation.

ACT. Application control table.

aggregate. See *data aggregate*.

application control blocks. The control blocks created from the output of DBDGEN and PSBGEN, e.g., a DMB of an internal PSB created by the ACB utility program.

application control block generation (ACBGEN). The process by which application control blocks are created.

application control table (ACT). A DL/I online table describing those CICS/VS application programs that utilize DL/I.

attribute. A property of an entity. It contains a value. Synonymous with *field*.

batch processing. A processing environment in which data base transactions requested by applications are accumulated and then processed periodically against a data base.

business process. A defined function of a business enterprise, usually interrelated through information requirements with other business processes. Example: Personnel management is the business process responsible for employee welfare from pre-hire to retirement. Related to the accounting business process through payroll.

child. Synonymous with *child segment*.

child segment. A segment one or more levels below the segment which is its parent, but with a direct path back up to the parent. Depending on the structure of the data base, a parent may have many children; however, a child has only one parent segment. Referring to Figure 8:

- all the B, C, D, E, and F segments are children of A-001.

- C-5 and C-7 are children of B-01 (and A-001), but not children of the other B segments.
- B-02 has no children.

See also *logical child* and *physical child*.

concatenated key. The key constructed to access a particular segment. It consists of the key fields, including that of the root segment and successive children down to the accessed segment.

data aggregate. A group of data elements that describe a particular entity. Synonymous with *segment*. See also *data element*.

data base (DB). (1) (ISO)[1] A set of data, part of the whole of another set of data, and consisting of at least one file, that is sufficient for a given purpose or for a given data processing system. (2) A collection of data records comprised of one or more data sets. (3) A collection of interrelated or independent data items stored together without unnecessary redundancy to serve one or more applications. See *physical data base* and *logical data base*.

data base administration (DBA). The tasks associated with defining the rules by which data is accessed and stored. The typical tasks of data base administration are outlined in the *DL/I DOS/VS Data Base Administration*, SH24-5011.

data base administrator (DBA). A person in an installation who has the responsibility (full or part time) for technically supporting the use of DL/I.

data base definition (DBD). A description of the physical characteristics of a DL/I data base. One DBD is generated and cataloged in a core image library for each data base that is used in the installation. It defines the structure, segment keys, physical organization, names, access method, devices, etc., of the data base.

data base integrity. The protection of data items in a data base while they are available to any application program. This includes the isolation of the effects of concurrent updates to a data base by two or more application programs.

data base organization. The physical arrangement of related data on a storage device. DL/I data base organizations are hierarchical direct (HD) and hierarchical sequential (HS). See *hierarchical direct organization* and *hierarchical sequential organization*.

data base record. A collection of DL/I data elements called segments hierarchically related to a single root segment.

data dictionary. (1) A centralized repository of information about data, such as: its meaning, relationship to other data, usage, and format. (2) A program to assist in effectively planning, controlling, and evaluating the collection, storage, and use of data. For example, DOS/VS DB/DC Data Dictionary.

data element. The smallest unit of data that can be referred to. Synonymous with *field*. See also *data aggregate*.

data field. Synonymous with *field*.

data independence. (1) The concept of separating the definitions of logical and physical data such that application programs do not depend on where or how physical units of data are stored. (2) The reduction of application program modification in data storage structure and access strategy.

data management block (DMB). The data management block is created from a DBD by the application control blocks creation and maintenance utility, link edited, and cataloged in a core image library. The DMB describes all physical characteristics of a data base. Before an application program using DL/I facilities can be run, one DMB for each data base accessed, plus a PSB for the program itself, must be cataloged in a core image library. The DMBs and the associated PSB are automatically loaded into main storage from the core image library at the beginning of the application program execution (their loading is controlled by the parameter information supplied to DL/I at the beginning of program execution).

data set. A named organized collection of logically related records. They may be organized sequentially, as in the case of DOS/VSE SAM, or in key entry sequence, as in the case of VSE/VSAM. Synonymous with *file*.

DB. Data base.

DBA. (1) Data base administration (2) Data base administrator.

DBD. Data base description.

DBDGEN. Data base definition generation—the process by which a DBD is created.

DB/DC. Data base/data communication.

[1] International Organization for Standardization, Technical Committee 97/Subcommittee 1

DC. Data communication.

dependent segment. A DL/I segment that relies on at least the root segment (and other dependent segments) for its full hierarchical meaning. Synonymous with *child segment*.

destination parent. The physical or logical parent segment reached by the logical child path.

device independence. The concept of writing application programs such that they do not depend on the physical characteristics of the device on which data is stored.

direct access. The retrieval or storage of a VSAM data record independent of the record's location relative to the previously retrieved or stored record.

DMB. Data management block.

entity. An item about which information is stored. It has properties that can be recorded. Information about an entity is a record.

entry sequenced data set (ESDS). A VSAM data set whose records are physically in the same order as they were put in the data set. It is processed by addressed direct access or addressed sequential access and has no index. New records are added at the end of the data set.

ESDS. Entry sequenced data set.

field. (1) a unique or nonunique area (as defined during DBDGEN) within a segment that is the smallest unit of data that can be referred to. See also *key field*. (2) Any designated portion of a segment.

field level sensitivity. The ability of an application program to access data at the field level. See *sensitivity*.

forward. Movement in a direction from the beginning of the data base to the end of the data base, accessing each record in ascending root key sequence, and accessing the dependent segments of each root segment from top to bottom and from left to right. Referring to Figure 8, forward accessing of all segments shown would be in the following sequence:

A-001, B-01, C-5, C-7, B-02, B-03, C-2, D, E, F, F, F, A-002.

HD. Hierarchical direct.

HDAM. Hierarchical direct access method.

HIDAM. Hierarchical indexed direct access method.

HIDAM index. A data base that consists of logical DL/I records each containing an image of the key field of a HIDAM root segment. A HIDAM index data base consists of one VSAM KSDS (keyed sequenced data set).

hierarchical direct access method (HDAM). Provides for direct access to a DL/I data base in the HD organization. Segments are stored in VSAM control intervals and are referenced by a relative byte address. Root segments are accessed through a randomizing routine. An HDAM data base consists of one VSAM entry sequenced data set (ESDS).

hierarchical direct organization. An organization of DL/I segments of a data base that are related by direct addresses and may be accessed through an HD randomizing routine or an index.

hierarchical indexed direct access method (HIDAM). Provides for indexed access to a DL/I data base in the HD organization. Segments are stored in VSAM control intervals and are referenced by a relative byte address. Root segments are accessed through a HIDAM index data base. A HIDAM data base consists of one VSAM Entry Sequenced Data Set (ESDS).

hierarchical indexed sequential access method (HISAM). Provides for indexed access to a DL/I data base. A HISAM data base consists of one VSAM key sequenced data set (KSDS) and one VSAM entry sequenced data set (ESDS).

hierarchic sequence. The sequence of segment occurrences in a data base record defined by traversing the hierarchy from top to bottom, front to back, and left to right.

hierarchical sequential access method (HSAM). The segments of a DL/I HSAM physical data base record are arranged in sequential order with the root segments followed by the dependent segments. HSAM data bases are accessed by the DOS/VSE sequential access method (SAM).

hierarchical sequential organization. An organization of DL/I segments of a data base that are related by physical adjacency.

hierarchy. (1) An arrangement of data segments beginning with the root segment and proceeding downward to dependent segments. (2) A “tree” structure.

HISAM. Hierarchical indexed sequential access method.

HS. Hierarchical sequential.

HSAM. Hierarchical sequential access method.

index data set. An ordered collection of DL/I index entries consisting of a key and a pointer used by VSAM

to sequence and locate the records of a key sequenced data set (KSDS). Organized as a balanced tree of levels of index.

index pointer segment. The segment that contains the data and pointers used to index the index target segments.

index record. A system-created collection of VSAM index entries that are in collating sequence by the key in each of the entries.

index segment. The segment in the index data base that contains a pointer to the segment containing data (the indexed segment). Synonymous with *index pointer segment*.

index source segment. The segment containing the data from which the indexing segment is built.

index target segment. The segment pointed to by a secondary index entry, that is, by an index pointer segment.

indexed segment. A segment that is located by an index. Synonymous with *index target segment*.

intersection data. Any user data in a logical child segment that does not include the logical parent's concatenated key.

key. (1) The field in a segment used to store segment occurrences in sequential order. (2) A field used to search for a segment. (3) Synonymous with *key field* and *sequence field*.

Note: A segment may or may not have a key, that is, a sequence field. All root segments, except for HSAM and simple HSAM data bases, must have keys. DL/I ensures that multiple segments of the same type that have keys are maintained in strict ascending sequence by key. The key may be located anywhere within a segment; it must be in the same location in all segments of the same type within a data base. The maximum sizes for keys are 236 alphameric characters for root segments, and 255 for all dependent segments. Keys provide a convenient way to retrieve a specific occurrence of a segment type, maintains the uniqueness and sequential integrity of multiples of the same segment type, and determine under which segment of a group of multiples new dependent segments are to be inserted. Keys should normally be prescribed for all segment types; the exception being if there will never be multiples of a particular type or if a particular segment type will never have dependents.

level. Level is the depth in the hierarchical structure at which a segment is located. Roots are always the highest level and the segments at the bottom of the structure are the lowest level. The maximum number of levels in a a DL/I data base is 15. For purposes of documentation and reference, the levels are numbered from 1 to 15, with the root segments being level number 1. Referring to Figure 8:

- Three levels are shown.
- The A segments (roots) are at the highest level (level 1).
- The C and E segments are at the lowest level (level 3).

local view. A description of the data that an individual business process requires. See *system view*.

logical. When used in reference to DL/I components, logical means that the component is treated according to the rules of DL/I rather than physically as it may exist, or as it may be organized, on a physical storage device. For example, a logical DL/I record (a root segment and all of its dependent segments grouped) might be contained on several physical records or blocks on a storage device; and because of prior insertions and deletions, the segments might be in different physical sequence than that by which they are retrieved logically for the application program by DL/I.

logical child. A pointer segment that establishes an access path between its physical parent and its logical parent. It is a physical child of its physical parent; it is a logical child of its logical parent. See also *logical parent* and *logical relationship*.

logical data base. A data base composed of one or more physical data bases representing a hierarchical structure derived from relationships between data segments that can be different from the physical structure.

logical data base record. (1) A set of hierarchically related segments of one or more segment types. As viewed by the application program, the logical data base record is always a hierarchic tree structure of segments. (2) All of the segments that exist hierarchically dependent on a given root segment, and that root segment.

logical data structure. A hierarchic structure of segments that is not based solely on the physical relationship of the segments. See also *logical relationships*.

logical parent. The segment a logical child points to. A logical parent segment can also be a physical parent. See also *logical child* and *logical relationship*.

logical relationship. A user defined path between two segments; that is, between logical parent and logical child, which is independent of any physical path. Logical relationships can be defined between segments

in the same physical data base hierarchy or in different hierarchies.

logical twin. All occurrences of one type of logical child with a common logical parent. Contrast with *physical twin*. See also *twin segment*.

MPS. Multiple partition support

multiple partition support (MPS). Multiple partition support provides a centralized data base facility to permit multiple applications in different partitions to access DL/I data bases concurrently. MPS follows normal DL/I online conventions in that two programs cannot both update the same segment type in a data base concurrently. However, two or more programs can retrieve from a data base while another program updates it. If one program has exclusive use of a data base, no other program can update it or retrieve from it.

multiple SSA. A series of segment search arguments (SSAs) included in a DL/I call to identify a specific segment or path. See also *segment search argument*.

object segment. The segment at the lowest hierarchical level specified in a particular command. See also *path call*.

option. A command keyword used to qualify the requested function.

parent. Synonymous with *parent segment*.

parent segment. (1) A segment that has one or more dependent segments. Contrast with *child*. (2) A parent is the opposite of a child, or dependent segment, in that dependent segments exist directly beneath it at lower levels. A parent may also itself be a child. Referring to Figure 8:

- A-001 is the parent of all B, C, D, E, and F segments.
- D is a parent of E; yet a child of A.
- B-02 is not a parent.
- None of the level-3 segments are parents.

path. The chain of segments within a record that leads to the currently retrieved segment. The formal path contains only one segment occurrence from each level from the root down to the segment for which the path exists.

PCB. Program communication block.

physical child. A segment type that is dependent on a segment type defined at the next higher level in the data base hierarchy. All segment types, except the root segment, are physical children because each is dependent on at least the root segment. See also *child segment*.

physical data base. An ordered set of physical data base records.

physical data base record. A physical set of hierarchically related segments of one or more segment types.

physical data structure. A hierarchy representing the arrangement of segment types in a physical data base.

physical parent. A segment that has a dependent segment type at the next lower level in the physical data base hierarchy. See also *parent*.

physical segment. The smallest unit of accessible data.

physical twins. All occurrences of a single physical child segment type that have the same (single occurrence) physical parent segment type. Contrast with *logical twins*. See also *twin segment*.

pointer. A physical or symbolic identifier of a unique target.

primary key. The data element or combination of data elements within a data aggregate that uniquely identifies an occurrence of that data aggregate. See *secondary key*.

program communication block (PCB). Every data base accessed in an application program has a PCB associated with it. The PCB actually exists in DL/I and its fields are accessed by the application program by defining their names within the application program.

program specification block (PSB). A PSB is generated for each application program that uses DL/I facilities. The PSB is associated with the application program for which it was generated and contains a PCB for each data base that is to be accessed by the program. Once it is generated, the PSB is cataloged in a core image library, and subsequently processed by a utility along with the associated DBDs to produce the updated PSB and DMBs; all of these are cataloged in a core image library for subsequent use by the application program during execution.

PSB. Program specification block.

PSBGEN. PSB generation—the process by which a program specification block is created.

qualified call. A DL/I call that contains at least one segment search argument (SSA). See also *segment search argument*.

qualified segment selection. The identification of a specific occurrence of a given segment type in a command, by using the WHERE option in the command for the desired segment. Contrast with *qualified SSA*.

qualified SSA. A qualified segment search argument contains both a segment name that identifies the specific segment type, and segment qualification that identifies the unique segment within the type for which the call function is to be performed. See also *segment search argument* and *multiple SSA*.

RAP. Root anchor point.

record. A data base record is made up of at least a unique root segment, and all of its dependent segments. See also *data base record*.

Referring to Figure 8: A-001, B-01,C-5, C-7, B-02, B-03, C-2, D, E, F,F,F constitute a data base record.

root anchor point (RAP). A DL/I pointer in an HDAM control interval pointing to a (chain of) root segment(s).

root segment. The highest level (level 1) segment in a record. A root segment must have a key unless the organization is HSAM or simple HSAM. The sequence of the root segments constitutes the fundamental sequence of the data base. There can be only one root segment per record. Dependent segments cannot exist without a parent root segment; but a root segment can exist without any dependent segments.

secondary index. Secondary indexes can be used to establish alternate entries to physical or logical data bases for application programs. They can also be processed as data bases themselves. See also *secondary index data base*.

secondary index data base. An index used to establish accessibility to a physical or logical data base by a path different from the one provided by the data base definition. It contains index pointer segments.

secondary key. A data element or combination of data elements within a data aggregate that identifies those occurrences of the aggregate that have a property named by the key. Used to locate those occurrences. See *primary key*.

segment. A segment is a group of similar or related data that can be accessed by the application program with one I/O function call. There may be a number of segments of the same type within a record.

segment name. A segment name is assigned to each segment type. Segment names for the different segment types must be unique within a data base. Synonymous with *segment type*.

segment occurrence. One instance of a set of similar segments.

segment type. Different segment types may have different lengths, but within each single type, all segments must be the same length (unless variable length segments have been specified by DBA).

Referring to Figure 8, there are six different types of segments: A through F. Synonymous with *segment name*.

sensitivity. (1) A DL/I capability that ensures that only data segments or fields predefined as “sensitive” are available for use by a particular application program. The sensitivity concept also provides a degree of control over data security, inasmuch as users can be prevented from accessing particular segments or fields from a logical data base. (2) Sensitivity to the various segments and fields that constitute a data base is controlled, on a program-by-program basis, when the PSB for each program is generated. For example, a program is said to be sensitive to a segment type when it can access that segment type. When a program is not sensitive to a particular segment type, it appears to the program as if that segment type does not exist at all in the data base. Segment sensitivity applies to types of segments, not to specific segments within a type, and to all segment types in the path to the lowest level sensitive segment type.

sequence field. Synonymous with *key field*.

sequential processing. Processing or searching through the segments in a data base in a forward direction. See also *forward*.

simple HISAM. A hierarchical indexed sequential access method data base containing only one segment type.

source segment. A segment containing the data used to construct the secondary index pointer segment. See also *secondary index data base*.

system view. A conceptual data structure that integrates the individual structures of local views into an optimum arrangement for physical implementation as a data base. See *local view*.

transaction. A specific set of input data that triggers the execution of a specific process or job.

twin segments. All child segments of the same segment type that have a particular instance of the same parent type. See also *physical twins* and *logical twins*.

twins. Synonymous with *twin segments*.

unqualified call. A DL/I call that does not contain a segment search argument.

Unqualified segment selection. The identification of a given segment type in a command without specifying a particular occurrence of that segment type (without using the WHERE option). As a general rule, unqualified segment selection retrieves the first occurrence of the specified segment type. Contrast with *unqualified SSA*.

unqualified SSA. An unqualified SSA contains only a segment name that identifies the specific type of segment for which the I/O function is to be performed. As a general rule, the use of an unqualified SSA retrieves the first occurrence of the specified type of segment. See also *segment search argument*

update intent. The scheduling intent type that permits application programs to be scheduled with any number of other programs except those with exclusive intent.

Index

F

H

I

J

L

M

N

P

R

S

U

V

Communicating Your Comments to IBM

IBM Data Language/I Disk Operating System/
Virtual Storage (DL/I DOS/VS)
Recovery/Restart Guide
Version 1 Release 7

Publication No. SH24-5030-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF form and either send it postage-paid in the United States, or directly to:

  IBM Deutschland Entwicklung GmbH
  Department 3248
  Schoenaicher Strasse 220
  D-71032 Boeblingen
  Federal Republic of Germany

- If you prefer to send comments by FAX, use this number:
  - (Germany): 07031-16-3456
  - (Other countries): (+49)+7031-16-3456

- If you prefer to send comments electronically, use this network ID:

  INTERNET: s390id@de.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

IBM Data Language/I Disk Operating System/
Virtual Storage (DL/I DOS/VS)
Recovery/Restart Guide
Version 1 Release 7

Publication No. SH24-5030-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SH24-5030-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

File Number: S370/S390-50
Program Number: 5746-XX1

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SH24-5030-00