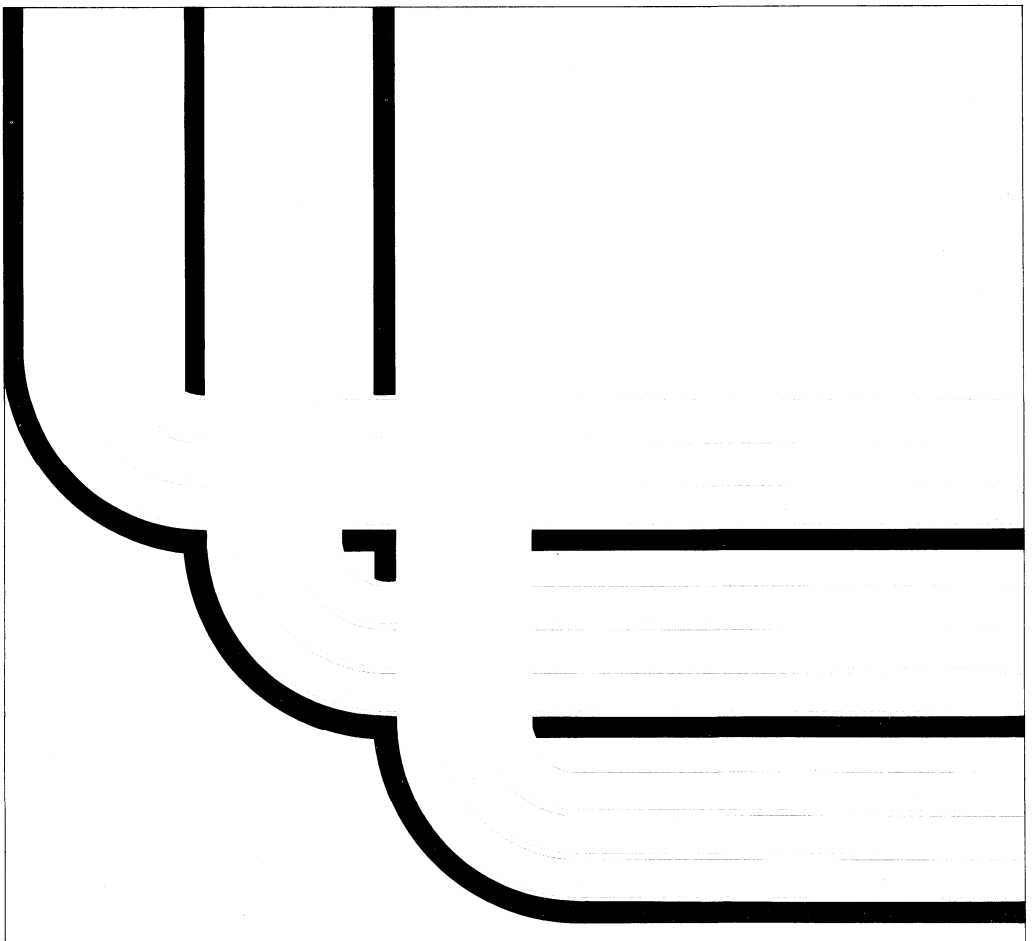
Application System/400

SC41-8079-02

Advanced Backup and Recovery Guide

Version 2



System and Application Support

Application System/400

SC41-8079-02

Advanced Backup and Recovery Guide

Version 2

Take Note!

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

Third Edition (November 1993)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 3 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. This major revision makes obsolete SC41-8079-01. Make sure you are using the proper edition for the level of the product.

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

A Customer Satisfaction Feedback form for readers' comments is provided at the back of this publication. If the form has been removed, you can mail your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899 USA

or you can fax your comments to:

United States and Canada: 800+937-3430
Other countries: (+1)+507+253-5192

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 or restricting your use of it.

Contents

Part 1. Journaling and Commitment Control

Part 2. Save-While-Active Function

Part 3. Auxiliary Storage Pools

Part 4. Checksum Protection

Part 5. Device Parity Protection

Part 6. Mirrored Protection

Part 7. Uninterruptible Power Supply

Part 8. Appendixes

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 Commercial Relations, IBM Corporation, Purchase, NY 10577, U.S.A.

This publication could contain technical inaccuracies or typographical errors.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country. IBM makes no commitment to make available any unannounced products referred to herein. The final decision to announce any product is based on IBM's business and technical judgment.

Changes or additions to the text are indicated by a vertical line (|) to the left of the change or addition.

Because the changes and additions are extensive, this publication should be reviewed in its entirety.

Refer to the “Summary of Changes” on page xxv for a summary of changes made to the .OS/400 licensed program and how they are described in this publication.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This publication contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

If you are viewing this manual from a compact disk (CD-ROM), the photographs and color illustrations do not appear.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

Application System/400
AS/400
C/400
COBOL
COBOL/400
FORTRAN/400
IBM
OfficeVision/400
Operating System/400
OS/400
PL/I
PS/2
RM/COBOL-85
RPG III
RPG/400
SQL/400
400

About This Guide

This guide provides information about advanced availability options and procedures to implement those options on the AS/400 system.

This guide does not provide:

- Information about backup and recovery strategy planning
- Basic save and restore procedures
- Procedures to recover from disk unit failures when checksum, device parity, or mirrored protection are not in effect

See the *Basic Backup and Recovery Guide*, SC41-0036, for strategy planning and disk recovery when checksum and mirrored protection are not in effect.

This guide is intended for someone who is assigned the responsibilities of backup and recovery planning and recovering the system after a failure.

You should be familiar with the information contained in the *System Operator's Guide*, SC41-8082, and the *New User's Guide*, SC41-8211, before using this guide.

If you know how to operate the system, you should be ready to use this guide to plan for, and implement the advanced availability options discussed in this guide.

This guide does not describe application recovery.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of related publications, see the "Bibliography."

Summary of Changes

This section lists major changes to the *Advanced Backup and Recovery Guide*.

Journal Management and Commitment Control

Changes were made to the journal management and commitment control information in chapters 2, 3, and 4.

Save-While Active Function

Information has been added about the save-while-active function in Chapter 5, "Save-While-Active Function."

Device Parity Protection

Information about working with device parity protection has been added to Chapter 12, "Working with Device Parity Protection" and to the topic, "Storage Planning for Device Parity Protection" in Chapter 10.

Chapter 1. Introduction to Availability and Recovery

The objective of a backup and recovery strategy for information services is to make sufficient preparations, and to establish a sufficient set of agreed upon procedures, for responding to a disaster or emergency. This will minimize the effect upon the operation of the business.

The *Basic Backup and Recovery Guide*, SC41-0036, contains information about planning a backup and recovery strategy. The *Basic Backup and Recovery Guide* provides information about why you would want to use the availability and recovery options discussed in this guide. It does not contain detailed information about the availability and recovery options discussed in this guide.

The purpose of this chapter is to provide a review of the availability and recovery options available for the AS/400* system.

Availability and Recovery Options Available for the AS/400 System

This topic provides a high-level review of the availability and recovery options for the AS/400 system. Detailed descriptions of these options are discussed later in this guide. These options include:

- Journal management
- Access path journaling
- Commitment control
- Save-while-active function
- Auxiliary storage pools
- Checksum protection
- Device parity protection
- Mirrored protection
- Uninterruptible power supply
- Battery power unit

Journal Management

Two objects unique to journal management are the journal and the journal receiver:

- The **journal receiver** (object type *JRNRCV) is an object that contains entries (called journal entries) added when a change is made to an object (for example, when an update is made to a file being journaled.
- The **journal** (object type *JRN) identifies the journaled objects, the current journal receiver, and all journal receivers that are on the system for the journal.

Using journal management is a way to recover database files. When a change is made to a journaled file, the change is first recorded in the journal receiver and then written to auxiliary storage. If the system ends abnormally before the change for the file is actually written to auxiliary storage, the change is saved in the journal receiver. The next IPL of the system, after an abnormal system end, ensures that all changes in the journal receiver are also in the files being journaled. If the database file is damaged, you can restore the file from the save media and then apply the journaled changes saved in the journal receiver.

Figure 1-1 on page 1-2 shows journal processing. Files A and B are being journaled, file C is not. Programs PGMX and PGMY use file B. When you make a change to a record in file A or B, the following occurs:

1. The change is written in the active journal receiver.
2. The journal receiver is written to auxiliary storage.
3. The record is written to the file in auxiliary storage.

File C record changes are written directly to the main storage copy of the file because it is not being journaled. Only the changes made to the journal receiver are written immediately to auxiliary storage (disk). Changes against the physical file may stay in main storage until the file is closed.

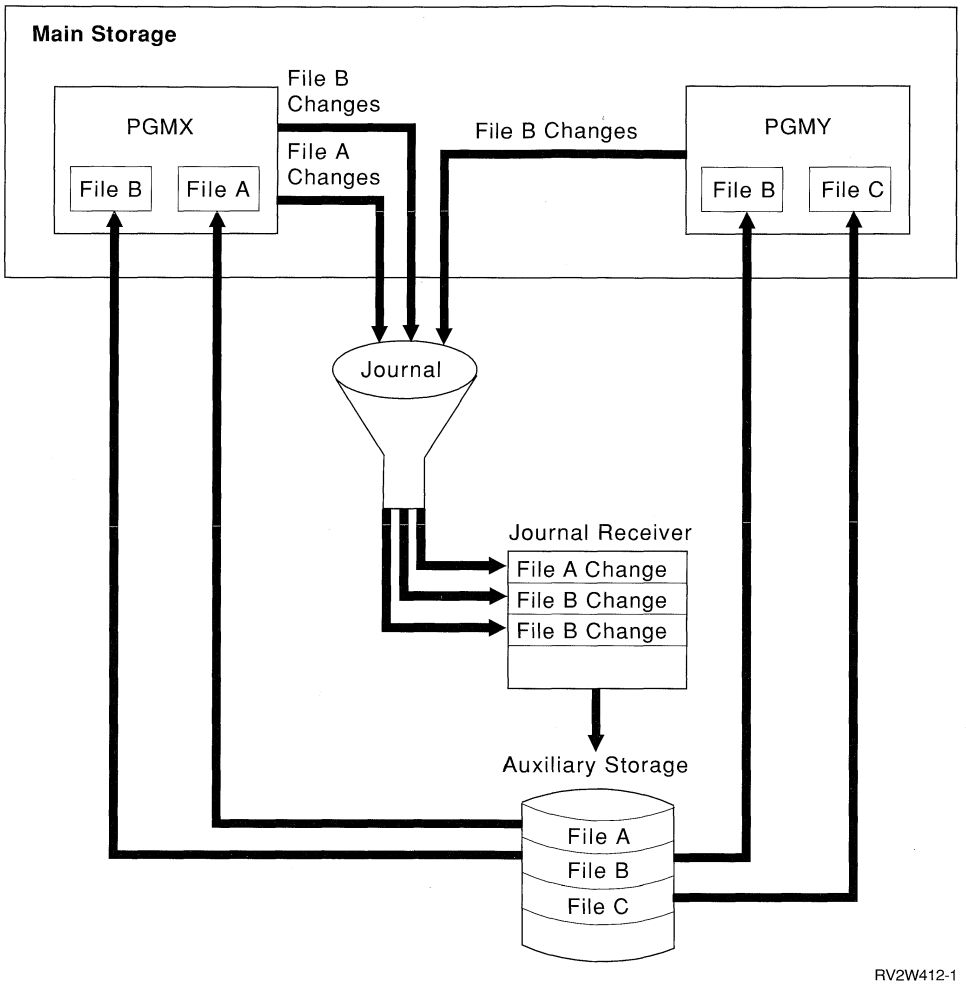


Figure 1-1. Journaling Overview

The system journals some file-level changes, including moving a file and renaming a file. The system also journals member-level changes, such as initializing a physical file member, and system-level changes, such as initial program load (IPL). Users can also add their own entries to a journal receiver to identify significant programming events or to help in the recovery of their programs. These types of entries cannot be applied or removed as entries for database files can.

Access Path Journaling

An **access path** describes to the system the order in which records are read. If access path changes are not journaled to the journal receivers, and a file has been changed and not closed properly, the system can spend a significant amount of time rebuilding the access paths during the IPL following an abnormal system end. If the decision is made to use access path journaling, and the system fails, the system may recover (rather than rebuild) the access paths automatically, using the information in the journal. This can greatly decrease the time required to recover. On the AS/400 system, journaling access paths is the primary method of reducing the time to recover access paths. Rebuilding access paths after an abnormal system end can take several hours. Recovering the access paths from a journal usually takes no more than a few seconds for each access path.

Commitment Control

Commitment control is a function that allows you to define and process changes to resources (such as database files or tables) as a single transaction.

If an application is written that does not use commitment control, data recovery after an abnormal

job or system end may be very difficult. Restoring the data after such a failure to the last completed transaction may require a user program or a data file utility (DFU) program to reverse, or back out, the transactions that are not complete. This becomes more difficult when many users are accessing the files. For example, when a job ends or the system ends abnormally, journaling assures that all records will exist in the database. Because end-user applications can require multiple changes to files in a transaction, journaling may reflect only a partially completed transaction.

You can use commitment control to ensure:

- All changes within a transaction are completed for all files affected.
- All changes within a transaction that are not complete are removed if processing is interrupted.
- Changes made during a transaction can be removed when the user program determines that it is necessary to do so. This is called a rollback operation.

The commit (COMMIT) and rollback (ROLLBACK) operations are available in several AS/400 programming languages including RPG III*, COBOL*, PL/I*, C/400*, control language (CL), and Structured Query Language (SQL).

Using the Save-While-Active Function

The **save-while-active** function provided by the AS/400 system allows you to save objects while they are being changed by another job. This function can be used along with your other backup procedures to reduce or eliminate the time the system is not available during backup processing.

However, the save-while-active function can result in more complex and potentially longer recovery times. See "Restore Recovery Procedures Considerations" on page 5-8 for recovery considerations when using the save-while-active function.

Auxiliary Storage Pools (ASPs)

An **auxiliary storage pool (ASP)** is a group of units defined from all the disk units that make up auxiliary storage. ASPs provide the means of isolating objects on a specific set of disk units to prevent the loss of data due to a disk media failure on other disk units not included in the ASP.

The system ASP (ASP 1) is created by the system and is always configured. It contains the licensed internal code, licensed programs, and system libraries. The system ASP also contains all other configured disk units that are not assigned to a user ASP.

A user ASP is created by grouping together a physical set of disk units and assigning them a number 2 though 16. ASP 1 is always reserved as the system ASP.

User ASPs can be used to isolate libraries and objects within these libraries from the system ASP. If a library exists in a user ASP, all objects in the library must be in the same ASP as the library. "Object Types Not Allowed in a User ASP" on page 6-14 shows a list of object types that are not allowed in a user ASP.

The exceptions to this rule are journals, journal receivers, and save files. These object types can be placed in user ASPs when their libraries are in the system ASP and the selected user ASP contains no libraries. However, this type of ASP is **not** recommended because of the complex recovery steps. These object types can also be placed in libraries in user ASPs.

In addition to the recovery advantage, placing libraries and objects in a separate user ASP can improve performance. In a heavy journaling environment, isolating journal receivers to a user ASP can reduce disk arm contention between the files and journal receivers, and improve journaling performance.

Note: If journal receivers and save files are not put in a separate user ASP, consider saving them regularly on tape or diskette to protect them from loss due to disk failure.

Checksum Protection

Checksum protection is a function that protects data stored in an auxiliary storage pool from being lost because of damage or a disk unit media failure. When checksum protection is in effect and a disk unit media failure occurs on a protected unit, the system automatically reconstructs the data after the disk unit is repaired.

Damage to objects can occur because of slight imperfections on a disk surface. If this occurs to objects in the ASP when checksum protection is in effect, the data is automatically re-created. This avoids having the system mark the object as damaged.

After checksum protection is started, the system automatically groups the disk units in the ASP into checksum sets during the IPL to OS/400. Space equivalent to approximately one disk unit in each set is used to store checksum data that provides protection for the user data stored on the other units in the set.

The data residing on several disk units (checksum set) is combined onto other units in such a way that, if any one of the units fails, its contents may be recovered by recombining the data on remaining units in the checksum set. The reconstructed data reflects the most up-to-date information that was on the disk at the time of the failure.

When a disk unit fails, checksum protection does not prevent the system from ending abnormally. Rather, its main advantage is that no data is lost and it helps avoid installing the entire system and loading information to disk again if the failed unit must be replaced and the data on it is lost. In contrast, normal save and restore methods allow you to recover only to the point of the last save operation and may result in entering many transactions again.

Any changes made to permanent objects residing in the ASP are automatically updated and maintained in the checksum data (associated with the space allocated to the objects).

Libraries in the system ASP should not have objects in the user ASPs. If the libraries have objects in user ASPs, then the ASP is of the old type that contains only journals, journal receivers, and save files. Checksum protection is not recommended for user ASPs that contain only journals, journal receivers, and save files.

Figure 1-2 on page 1-5 illustrates a system ASP configuration with checksum protection. In this example, assume that two of the three devices contain user application data, and the other device contains checksum data.

The assignment of units to checksum sets is shown for example purposes only. The actual assignment of checksum sets can vary because the system uses an algorithm to determine the checksum sets.

If the user data on unit 6 is lost, the system automatically reconstructs that data from unit 5 and the checksum data on unit 7 after unit 6 is replaced.

The actual implementation of checksum protection is more complex because there are multiple checksum areas spread across all units in a checksum set to distribute the disk activity more evenly over all units in a checksum set.

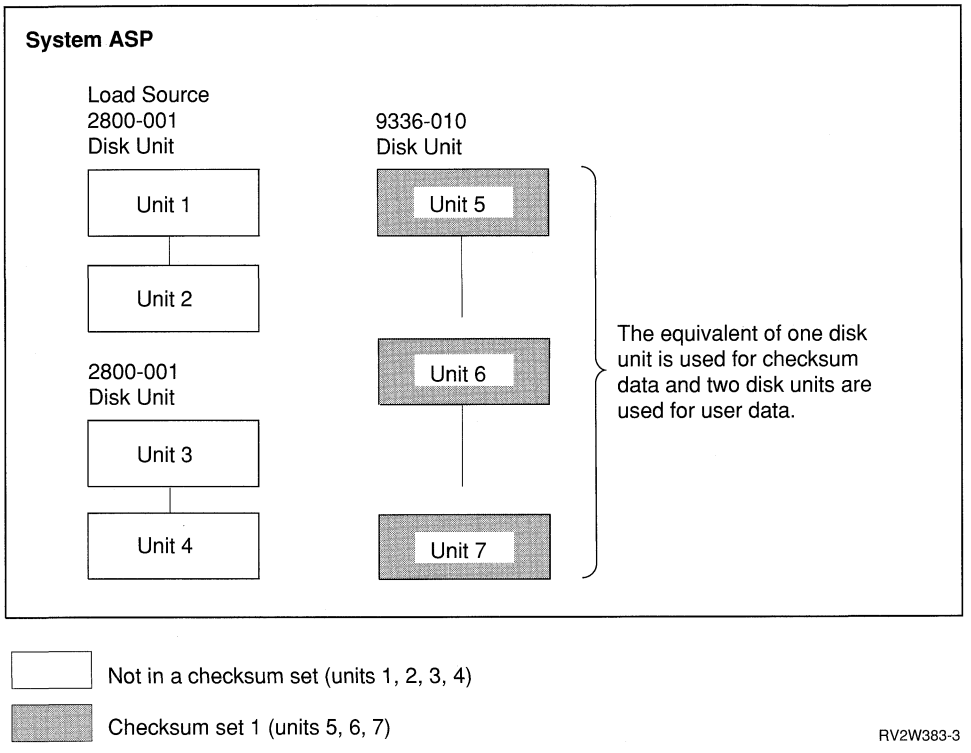


Figure 1-2. Example of Checksum Protection for the System ASP

Checksum protection should not be used as a replacement for system backup procedures. Although checksum protection can fully recover from most disk failures, it is important to be aware that there are situations where checksum protection will not be able to recover some or all of the data. For example, if more than one disk unit within the same checksum set fails and is lost or is damaged by fire, checksum protection will not be able to reconstruct any data. Even though these occurrences are rare, checksum protection should not be used as a substitute for saving the entire system on a regular basis.

Device Parity Protection

Device parity protection improves system availability by providing data protection using technology similar to RAID-5 (Redundant Array of Independent Disks) redundant power, and concurrent maintenance for single disk and power supply failures.

Device parity protection is built into some of the 9337 disk unit subsystems. The models with device parity protection use a data redundancy technique to protect the data. The parity information in the 9337 disk unit subsystems with device parity protection is spread across multiple units to improve performance.

When a failure occurs on a disk unit subsystem that has device parity protection, the disk subsystem controller automatically reconstructs the data from the active units in the 9337 disk unit subsystem. If a disk unit without device parity protection fails in an ASP that has disk unit subsystems with device parity protection, then the system is unusable until the disk unit that does not have device parity protection has been repaired or replaced.

An ASP should have device parity protection, mirrored protection, or both to ensure the system remains available after a single disk failure.

The advantage of device parity protection is that the system remains usable when a single storage unit within a disk unit subsystem with device parity protection fails. However, a decrease in performance is likely because the data on the failed unit must be reconstructed by the disk controller. Mirrored protection is better than device parity protection in this respect because a copy of data exists on two different storage units. Another advantage of device parity protection is the auto-

matic recovery of disk errors that otherwise cause objects to be damaged on system.

Device parity protection should not be used as a replacement for system backup procedures. Although device parity protection can fully recover from most disk failures, it is important to be aware that device parity protection is sometimes not able to recover some or all of the data. For example, if more than one unit within the 9337 disk unit subsystem fails or is damaged, device parity protection is not able to reconstruct any data and the entire ASP must be restored. Even though these occurrences are rare, device parity protection should not be used as a substitute for saving the entire system on a regular basis.

Device parity protection does not protect against system outages that can result from failures in other disk related hardware like a disk controller, I/O processor, or bus. If these types of outages cannot be tolerated, consider mirrored protection. Figure 1-3 illustrates a disk unit with device parity protection.

The Chapter 10, "Device Parity Protection and Mixed ASP Support" on page 10-1 has more information about capacity planning and considerations.

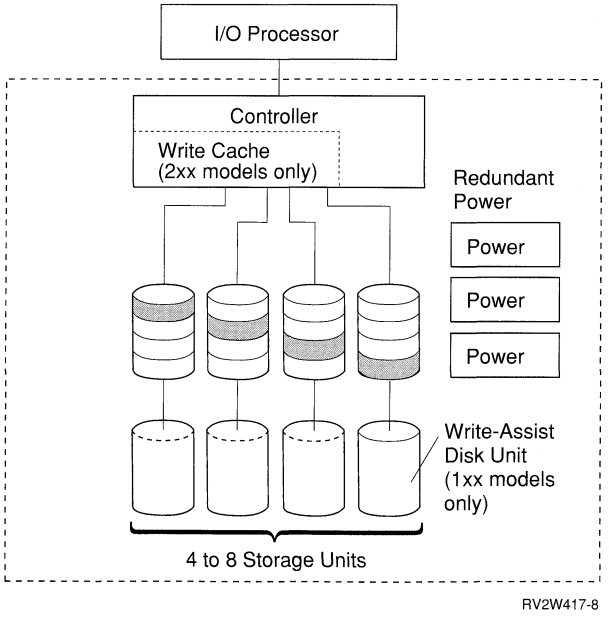


Figure 1-3. Example of Device Parity Protection for User ASPs

Mirrored Protection

Mirrored protection is a function that increases the availability of the AS/400* system in the event of a failure of a disk-related hardware component. It can be used on any model of the AS/400* system and is a part of the licensed internal code. Different levels of mirrored protection are possible, depending on what hardware is duplicated. The system remains available during a failure of a disk-related hardware component such as a disk unit, a disk controller, a disk I/O processor, or a bus, if the failing hardware component and hardware components attached to it are duplicated. For the 9406 system unit, some failed hardware components can be serviced while the system remains available.

Note: It is not possible to have bus-level protection for unit 1 on a 9406 system unit because both units of the mirrored pair must be on the same bus.

Figure 1-4 on page 1-7 shows a system that has bus-level protection.

The disk units in an ASP are automatically paired by the system when mirrored protection is started. The system pairs the disk units to provide the maximum level of protection for the current hardware configuration. Because all disk units have at least disk unit level protection, the system is protected against the failure of a single disk unit. In this example, if a controller or I/O processor failure occurs, the system continues to run.

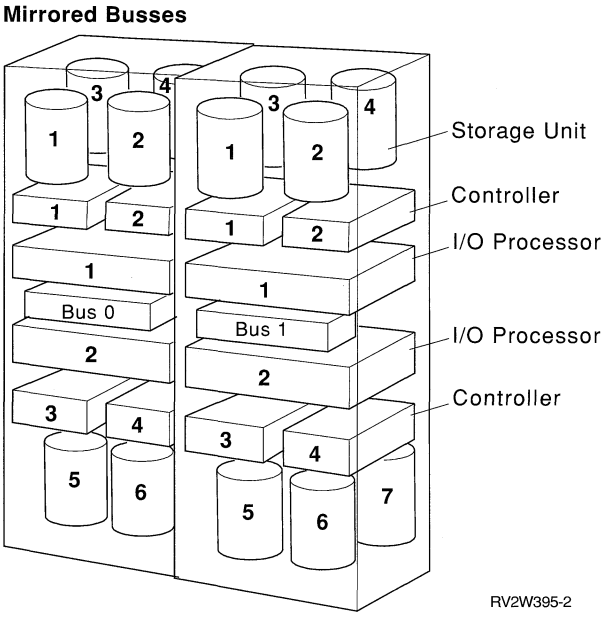


Figure 1-4. Example of a Mirrored Buses

Uninterruptible Power Supply

An uninterruptible power supply is used to provide auxiliary power to any of the system units and devices attached to the system.

Normally, an uninterruptible power supply does not provide power to all work stations. If the application handles the errors and stops, the system resource is not used to do nonproductive error recovery. Examples of using error feedback areas and error recovery routines can be found in the programming languages reference manuals.

Assuming an uninterruptible power supply is not provided to the work stations if utility power is lost, work station jobs may end abnormally if your user-written programs do not provide for this situation. However, the system remains stable. After utility power is restored, the users can sign on to the work stations. An uninterruptible power supply that provides limited support provides power to the processing unit, unit 1, and all storage controllers. The system continues to run for a specified number of minutes.

Uninterruptible power supplies vary, but Figure 1-5 on page 1-7 shows a logical view of a typical uninterruptible power supply:

With the AS/400 system, the uninterruptible power supply provides the system with the ability to:

- Continue operations during brief power interruptions.
- Provide normal ending of operations so that the next time the system performs an IPL, there is minimal recovery time. If the system ends abnormally before completing a normal ending of operations, the recovery time can be significant.

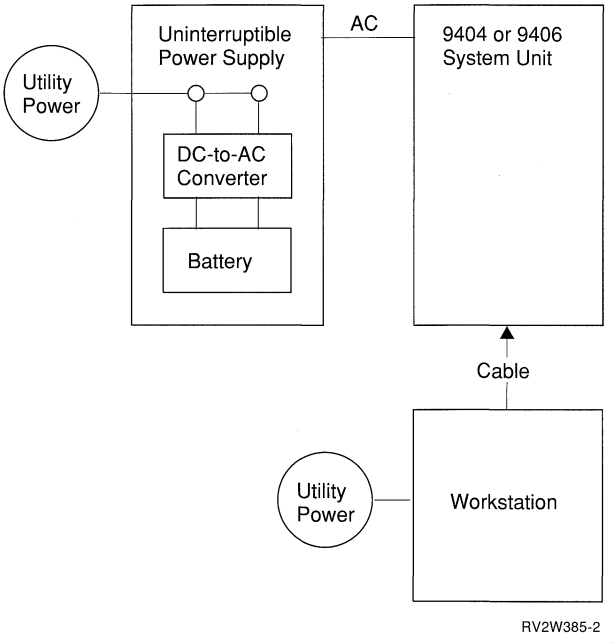


Figure 1-5. Logical View of a Typical Uninterruptible Power Supply

Optional Battery Power Unit for the 9402 and 9404 System Units

A battery power unit exists as an optional feature for the 9402 and some models of 9404 system units. For the system unit to be protected against temporary power loss, each system unit must have a battery power unit. The battery feature on the 9402 and 9404 system units supports all disk units in the system unit. The system unit has a battery with sufficient power to keep the processing unit and the disk units powered for a minimum of five minutes in the event of a loss of utility power. For an illustration of the built-in Battery, see Figure 1-6 on page 1-8.

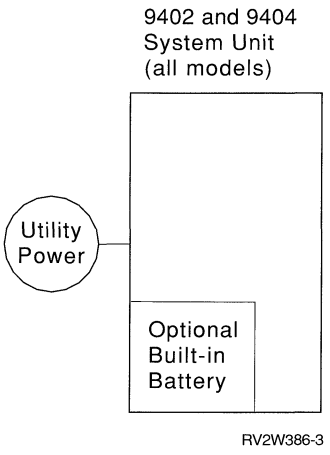


Figure 1-6. Built-in Battery Feature

Standard Battery Power Unit for the 9404 and 9406 System Units

A battery power unit exists as a standard feature for the 9404 Model E and F system units, and the 9406 Models D, E, and F system units. This standard battery provides power to the card enclosure in the 9406 system unit and to the entire 9404 system unit. All storage controllers have power including storage unit 1.

The system unit has a battery with sufficient power to keep the processing unit powered for a minimum of five minutes in the event of a loss of utility power.

Comparison of Availability and Recovery Options

A simple comparison of selected availability and recovery options for journaling, checksum protection, and mirrored protection follows. The comparison discusses the four options separately, even though journaling can also be used in combination with checksum protection or mirrored protection. Mirrored protection and checksum protection cannot be used together on an ASP.

For journaling, the comparison assumes your system has an unprotected system ASP plus a user ASP for the journal. For checksum protection, your system assumes a checksum-protected system ASP and no user ASPs. For mirrored protection, it assumes for your system that all ASPs have mirrored protection.

- Data loss after a single disk unit failure
  - Journaling – minimal loss to files being journaled if good backups are available.
  - Checksum protection – none.
  - Device parity protection – none.
  - Mirrored protection – none.
- Recovery time after a single disk unit failure
  - Journaling – potentially many hours. After the failed disk unit is repaired, the system must be reloaded, long system recoveries must be run, data must be restored, and journal updates must be applied.
  - Checksum protection – a few hours. After the failed disk unit is repaired, the system can rebuild the lost data.
  - Device parity protection – a few hours maximum. After the failed disk unit is repaired, the disk unit subsystem can rebuild the lost data.
  - Mirrored protection – none. The system continues to run while the failed disk unit is repaired on-line.
- Performance impacts
  - Journaling – varies from minimal to significant.
  - Checksum protection – varies from minimal to significant.
  - Device parity protection – varies from minimal to significant.
  - Mirrored protection – negligible. In some environments there is a performance improvement. After repair actions, there is a temporary decrease in performance.
- Planning complexity
  - Journaling – minimal. Additional disk units are needed for the user ASP.
  - Checksum protection – careful planning is necessary to determine the size of the unprotected area. Additional disk units may be needed.
  - Device parity protection – careful planning is necessary to determine the restore time. Additional disk units may be needed because of the parity information.
  - Mirrored protection – Careful planning is necessary to determine hardware and

configuration needed to provide the best possible protection.

- Setup complexity and time required
  - Journaling – minimal.
  - Checksum protection – considerable. New disk units and other hardware must be installed and the disk units must be added to the ASP. The protected and unprotected storage must be defined for all disk units in the ASP.
  - Device parity protection – minimal. New disk units and other hardware must be installed and the disk units must be added to the ASP.
  - Mirrored protection – minimal. New disk units and other hardware must be installed and the disk units that will have mirrored protection must be added to the ASP. Saving and restoring of the system is not required. Starting mirrored protection is a simple operation that does not require user intervention after it is started. The process takes a maximum of a few hours on the largest systems.
- Operational and management complexity
  - Journaling – minimal.
  - Checksum protection – negligible unless the checksum configuration or the protected-unprotected storage boundary must be changed. (The boundary between protected storage and unprotected storage is changed; the storage is not changed.) It can be difficult and time consuming if a change is required.
  - Device parity protection – negligible.
  - Mirrored protection – negligible. Mirrored protection is generally transparent to all parts of the system except at the lowest levels.
- Additional hardware required
  - Journaling – one or two storage units for the user ASP.
  - Checksum protection – one additional storage unit for each checksum set (this is a minimum of one extra storage unit for each seven existing storage units). However, to provide good checksum protection and avoid having units not in a checksum set, often more than the minimum number of extra storage units is required.
  - Device parity protection – the equivalent of one disk unit is required to contain the parity information. Storage planning is required.
  - Mirrored protection – normally twice as many storage units. For concurrent maintenance and higher availability, other disk-related hardware may be required.

Part 1. Journaling and Commitment Control

Chapter 2. Journal Management

The AS/400 system has recovery functions and commands that help recover data in a database file.

This chapter provides information about these functions and how you use them to back up and recover database files.

Journal Management

Journaling can be started and stopped very quickly. It requires no additional programming or changes to existing programs. If a database file is destroyed or damaged and the journaling function is being used, you can recover most of the changes made to the file. Optionally, you can remove changes to the file.

Two objects are associated with journal management. The following is a description of the objects and how they are used:

- A **journal receiver** (*JRNRCV). The journal receiver is an object that contains the entries (called **journal entries**) added when you make a change to journaled objects. These entries include:
  - The image of each record after it was changed (after-image)
  - Optionally, the image of each record before it was changed (before-image)
  - System-created entries
  - Any user-created entries
- A **journal** (*JRN). The journal is an object that identifies the files and access paths being protected. The system also uses the journal to record information about the journal receivers and the database files being journaled.

Journal receivers are attached to a journal on the Create Journal (CRTJRN) and Change Journal (CHGJRN) commands. Journal entries are added to the attached receivers. Journal receivers that have been attached to a journal and are still known to the system are considered to be *associated with* that journal. Use the Work with Journal Attributes (WRKJRNA) command to see a list of receivers *associated with* a journal.

Journal entries identify activity for a specific record (added, changed, or deleted) and for a save, open, or close operation for a file. Each entry includes additional control information identifying the source of the activity, including the user, job, program, time, and date. The entire image of the database record is journaled (not just the changed information).

The system adds an entry to the attached journal receiver when you change a record in a journaled database file member. Each entry is sequentially numbered and contains information that identifies:

- Type of change
- Record that has been changed
- Change that has been made to the record
- Information about the change (such as the job being run and the time of the change)

Changes to physical files (whether made directly to the physical file or through a logical file) and access paths are added to the journal receiver. The system does not journal data that you retrieved but did not change. If the logical file record format does not contain all the fields that are in the dependent physical file record format, the journal entry still contains all the fields of the physical file record format. In addition, if you are journaling access paths, entries for those access paths are added to the journal.

Figure 2-1 on page 2-3 shows journal processing. Files A and B are being journaled; file C is not. Programs PGMX and PGMY use file B. When you make a change to a record in file A or B, the following occurs:

- The change is added to the attached journal receiver.
- The journal receiver is written to auxiliary storage.
- The record is written to the main storage copy of the file.

File C record changes are written directly to the main storage copy of the file because it is not

being journaled. Only the entries added to the journal receiver are written immediately to auxiliary storage. Changes against the physical file may stay in main storage until the file is closed.

Planning a Journal Management Strategy

When planning your journal management strategy, consider the following:

- You can use journal management to do the following:
  - Recover a file member from some form of damage to the member.
  - Recover access paths after an abnormal system end.
  - Provide an audit trail of file or file member activity.
  - Analyze problems or for testing programs.
  - Provide an activity trail.
  - Review the security plans for the files.
  - Minimize recovery time when restoring from the save-while-active media.
- Do the following for your important files being journaled:
  - Attach two (dual) journal receivers to the journal.
  - Use save files to save the files. (The operation of copying a file using the CPYF command is not journaled in the journal, but the save operation to a save file is.)
  - Back up the journal, the journal receivers, and the journaled files on tape or diskette.
- Determine who is responsible for the following:
  - Recovery functions
  - Authorization to use the journal
  - Use of the journal
- Other items to consider include:
  - Legal considerations or restraints, such as who is allowed to look at the data
  - Business practices that influence your use of journal management, such as auditing or security requirements
  - Amount of system space to use journal management
  - Effect of journal management on the performance of your system

After considering the above, you may come to these conclusions:

- The recovery functions provided by the save and restore operations are adequate for your needs.
- You will use journal management for your recovery needs on some files.
- You will use journal management for reasons other than recovery.

Considerations for Planning a Journal Management Strategy

Consider the following when planning a journal management strategy:

- Which physical files will be journaled?
- What attributes will you use to journal those files? (For example, should you journal only after-images of the record changes or before-images and after-images.)
- Which access paths for database files will be journaled?
- Do you need multiple journals?
- Will you journal multiple files to the same journal?
- How often will you attach new journal receivers to the journal?
- How often will you save the journal receivers?
- How often will you save the physical files?
- What information will you make available for journal users?
- How will physical files be recovered?
- Will application programs place entries directly into the journal?
- What information should you record with user-created journal entries?

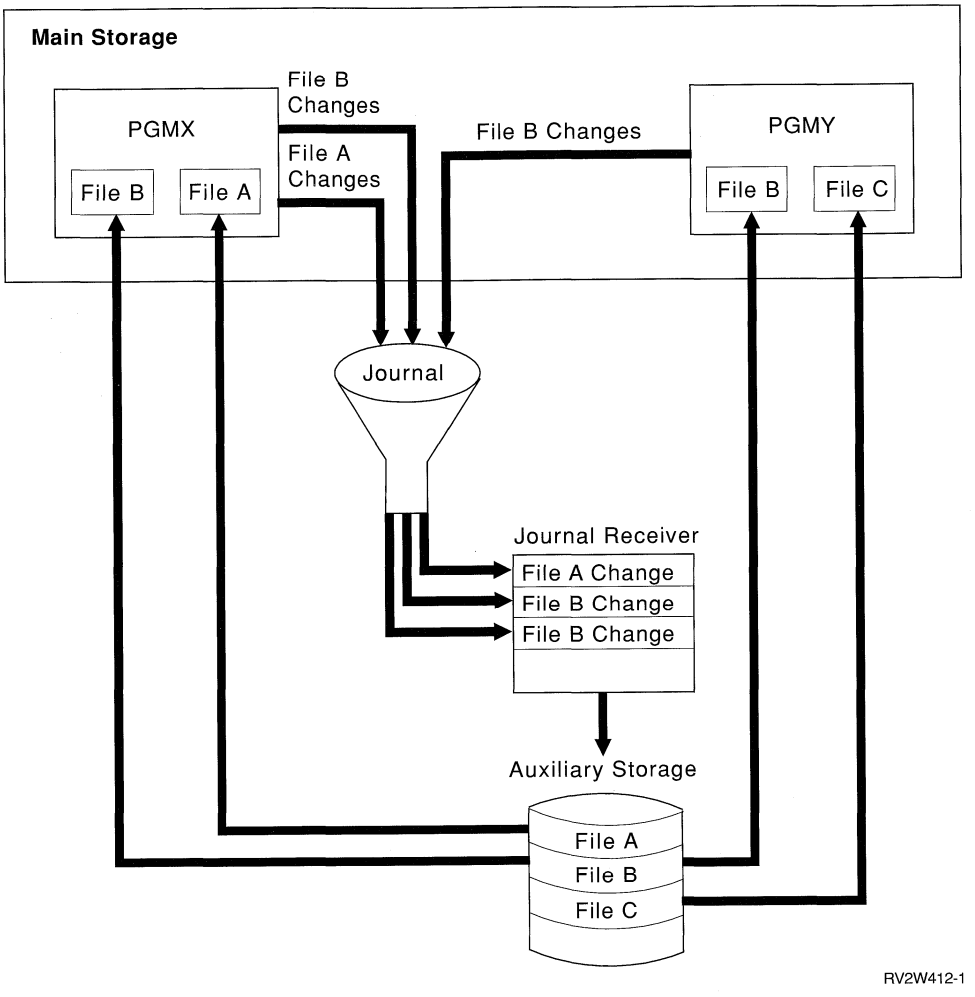


Figure 2-1. Journaling Overview

Journaling Performance and Space Considerations

The following considerations apply to system performance and space when journaling is used:

- Space requirements increase if both before- and after-images are journaled, but performance is minimally affected.
- Space requirements increase if you journal access paths. The actual increase is application-dependent. The minimum increase occurs when you make all changes to the same primary value of the access path (for example, when the access path uses the date added field as its primary value and you add many records on the same day).

  Journaling access paths has a minimal effect on performance. The system packages before- and after-record images and any access path changes into a single write operation to disk. Journaling files and access paths help reduce the time it takes to start your applications again after an abnormal system end.
- To save space, you can use the OMTJRNE parameter on the Start Journal Physical File (STRJRNPF) command to leave out the open and close entries that are added to the journal receiver when a file is being opened or closed. Depending on the application, this can be a space and a performance advantage. When you omit the open and close entries, you cannot:
  - Use the TOJOBO and TOJOBC parameters on the APYJRNCHG and RMVJRNCHG commands.
  - Use the journal to determine which users have opened the file but have not made changes.

The *Programming: Control Language Reference*, SC41-0030, has more information about these commands.

- Space requirements also increase when the number of files and/or the number of file access paths being journaled increases.
- As the number of files being journaled increases, the general performance of the system, including the IPL, is slower. (This depends to some extent on the number of changes being made to the journaled files.) The effect on performance of the IPL occurs primarily after an abnormal system end and depends on the number of files being journaled, and the number of access paths that were actively being changed when the abnormal system end occurred.
- Space requirements increase when you use two receivers. The second receiver does not require the same performance overhead as the first receiver because the write operations are done at the same time.
- Using the force-write ratio (FRCRATIO) parameter on physical files to force data to be physically written to disk decreases performance. You can discontinue using the FRCRATIO parameter on physical files being journaled and for the logical files over the physical files. The journal entries for a record-level change are written to auxiliary storage before they are written to the database file.
- If sequential-only processing (SEQONLY(*YES)) is specified for files being journaled, records added or inserted in the file are blocked. When the block is full, the records are added to the journal with one write operation and to the database file. The records appear in the journal as consecutive journal entries.

  You can decrease the number of records in a block or delete blocking altogether for recovery reasons using the Override with Database File (OVRDBF) command. However, this may make the application program run slower. Sequential-only processing is specified by the OVRDBF command, or, in some cases, the high-level language implicitly specifies sequential-only processing. For a description of when high-level languages specify sequential-only, see the appropriate high-level language manual.

To determine the approximate size of a journal receiver, estimate that the system-created portion of a journal entry is 155 bytes. If the record size in your files is, for example, 115 bytes, the journal entry would be approximately 270 bytes. (These values are approximate because the system-created portion of the journal entry varies according to the journal entry type.) The number of bytes used for the storage of a journal entry in a receiver is not the same as the number of characters required when the journal entry is shown.) Use the following formula to estimate the size of a receiver:

```
Journal receiver size per day =
          (155 bytes + average record size)
          x (number of changes per day)
```

If you specify only after-images and if you have 10 000 changes per day and an average record length of 115, the journal receiver could reach approximately 2 700 000 bytes. (This size does not include any user-created entries.) Remember that any formula to determine the size of a receiver provides only an estimate of the size. You can determine the actual size of a receiver by using the Display Object Description (DSPOBJD) or the Display Journal Receiver Attributes (DSPJRNRCVA) command. As with other objects, the system allocates space as needed in increasingly larger allocations. If the size of a journal receiver is shown immediately after creation, it shows the initial allocation, even though no entries exist yet.

If you use before- and after-images, the journal receiver size does not necessarily double because not all journal entries contain a record image and not all records have both before- and after-images. For example, if all operations to the journaled files are write operations, the journal receiver size does not double (no before-images exist). If the operations include delete operations, the size increases but does not double. If the operations are all update operations, the receiver could double in size.

Journaling access paths requires more auxiliary storage than journaling physical files with after-images only. Several journal entries that cannot be displayed are also added in the journal when journaling access paths. The amount of additional storage needed depends on how many access

paths are being journaled, and how the journaled access paths are updated by various applications. Applications that update, delete, or add keys in an ascending or descending order use less auxiliary storage than applications that update, delete, or add keys in random order.

Managing Journal Receivers

Following are examples of how to manage journal receivers:

- Create all the physical files that are journaled in a particular application, the journal, and the associated journal receivers in the same library.

  At the end of each day, run a CL program that:

  1. Creates a new journal receiver.
  2. Attaches it to the journal.
  3. Saves the entire library with the SAVLIB command.

  Only one receiver is used throughout the day. Leaving the detached journal receivers on-line provides an easier recovery because the receivers do not have to be restored. However, if you have insufficient space, you can save the detached receivers and then delete them.

- Create all the physical files that are journaled in a particular application, the journal, and the associated journal receivers in the same library, and use the SAVCHGOBJ command instead of the SAVLIB command.

  At the end of each day, run a CL program that:

  1. Creates a new journal receiver.
  2. Attaches it to the journal.
  3. Saves the previous receiver and saves the changed objects (SAVCHGOBJ command) in the library, excluding the journaled files and the attached receivers.

  You can then save the journaled files periodically (for example, weekly) by using the SAVCHGOBJ command and specifying OBJJRN(*YES) so that you have a starting point for recovery of the files.

- Create all the physical files that are journaled in a particular application and the journal library, and then create the journal receivers in another library.

  At the end of each day, run a CL program that:

  1. Creates a new receiver.
  2. Attaches it to the journal.
  3. Saves the library with the journal receiver.

  Then save the files on a regular, but not a daily basis. (Use the SAVLIB command to save the library that contains the files.)

  If you have sufficient space, leave the journal receivers on-line for other uses, such as audit trails or debugging. If you do not have the space, and do not need the receivers for any other purposes, save the detached receivers (specify STG(*FREE)), and if necessary, delete them.

- To improve performance, as well as recoverability, consider using user ASPs for the journal receivers. One possible configuration is to create the journal and the journaled files in the same ASP (the system ASP or a user ASP) and the associated journal receivers in a library in a different ASP. This minimizes write contention between the journaled files and the journal receivers attached to the journal. Additionally, a disk failure in the ASP containing the journaled files leaves the journal receivers intact for recovering data.

- Journal receivers can be saved while they are attached to a journal. However, the copy on the media is not complete because entries can be added to the receiver after it is saved. In general, ensure you have a complete copy of the receivers by saving them as soon as they are detached from the journal.

- You should also place journal receivers in a different ASP if your journal entries are critical.

  If the data in your files is extremely critical and difficult to re-create, attach two receivers to the journal and place each receiver in a different ASP. If one of the journal receivers becomes inoperable, the system continues to use the other one. The CL programs to control the journal receivers are the same as described in the preceding examples except that two journal receivers are attached and detached with each CHGJRN command. These journal receivers can be saved on the

same volume or on different volumes for more reliability.

You can then perform file recovery using either one of the journal receivers if they are both usable.

In deciding how you are going to manage your journal receivers, you should also keep the following points in mind:

- If it is seldom necessary for you to recover a file, save the file less frequently, and save the journal receivers on a regular basis.
- If you use the CHGJRN command and specify JRNRCV(*GEN), you do not have to explicitly create a receiver.
- You do not need to delete the journal receivers as soon as they are saved. If you have sufficient space on your system for more than one receiver, keep the receivers on-line so they are available for use in producing audit trails, for problem analysis, and for recovery purposes. In this case, you do not have to restore the receivers when you need to do a file recovery.

  However, if your storage space is limited, save the journal receivers with the storage freed rather than deleting them to save space. Saving journal receivers with storage freed also facilitates restore and recovery procedures. (The system keeps track of all receivers that have not been deleted.) If you do not have enough space on your system, delete the receivers after they have been saved.
- If you anticipate transferring an application to another AS/400 system, saved journals and journal receivers can be restored only to the libraries they were originally created in. Move the application using the Save Library (SAVLIB) and Restore Library (RSTLIB) commands.
- If you anticipate using the save-while-active function in your backup and recovery strategy and the applications that are making changes to the objects being saved are not ended during checkpoint processing, consider journaling all files that are required for the application. This greatly reduces the amount of recovery required after restoring objects from the save-while-active media.

  After the save-while-active operation is complete, plan to save all associated journal receivers for the files being saved. This allows you to apply or remove application changes to an application boundary after restoring the objects from the save-while-active media. For more information about the save-while-active function, see Chapter 5, "Save-While-Active Function."

Managing a Journal

The frequency with which you perform the following steps and the methods you use depends on how you decide to manage journaling.

1. Attach a new journal receiver.
2. Save the detached journal receiver.
3. Periodically save the journal.
4. Periodically save the files and all associated logical files.
5. Save a file after you add a member so that it can be recovered if needed.

Do the above steps after you complete a recovery operation using journaling. You should periodically save the journal and all logical files. Save the detached journal receiver after you run a CHGJRN command.

Journaling a Physical File

When you specify that a physical file is to be journaled, the system ensures that an active journal receiver exists, that all changes to members of the file are journaled through the same journal, and that new members added to the journaled file are automatically journaled. Each time a member is added to a file, the file should be saved. You use the Start Journaling Physical File (STRJRNPF) command once per file to have those changes journaled for every job that subsequently uses the file. When journaling of a file ends, an entry is added to the journal receiver unless the receiver is damaged.

If you use user ASPs in the recommended way, by creating the library in the ASP, then the journals and journaled files must be in the same ASP. However, if you use the other method of creating the library in the system ASP and the journals, journal receivers, and save files in a user ASP,

then the files to be journaled must be in the system ASP. The journals can be in either the system ASP or isolated in a user ASP.

Changes for several physical files can be journaled to the same journal, but changes for any one physical file can be journaled to only one journal at a time. If a logical file is dependent on several physical files, it is recommended that all the physical files be journaled to the same journal.

In general, database source files should not be journaled. If you use the Start Source Entry Utility (STRSEU) command to update a member, every record in that member is considered changed and every record is journaled to the journal. However, if changes to a source file are critical, you can journal the file in the same manner as data files.

It is recommended that you do not journal changes to IBM-supplied database files. These files are sometimes created and managed differently than user-created files. The system does not assure the recovery of these files even though all recovery activity normally succeeds.

Saving Journaled Files

Save the physical files after you start journaling them and after you have added members to them. Logical files dependent on the physical files, should also be saved in case the physical file becomes damaged. You do not need to save access paths being journaled in order for the system to recover those access paths after an abnormal system end. However, if you save the access paths being journaled (when saving files), the system does not have to rebuild the access paths after restoring the files. You can use the Save Changed Objects (SAVCHGOBJ) command and specify OBJTYPE(*FILE) OBJJRN(*YES), or you can use the Save Object (SAVOBJ) or Save Library (SAVLIB) command to save the journaled files.

When you start journaling a physical file, the system assigns a unique **journal identifier** (JID) to every member in the file. This unique journal identifier is part of every journal entry added to the journal receiver for a given file member. This identifier is the way that the journal entry is associated with the corresponding journaled object. The copy of the physical file on the save media that was saved before it was journaled does not have the journal identifier saved with it. Therefore, if this copy of the file is restored to the system, the journal entries cannot be associated with the file and cannot be applied. This is why it is critical to save the journaled file after journaling is started, and every time a member has been added to it. This ensures that the journal identifiers are saved with the file members.

Using One or More Journals

Using one journal allows you to establish standard operational procedures for files with similar backup and recovery needs, and makes it easier for you to manage the journal and the journal receivers. If you are journaling to a user auxiliary storage pool (ASP), a single journal receiver for each ASP provides the best performance. Journal receivers do not need to be in the same ASP as the journal and files being journaled.

You can specify after-images for some files, and simultaneously specify both before-images and after-images for other files, even though they use the same journal. All files opened under commitment control using the same commitment definition within a job must be journaled to the same journal. For more information on commitment control, see the topic Chapter 4, "Commitment Control" on page 4-1.

Use more than one journal if one or more of the following are true:

- Some files have different backup and recovery requirements. For example, if the files for your major application programs are in separate libraries and the libraries are saved on different schedules, journal the files for each application program to separate journals.
- The security of certain files requires that you exclude their backup and recovery procedures from the procedures for other files.
- Some files must be treated separately from others because they need to be transferred from one system to another or replaced on the system periodically.
- You are journaling the files for different reasons, such as some for audit or activity trail purposes, and some for debugging purposes.

- You have many application programs that use different files. Journaling all the files to one journal decreases performance because of the journal entries being added to the journal receiver at the same time. However, if you have a journal receiver isolated on a single ASP, you may achieve better performance with a single journal.
- Job accounting must be done to the QACGJRN journal in library QSYS. You should not mix database file journaling with the accounting journal. For more information about job accounting, see the manual *Programming: Work Management Guide*, SC41-8078.
- Security auditing must be done to the QAUDJRN in library QSYS. You should not mix database journaling with the security auditing journal. For more information about security auditing, see the manual *Security Reference*, SC41-8083.

Naming Journal Receivers

Use generic names to help identify the use of each receiver. When you use the CHGJRN command, you can specify that the system create and name the new receivers by specifying JRNRCV(*GEN). In this case, the system adds a number to the name or increases by one the number already in the attached journal receiver name.

The system uses the following rules to adds a number to the name of a currently attached journal receiver:

- If the name of the current journal receiver does not end in a number and does not exceed 6 characters, the system adds 0001.
- If the name of the current journal receiver exceeds 6 characters, and the last four characters are not numeric, the system truncates the name to 6 characters and adds 0001.
- If the name of the current journal receiver ends in a number and does not exceed 6 characters, the system adds 1.

  For an example, see Table 2-1.

Table 2-1. System-Created Journal Receiver Names

Current Journal Receiver Name	System-Created Journal Receiver Name	Comments
A	A0001	
ABCDEF	ABCDEF0001	
ABCDEFG	ABCDEF0001	(The last character of the current name is dropped because it exceeds 6 characters.)
A0001	A0002	
A1	A2	
A9	A10	
ABCDEF1	ABCDEF0001	(The last character of the current name is dropped because it exceeds 6 characters.)
ABCDEF9999	Error	Adding 1 to 9999 causes an overflow condition.

Naming Journals and the Files being Journaled

To ensure that journaling is automatically started again, the journals must be restored before the files being journaled. If the journals and associated files are in the same library, then the *system* automatically restores the objects in the correct order.

When the journals and associated objects are in different libraries, *you* must ensure that the objects are restored in the correct order. You can name the libraries for the journals, database files, and journal receivers in such a way to ensure that the objects are restored in the correct order. For example, starting the name of the library for the journal with a special character, such as #, $, or @ will ensure that the library for the journal is restored before the library for database files.

Using Dual Journal Receivers

You can specify that a journal have two journal receivers attached to it at the same time. When you use two receivers, all entries are added to both receivers. The use of two receivers provides more protection from damage and should be considered if the changes to your files are not easily reconstructed. If one receiver is damaged, the system continues to write changes to the other receiver.

The following rules are used to name two new journal receivers if JRNRCV(*GEN *GEN) is specified.

- If you currently have two journal receivers, the name of the first journal receiver is used for the first name, and the name of the second journal receiver is used for the second name. For example, if the names of the current journal receivers are A1 and B5, the names of the two new receivers are A2 and B6.
- If you currently have a single journal receiver attached, the system uses the same naming convention for both new journal receivers. For example, if the current journal receiver is A1, the two new journal receivers are A2 and A3.

If your journal entries are critical, you can have one receiver in one ASP and the other receiver in a different ASP. If one ASP fails, one receiver would still exist.

Journal Receiver Chains

Journal receivers associated with a journal (presently or previously attached to the journal) are linked in one or more **receiver chains**. Each journal receiver, except the first one, has a previous receiver that was detached when the current receiver was attached. Each journal receiver, except the one currently attached, also has a next receiver. If you use dual receivers, each dual journal receiver, except the first, has two previous receivers and each dual journal receiver, except the last, has two next receivers.

Figure 2-2 illustrates the process by which journal receiver chains are created. If you leave the previously attached receivers RCVA7 through RCVA9 online, you can use them to apply changes, to remove changes, or to display journal entries without restoring them first.

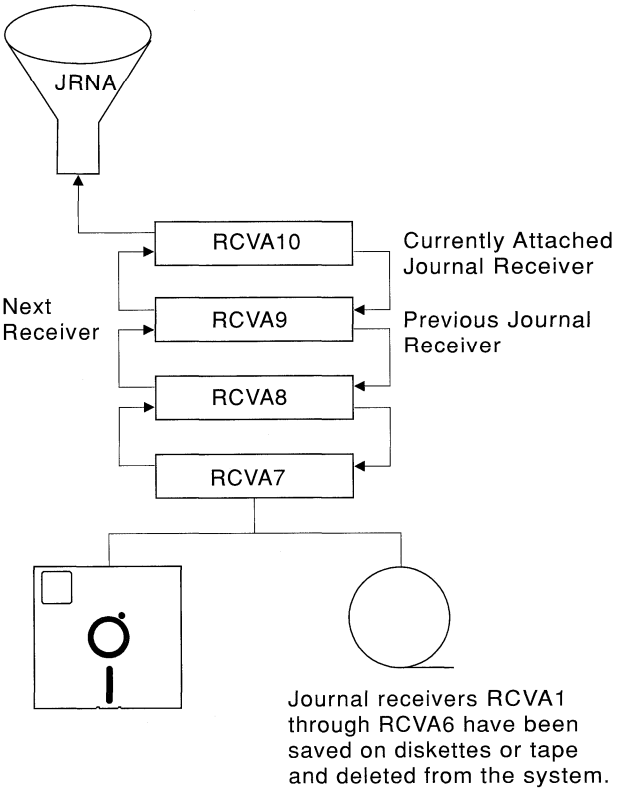


Figure 2-2. Creating a Journal Receiver

If a complete copy of a receiver is missing in a chain of journal receivers linked together in the previously described relationship, the result is a **chain break**. Receiver chain breaks should be avoided. A receiver chain break indicates that any changes made between the last entry in the last receiver in one chain and the first entry in the first receiver in the next chain are not available in any journal receiver on the system.

A set of receivers for a journal that has one or more receiver chain breaks has multiple receiver chains. Receiver chain breaks result from the following:

- You restored an old journal receiver and its next receiver is not on the system.
- A journal receiver was saved while it was attached, a partial receiver is restored, and no complete copy of the receiver is on the system or restored.
- A receiver that has not had its storage freed by a save operation is restored, and the next receiver has had its storage freed by a save operation.

- Both receivers in a set of dual receivers are unusable, destroyed, or damaged.
- The journal is restored. All journal receivers associated with the previous copy of the journal (before the journal was deleted and restored) will not be in the same receiver chain as the currently attached journal receiver.
- A damaged or destroyed journal receiver was deleted from the middle of a chain, and dual receivers do not exist on the system.
- A journal receiver from another system is restored. The journal receiver will be associated with a journal at restore time if the associated library and journal on the source system had the same library name and journal name as the library and journal on the target system.

The APYJRNCHG, RMVJRNCHG, RCVJRNE, DSPJRN, RTVJRNE, and CMPJRNIMG commands cannot be used across multiple receiver chains. (If multiple receiver chains exist, run a separate command for each receiver chain.)

Changing Journal Receivers

The Change Journal (CHGJRN) command is used to make the following changes:

- Change receivers for a journal.
- Reset the sequence number.
- Change the journal threshold message queue.
- Change text for the journal.

Use the CHGJRN command to change the journal receiver for a given journal at any time, even if the files being journaled are open. As a new journal receiver is attached to the journal, the previously attached journal receiver is automatically detached. You can periodically change the journal receiver without interrupting normal system functions and without affecting the use of the files being journaled. No database changes are lost. However, a significant performance penalty is associated with each open file. The overhead is similar to the cost of closing a file. The system creates journal entries in both receivers indicating that a new receiver has been attached. Changing the journal receivers allows you to save and delete the detached journal receiver.

Avoid changing receivers during peak production periods because of the possible effect on performance for other jobs. Changing to a new receiver can cause the system to force many database file changes to auxiliary storage. Thus, the time to change receivers can depend on the status of the files. It is best to schedule receiver changes during low production periods.

When you change from one receiver to another and the SEQOPT is *CONT, the sequence number for the first journal entry in the new receiver is one greater than the last sequence number in the detached receiver. You can only reset the sequence number when changing journal receivers. Use SEQOPT(*RESET) to reset the starting sequence for the first journal entry in the new receiver to 1. You can only reset the sequence number when all changes are forced to auxiliary storage for all journaled objects and commitment control is not active for the journal.

In general, you should reset the sequence number only when the sequence number becomes extremely large. Resetting the sequence number has no effect on the naming of the journal receivers. See "Naming Journal Receivers" on page 2-8.

When you use JRNRCV(*GEN) on the Change Journal (CHGJRN) command, the system creates the new receiver with the same attributes as the currently attached receiver, and in the same library. These attributes include the owner, private authorities, public authority, object auditing, ASP identifier, threshold, and text.

Use the CHGJRN command to change from dual receivers to a single receiver, or from a single to dual receivers. If dual receivers are used, additional protection is provided against the loss of entries. If you place one of the receivers in a separate user ASP, you are also protected against a disk unit media failure.

Note: Entries are placed in both receivers if you are using dual receivers. Best performance is achieved by placing each receiver in a separate user ASP.

When a journal receiver is detached from a journal, the receiver cannot be reattached to any journal. The receiver can be used only with system functions such as the save and restore operations, or on the following commands:

- Apply Journaled Changes (APYJRNCHG)
- Remove Journaled Changes (RMVJRNCHG)
- Display Journal Receiver Attributes (DSPJRNRCVA)
- Compare Journal Image (CMPJRNIMG)
- Retrieve Journal Entry (RTVJRNE)
- Receive Journal Entry (RCVJRNE)
- Work with Journal (WRKJRN)
- Display Journal (DSPJRN)

Deleting Journals and Journal Receivers

If you no longer need a journal receiver, delete it using the Delete Journal Receiver (DLTJRNRCV) command. If you delete a journal receiver, you must restore it from a saved copy before the recovery functions (Apply Journaled Changes (APYJRNCHG) command or Remove Journaled Changes (RMVJRNCHG) command) can use that receiver. The system does not allow you to delete a journal receiver attached to a journal.

You can also delete a journal receiver using the Work with Journal Attributes display to select the Work with Journal Receiver Directory display.

Journal receivers must be deleted in the same order they were attached to a journal except that a damaged or inoperable receiver can be deleted regardless of this restriction. If an attached receiver is damaged, you must detach it (CHGJRN command) before you delete it. You can also delete a journal receiver in the middle of a chain if the corresponding dual receiver still exists and is usable (was not saved with storage freed).

You should ensure that the journal receivers are not deleted without first being saved. Display the journal receiver directory using the Work with Journal Attributes (WRKJRNA) command to determine which receivers have been saved. A date of 00/00/00 in the *Save date* column indicates that a journal receiver has not been saved.

If you attempt to delete a receiver that has not been saved, the system sends the inquiry message CPA7025. If the reply to the message is ignore (I), the journal receiver is deleted. If the reply is cancel (C), the operation is not completed. You can use the system reply list to specify the reply the system sends for a particular job.

Each journal on the system causes additional time and resource to be used at each IPL after an abnormal end. If you delete a journal, you must restore it from a saved copy or create a new journal with the same name in the same library. Then restore all the needed receivers before using the APYJRNCHG or RMVJRNCHG command to apply or remove changes with that journal. Receivers already on the system need to be saved and restored to become associated with the journal.

If you no longer need a journal, you should delete it using the Delete Journal (DLTJRN) command. The system does not allow a journal to be deleted if files are being journaled to it, or if commitment control is active and the journal is associated with it.

If you no longer need a journal and its associated receiver, perform the following steps:

1. End journaling of all logical files with the ENDJRNAP command.
2. End journaling of all files associated with the journal using the ENDJRNPF command.
3. If commitment control is active, and the journal is associated with it, end commitment control with the ENDCMTCTL command.
4. Delete the journal using the DLTJRN command.
5. Delete the journal receiver using the DLTJRNRCV command.

Saving and Restoring Journals and Journal Receivers

The following commands are used to save or restore a journal:

- Save Object (SAVOBJ)
- Save Changed Object (SAVCHGOBJ)
- Save Library (SAVLIB)
- Restore Object (RSTOBJ)
- Restore Library (RSTLIB)

Journal receivers attached to a journal can be saved without being detached from the journal.

Journal receivers no longer attached can be saved with the storage freed, restored, or deleted.

Restoring Journal Objects in the Correct Order

Journaling for a file cannot be started again by the system after a restore operation unless the journal and associated objects are restored in the correct order. Additionally, the same relationship applies to the ASP for the journal and the ASP for the journaled files as described in "Journaling a Physical File" on page 2-6.

The journal and associated objects must be restored in the following order:

1. Journals
2. Based-on physical files
3. Dependent logical files
4. Journal receivers from newest to oldest

The SAVLIB command with the value *NONSYS or *ALLUSR specified on the LIB parameter saves the libraries in alphabetic sequence. The RSTLIB command with the value *NONSYS or *ALLUSR specified on the SAVLIB parameter restores the libraries in the order they were saved.

If the journals and associated objects are in the same library, then the *system* automatically restores the objects in the correct order.

If the journals and associated files are in different libraries and they are saved using SAVLIB LIB(*NONSYS or *ALLUSR), then the system cannot determine if the journals are restored before the files during the restore operation.

The based-on physical files must be restored before the dependent logical files. If the based-on files are not in the same library as the logical files, the system cannot determine if the based-on files are restored before the dependent logical files.

When the journals and associated objects are in different libraries, *you* must ensure that the objects are restored in the correct order. You can name the libraries for the journals, database files, and journal receivers in such a way to ensure that the objects are restored in the correct order. For example, starting the name of the library for the journal with a special character, such as #, $, or @ will ensure that the library for the journal is restored before the database files.

Inoperable Journal Receivers

If you have specified journaling for any files, the system ensures that you have corrected problems that affect journaling before continuing with operations on those files. If the attached journal receiver becomes inoperable, the database operation that writes a journal entry is interrupted and the system sends an inquiry message notifying the system operator. The operator can use the CHGJRN command to change the journal receiver. The user can then respond to the inquiry message. A receiver can become inoperable if the receiver is damaged, the maximum sequence number has been reached, or there is no more space.

If dual receivers are attached to a journal and one receiver becomes inoperable, the system continues placing journal entries in the operable receiver and discontinues placing them in the inoperable receiver. The system sends a message notifying the operator that one of the receivers is inoperable. The system operator can use the CHGJRN command to replace the inoperable receiver, which causes both currently attached receivers to be replaced.

Journal Entries

Every journal entry is stored internally in a compressed format and must be converted by the operating system to an external form before it can be shown to the user.

You can use the Display Journal (DSPJRN) command to display or print the journal entries, or write them to a database output file. You cannot directly access the journal entries. Not even the security officer can remove or change journal entries in a journal receiver. You can use these journal entries to help you recover your files or analyze changes made to the files.

You can write journal entries to tape or send them to another system using the Receive Journal Entry (RCVJRNE) command. This allows a user-defined program, called an **exit program**, to receive journal entries from the RCVJRNE command. The program writes the entries to tape

or to an OS/400 intersystem communications function file (ICF) file that sends them to a backup system. When added to a backup source, you can use the entries to update the primary database or a backup copy. You cannot use these retrieved entries with system-supplied recovery commands (APYJRNCHG and RMVJRNCHG) to update your files because the RCVJRNE command converts the entries to their external form. You must write your own program to apply the changes contained in the entries to the database files.

The RCVJRNE command supports most of the parameters provided by the DSPJRN command. This allows you to specify which entries go to the exit program.

You can use the RTVJRNE command in a CL program to retrieve a journal entry and place it in program variables. See the topic "Retrieving a Journal Entry" on page 2-27 for more information.

You can use the CMPJRNIMG command to compare and list the difference between the before- and after-image of a record or between the current after-image of a record and the previous after-image of the record. See the topic "Comparing Journal Images" on page 2-28 for more information.

The format of the converted journal entry depends on the value specified for the OUTFILFMT parameter if the DSPJRN command is used is used to write journal entries to a database output file, or the ENTFMT parameter if the RCVJRNE or RTVJRNE command is used. Each converted journal entry has a fixed-length prefix portion followed by a variable length portion containing entry-specific data and associated information. The fields that make up the fixed-length prefix of every converted journal entry are common to all entries and are described in the following pages.

The entry-specific data is either data for system-created entries or user-created data recorded in the journal by the SNDJRNE command. If the entry is system-created, the entry-specific data varies with the type of the journal entry. If the entry is a user-created entry, the entry-specific data is the data specified on the ENTDTA parameter of the SNDJRNE command. System-created entries are described in the following pages.

Contents of a Journal Entry

The entries created when a change is made to a member of a journaled file contain:

- Information identifying the type of change.
- Information identifying the record that was changed.
- The after-image of the record.
- Optionally, the before-image of the record (this is a separate entry in the journal).
- Information identifying the job, the user, the time of change, and so on.
- Information that identifies whether the file was opened, closed, reorganized, cleared, or saved.

The system also places entries in the journal that are not for a particular file member. These entries contain information about the operation of the system and the control of the journal receivers.

Each journal entry is sequentially numbered without any missing numbers until the Change Journal (CHGJRN) command resets the sequence number. However, when journal entries are converted and shown to the user, sequence numbers may be missing. The system uses some journal entries only internally and combines some entries into one during conversion.

When the system exceeds the largest sequence number (2 147 483 647), a message is sent to the system operator identifying the condition and requesting action. No other journal entries can be added to the journal until the journal receivers are changed and the sequence number is reset.

Fixed-Length Portion of a Journal Entry:

Table 2-2 on page 2-14 shows the information in the fixed-length prefix portion of a converted journal entry that is written to a database output file if OUTFILFMT(*TYPE1) is requested on the DSPJRN command, or ENTFMT(*TYPE1) is requested on the RCVJRNE or RTVJRNE command. The field names shown in parentheses in the table are the names of the fields in the system-supplied output file QSYS/QADSPJRN.

*Table 2-2 (Page 1 of 2). Field Descriptions of the Fixed-Length Portion of a Journal Entry *TYPE1*

Field	Format	Description
Entry length (*JOENTL*)	Zoned decimal (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry.
Sequence number (*JOSEQN*)	Zoned decimal (10,0)	Assigned by journal management to each journal entry. It is initially set to 1 for each new or restored journal and is incremented by 1 until you request that it be reset when you attach a new receiver. There are occasional gaps in the sequence numbers because the system uses internal journal entries for control purposes.
Journal code (*JOCODE*)	Character (1)	Identifies the primary category of the journal entry: **A =** Job accounting entry **C =** Commitment control information **F =** File level information **J =** Information about journal receivers and operation of the journal **L =** Information about license management changes to usage limits and usage limit violations **P =** Performance entry **R =** Information about a change to a specific record **S =** SNA distribution services (SNADS) entry **T =** Security-related events **U =** User-created entry (added to the journal by the SNDJRNE command)
Entry type (*JOENTT*)	Character (2)	Further identifies the type of user-created or system-created entry. See the following tables for more information.
Date stamp (*JODATE*)	Character (6)	Specifies the system date when the entry was added and is in the format of the job attribute DATFMT. The system cannot assure that the date stamp is always in ascending order for sequential journal entries because you can change the value of the system date.
Time stamp (*JOTIME*)	Zoned decimal (6,0)	Corresponds to the system time (in the format hhmmss) when the entry was added. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.
Job name (*JOJOB*)	Character (10)	Specifies the name of the job that added the entry.
User name (*JOUSER*)	Character (10)	Specifies the user profile name of the user that started the job.
Job number (*JONBR*)	Zoned decimal (6,0)	Specifies the job number of the user that started the job.

*Table 2-2 (Page 2 of 2). Field Descriptions of the Fixed-Length Portion of a Journal Entry *TYPE1*

Field	Format	Description
Program name (*JOPGM*)	Character (10)	Specifies the name of the program that added the entry. If an application or CL program did not add the entry, the field contains the name of a system-supplied program such as QCMD or QPGMMENU. If the program name is the special value *UNKNOWN, then one of the following is true: • The program name does not apply to this journal entry. • The program name was not available when the journal entry was made. For example, the program name is not available if the program was destroyed.
Object name (*JOOBJ*)	Character (10)	Specifies the name of the object for which the journal entry was added. This is blank for some entries[1].
Library name (*JOLIB*)	Character (10)	Specifies the name of the library containing the object[1].
Member name (*JOMBR*)	Character (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
Count/relative record number (*JOCTRR*)	Zoned decimal (10,0)	Contains a value specified on a parameter of the command for which the journal entry was added or the relative record number of the record in the physical file member (see Table 2-5 on page 2-19 for more information).
Indicator flag (*JOFLAG*)	Character (1)	Contains an indicator for the operation (see Table 2-5 on page 2-19 for more information).
Commit cycle identifier (*JOCCID*)	Zoned decimal (10,0)	Contains a number that identifies the commit cycle. A commit cycle is from one commit or rollback operation to another. The commit cycle identifier is found in every journal entry that is associated with a commitment transaction. If the journal entry was not made as part of a commitment transaction, this field is zero.
Reserved field (*JORES*)	Character (8)	Always contains zeros. Contains hexadecimal zeros in the output file.

Note:

1 If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, then the fully qualified name is the most recent name of the file when the newest receiver in the receiver range was the attached receiver and when the file was still being journaled.

If a file name is specified or if library *ALL is specified on the FILE parameter, the current fully qualified name of the file appears in the converted journal entry.

If OUTFILFMT(*TYPE2) is requested on the DSPJRN command, or ENTFMT(*TYPE2) is requested on the RCVJRNE or RTVJRNE command, then the prefix portion of each converted journal entry fixed-length is the same as the format in Table 2-2 on page 2-14, except for the information that follows the commit cycle identifier field. The fields of the prefix that follow the commit cycle identifier are shown in Table 2-3 on page 2-16. The field names shown in parentheses in the table are the names of the fields in the system-supplied output file QSYS/QADSPJR2.

*Table 2-3. Field Descriptions of the Fixed-Length Portion of a Journal Entry *TYPE2*

Field	Format	Description
User profile (*JOUSPF*)	Character (10)	Specifies the name of the user profile under which the job was running when the entry was created.
System name (*JOSYNM*)	Character (8)	Specifies the name of the system on which the outfile was created.
Reserved field (*JORES*)	Character (20)	Always contains zeros. Contains hexadecimal zeros in the output file.

A third value, *TYPE3 is supported on the OUTFILFMT parameter for the DSPJRN command, and the ENTFMT parameter for the RCVJRNE and RTVJRNE commands. If OUTFILFMT(*TYPE3) is specified on the DSPJRN command, or ENTFMT(*TYPE3) is specified on the RCVJRNE or RTVJRNE command, the information in the prefix portion of a converted journal entry is shown in Table 2-4. The field names shown in parentheses in the table are the names of the fields in the system-supplied output file QSYS/QADSPJR3.

*Table 2-4 (Page 1 of 3). Field Descriptions of the Fixed-Length Portion of a Journal Entry *TYPE3*

Field	Format	Description
Entry length (*JOENTL*)	Zoned decimal (5,0)	Specifies the length of the journal entry including the entry length field, all subsequent positions of the journal entry, and any portion of the journal entry that was truncated if the length of the output record is less than the length of the record created for the journal entry. This field only appears when the journal entries are added to the database output file.
Sequence number (*JOSEQN*)	Zoned decimal (10,0)	Assigned by journal management to each journal entry. It is initially set to 1 for each new or restored journal and is incremented by 1 until you request that it be reset when you attach a new receiver. There are occasional gaps in the sequence numbers because the system uses internal journal entries for control purposes.
Journal code (*JOCODE*)	Character (1)	Identifies the primary category of the journal entry: **A =** Job accounting entry **C =** Commitment control information **F =** File level information **J =** Information about journal receivers and operation of the journal **L =** Information about license management changes to usage limits and usage limit violations **P =** Performance entry **R =** Information about a change to a specific record **S =** SNA distribution services (SNADS) entry **T =** Security-related events **U =** User-created entry (added to the journal by the SNDJRNE command)
Entry type (*JOENTT*)	Character (2)	Further identifies the type of user-created or system-created entry. See the following tables for more information.

*Table 2-4 (Page 2 of 3). Field Descriptions of the Fixed-Length Portion of a Journal Entry *TYPE3*

Field	Format	Description
Time stamp (*JOTMST*)	Character (26)	Corresponds to the system date and time when the journal entry was added in the journal receiver. The time stamp is in SAA format. The system cannot assure that the time stamp is always in ascending order for sequential journal entries because you can change the value of the system time.
Job name (*JOJOB*)	Character (10)	Specifies the name of the job that added the entry.
User name (*JOUSER*)	Character (10)	Specifies the user profile name of the user that started the job.
Job number (*JONBR*)	Zoned decimal (6,0)	Specifies the job number of the user that started the job.
User profile (*JOUSPF*)	Character (10)	Specifies the name of the user profile under which the job was running when the entry was added.
Program name (*JOPGM*)	Character (10)	Specifies the name of the program that added the entry. If an application or CL program did not add the entry, the field contains the name of a system-supplied program such as QCMD or QPGMMENU. If the program name is the special value *NONE, then one of the following is true: • The program name does not apply to this journal entry. • The program name was not available when the journal entry was made. For example, the program name is not available if the program was destroyed.
Object name (*JOOBJ*)	Character (10)	Specifies the name of the object for which the journal entry was created. This is blank for some entries[1].
Library name (*JOLIB*)	Character (10)	Specifies the name of the library containing the object[1].
Member name (*JOMBR*)	Character (10)	Specifies the name of the physical file member or is blank if the object is not a physical file[1].
Count/relative record number (*JOCTRR*)	Zoned decimal (10,0)	Contains a value specified on a parameter of the command for which the journal entry was created or the relative record number of the record in the physical file member (see Table 2-5 on page 2-19 for more information).
Indicator flag (*JOFLAG*)	Character (1)	Contains an indicator for the operation (see Table 2-5 on page 2-19 for more information).
Commit cycle identifier (*JOCCID*)	Zoned decimal (10,0)	Contains a number that identifies the commit cycle. A commit cycle is from one commit or rollback operation to another. The same commit cycle identifier is found in every journal entry that is associated with a particular commitment transaction. If the entry was not made as part of a commitment transaction, this field is zero.
System name (*JOSYNM*)	Character (8)	Specifies the name of the system on which the outfile was created.

*Table 2-4 (Page 3 of 3). Field Descriptions of the Fixed-Length Portion of a Journal Entry *TYPE3*

Field	Format	Description
Reserved field (*JORES*)	Character (20)	Always contains zeros. Contains hexadecimal zeros in the output file. This field is shown in the output file only when the DSPJRN command is used. It is not shown when the RCVJRNE or the RTVJRNE command is used.

Note:

1 If *ALLFILE is specified for the FILE parameter on the DSPJRN, RCVJRNE, or RTVJRNE command, then the fully qualified name is the most recent name of the file when the newest receiver in the receiver range was the attached receiver and when the file was still being journaled.

If a file name is specified or if library *ALL is specified on the FILE parameter, the current fully qualified name of the file appears in the converted journal entry.

System-Created Journal Entries

This section describes the system-created journal entries by journal code.

Journal Code A: Journal entries with a journal code of A contain information about job accounting. Refer to the manual *Programming: Work Management Guide*, SC41-8078, for a detailed description of the contents of converted journal entries with journal code A.

Journal Code C: Journal entries with a journal code of C contain commitment control information. Additional information about journal entries with journal code C follows in "Entry Types by Journal Code" on page 2-19.

Journal Code F: Journal entries with a journal code of F contain file level information about changes for a physical file member that are being journaled to this journal. (If you use a logical file in a program, the file level information reflects the physical file on which the logical file is based.) Journal entries with journal code F can also contain file level information for access paths associated with physical or logical file members that are being journaled to this journal. Additional information about journal entries with journal code F follows in "Entry Types by Journal Code" on page 2-19.

Journal Code J: Journal entries with a journal code of J contain information about the journal and the journal receivers. Additional information about journal entries with journal code J follows in "Entry Types by Journal Code" on page 2-19.

Journal Code L: Journal entries with a journal code of L contain information about license management changes to the usage limit and usage limit violations. Additional information about journal entries with journal code L follows in "Entry Types by Journal Code" on page 2-19.

Journal Code P: Journal entries with a journal code of P contain information about performance. For the description of the layout of these entries, refer to the manual *Programming: Work Management Guide*, SC41-8078.

Journal Code R: Journal entries with a journal code of R contain information about a change to a specific record in the physical file member that is being journaled to the journal. For a given physical file member, the record-level journal entries appear in the journal in the order that the changes were made to the file. Additional information about journal entries with journal code R follows in "Entry Types by Journal Code" on page 2-19.

Journal Code S: Journal entries with a journal code of S contain information about SNA distribution services (SNADS). For the description of the layout of these entries, refer to the manual *Communications: Distribution Services Network Guide*, SC41-9588. Journal code S also has journal entries for X.400. For the description of X.400 journal entries, refer to the manual *OSI Message Services/400 Guide*, SC41-0026.

Journal Code T: Journal entries with a journal code of T contain auditing information. For the description of the layout of audit journal entries, see the manual *Security Reference*, SC41-8083.

Entry Types by Journal Code: Table 2-5 describes, by journal code, the contents of the *Entry Type* field in the journal entries.

Table 2-5 (Page 1 of 4). Journal Entries by Journal Code and Type

Journal Code	Entry Type	Operation Description	See Note
A	DP	Direct print information.	1
A	JB	Job resource information.	1
A	SP	Spooled print information.	1
C	BC	Start commitment control (STRCMTCTL).	
C	CM	Set of record changes committed (COMMIT).	2
C	EC	End commitment control (ENDCMTCTL).	
C	RB	Set of record changes rolled back (ROLLBACK).	3
C	SC	Commit transaction started.	
F	AY	Journaled changes applied to a physical file member (APYJRNCHG).	4
F	CE	Change end of data for physical file member.	5
F	CL	Physical file member closed (for shared files, a close entry is made for the last close operation of the file).	6
F	CR	Physical file member cleared (CLRPFM).	
F	EJ	Journaling for a physical file member ended (ENDJRNPF).	
F	EP	Journaling access path for a database file member ended (ENDJRNAP).	
F	FD	Physical file member forced (written) to auxiliary storage.	7
F	IU	Physical file member in use at the time of abnormal system end.	8,9
F	IZ	Physical file member initialized (INZPFM).	10
F	JM	Journaling for a physical file member started (STRJRNPF).	11
F	JP	Journaling access path for a database file member started (STRJRNAP).	
F	MD	Physical file member deleted. This entry is created when you remove the member (RMVM) or delete the file (DLTF) containing the member.	
F	MF	Physical file member saved with storage freed (SAVOBJ, SAVCHGOBJ, or SAVLIB).	
F	MM	Physical file containing the member moved to a different library (MOVOBJ or RNMOBJ OBJTYPE(*LIB)).	12
F	MN	Physical file containing the member renamed (RNMM or RNMOBJ).	12
F	MR	Physical file member restored (RSTOBJ or RSTLIB).	13
F	MS	Physical file member saved (SAVOBJ, SAVLIB, or SAVCHGOBJ).	21
F	OP	Physical file member opened (for shared files, an open entry is added for the first open of the file).	6
F	PD	Database file member's access path deleted (this entry is created when you remove the member (RMVM) or delete the file (DLTF) containing the member).	
F	PM	The logical owner of a journaled access path was moved (MOVOBJ or RNMOBJ OBJTYPE(*LIB)).	12
F	PN	The logical owner of a journaled access path was renamed (RNMOBJ or RNMM).	12

Table 2-5 (Page 2 of 4). Journal Entries by Journal Code and Type

Journal Code	Entry Type	Operation Description	See Note
F	RC	Journaled changes removed from a physical file member (RMVJRNCHG).	4
F	RG	Physical file member reorganized (RGZPFM).	14
F	SA	The point at which the APYJRNCHG command started running.	
F	SR	The point at which the RMVJRNCHG command started running.	
F	SS	The start of the save of a physical file member using the save-while-active function.	22
J	IA	System IPL after abnormal end.	8
J	IN	System IPL after normal end.	8
J	NR	Identifier for the next journal receiver (the receiver that was attached when the indicated receiver was detached).	15
J	PR	Identifier for the previous journal receiver (the receiver that was detached when the indicated receiver was attached).	15
J	RD	Delete of a journal receiver (DLTJRNRCV).	
J	RF	Storage for a journal receiver freed (SAVOBJ, SAVCHGOBJ, or SAVLIB).	
J	RR	Restore of a journal receiver (RSTOBJ or RSTLIB).	13
J	RS	Save of a journal receiver (SAVOBJ, SAVCHGOBJ, or SAVLIB).	13
L	LL	Usage limit changed.	23
L	LU	Usage limit exceeded.	24
P	TP	Performance shared pool change.	1
R	BR	Before-image of record updated for rollback operation.	16,17
R	DL	Record deleted in the physical file member.	16,17
R	DR	Record deleted for rollback operation.	16,17
R	PT	Record added to a physical file member.	17,25
R	PX	Record added directly by RRN (relative record number) to a physical file member.	17,25
R	UB	Before-image of a record that is updated in the physical file member (this entry is present only if IMAGES(*BOTH) is specified on the STRJRNPF command).	16,17
R	UP	After-image of a record that is updated in the physical file member.	17
R	UR	After-image of a record that is updated for rollback information.	17
S	AL	SNA alert focal point information	
S	CF	A change was made to the next system table through the CFGDSTSRV command.	18
S	DX	X.400 process debug entry	
S	ER	An error was detected by a SNADS process (this includes, but is not limited to, the sender, receiver, and router).	18
S	LG	An operation, such as sending or receiving a distribution queue entry, was successfully performed.	18
S	MX	A change was made to X.400 MTA configuration.	
S	NX	A change was made to X.400 delivery notification.	

Table 2-5 (Page 3 of 4). Journal Entries by Journal Code and Type

Journal Code	Entry Type	Operation Description	See Note
S	RT	A change was made to the routing table through the CFGDSTSRV command.	18
S	RX	A change was made to X.400 route configuration.	
S	SY	A change was made to the SNA Distributed Services system information.	
S	UX	A change was made to X.400 user or probe.	
S	XE	DSNX error entry.	18
S	XL	DSNX logging entry.	18
S	XX	An error was detected by the X.400 process.	
T	AD	A change was made to the auditing attribute.	
T	AF	All authority failures.	19
T	AP	A change was made to program adopt.	
T	CA	Changes to object authority (authorization list or object).	19
T	CD	A change was made to a command string.	
T	CO	Create object.	19
T	CP	Create, change, delete, display, restore of user profiles.	19
T	DO	All delete operations on the system.	19
T	DS	DST security officer password reset.	19
T	JD	Changes to the USER parameter of a job description.	19
T	JS	A change was made to job data.	
T	ML	A change was made to office services mail.	
T	NA	Changes to network attributes.	19
T	OM	Object management change.	19
T	OR	Object restored.	19
T	OW	Changes to object ownership.	19
T	PA	Changes to programs (CHGPGM) that will now adopt the owner's authority.	19
T	PO	A change was made to printed output.	
T	PS	Profile swap.	19
T	PW	Passwords used that are not valid.	19
T	RA	Restore of objects when authority changes.	19
T	RJ	Restore of job descriptions that contain user profile names.	19
T	RO	Restore of objects when ownership information changes.	19
T	RP	Restore of programs that adopt their owner's authority.	19
T	RU	Restore of authority for user profiles.	19
T	SD	A change was made to the system directory.	
T	SE	Changes to subsystem routing.	19
T	SF	A change was made to a spooled output file.	
T	SM	A change was made by system management.	
T	ST	A change was made by system tools.	

Table 2-5 (Page 4 of 4). Journal Entries by Journal Code and Type

Journal Code	Entry Type	Operation Description	See Note
T	SV	Changes to system values.	19
T	YC	A change was made to DLO change access.	
T	YR	A change was made to DLO read access.	
T	ZC	A change was made to object change access.	
T	ZR	A change was made to Object read access.	
U	User-created	User-specified.	20

Notes

1 Refer to the *Programming: Work Management Guide*, SC41-8078, for the layout of the converted journal entry.

2 The *Count* field contains the length of the commit identification, and the *Entry-specific data* field contains the commit ID specified on the operation (character data). The *Flag* indicates whether the commit operation was started by the user or by the operating system.

Hex F0 = All record-level changes were committed for a commit operation initiated by a user.

Hex F2 = All record-level changes were committed for a commit operation initiated by the operating system.

3 The *Job name* and *Program name* fields do not appear if the entry was added during an IPL. The *Flag* field indicates whether the rollback operation was successful for record-level changes and whether it was initiated by a user or the operating system.

Hex F0 = All record-level changes were rolled back for a rollback operation initiated by a user.

Hex F1 = Not all record-level changes were successfully rolled back for a rollback operation initiated by a user.

Hex F2 = All record-level changes were rolled back for a rollback operation initiated by the operating system.

Hex F3 = Not all record-level changes were rolled back for a rollback operation initiated by the operating system.

4 The *Count* field contains the number of journal entries applied or removed. The *Flag* field indicates the completion status:

Hex F0 = command completed normally
Hex F1 = command completed abnormally

The *Entry-specific data* contains the following:

Sequence number of the first entry applied or removed, zoned decimal (10, 0)
Sequence number of the last entry applied or removed, zoned decimal (10, 0)

```
Starting receiver name   Character (10)
Library name             Character (10)
Ending receiver name     Character (10)
Library name             Character (10)
```

5 The *Count* field contains the relative record number of the last record kept in the physical file member.

6 The *Entry-specific data* field contains the following information on the file opened or closed. (This is a different name than the *Object Name* if a logical file was opened.)

```
File Name      Character (10)
Library name   Character (10)
Member name    Character (10)
Open options   Character (4) (not applicable to control language (CL)
                              entry type)
                        Byte   Contents
                        1      I = File opened for input
                               Blank = Input not specified
                        2      O = File opened for output
                                        Blank = Output not specified
                        3      U = File opened for update
                               Blank = Update not specified
                        4      D = File opened for delete
                               Blank = Delete not specified
```

7 If the FD entry occurs at IPL, the *Job name* and *Program name* fields are blank. The job number is zero.

8 The time stamp created at IPL is read from the battery-powered clock. If the battery-powered clock cannot be read, the time is that of the system power down, not the time of the IPL because the system time has not been updated yet at the time of the journal entry being logged.

9 The *Flag* field indicates whether the file was synchronized with the journal:

Hex F0 = file was synchronized with journal
Hex F1 = file was not synchronized with journal

10 The *Flag* field indicates the type of record:

Hex F0 = *DFT (default)
Hex F1 = *DLT (delete)

The *Count* field contains the number of records specified on the TOTRCDS parameter of the Initialize Physical File member (INZPFM) command.

If the member is initialized with default records, the *Entry-specific data* in the entry contains the default record image.

11 The *Flag* field indicates the type of images:

Hex F0 = after-images
Hex F1 = before- and after-images

The *Entry-specific data* field contains an indicator of the value specified on the OMTJRNE parameter of the STRJRNPF command:

Hex F0 = no entries are omitted from being journaled
Hex F1 = open and close entries are omitted

12 The *Entry-specific data* field contains the name of the member before a move or a rename operation in the following form:

```
File Name      Character (10)
Library name   Character (10)
Member name    Character (10)
```

13 The *Entry-specific data* field contains the following restore information:

```
Media type (DKT, TAP or SAV)          Character (3)
First Volume ID                       Character (6)
Date of restore operation             Character (6)
Time of restore operation             Zoned decimal (6, 0)
```

14 The *Entry-specific data* field contains the name of the file used in the Reorganize Physical File Member (RGZPFM) command. This field is blank if you specified KEYFILE(*NONE). If you specified other than KEYFILE(*NONE), for example, if you specified KEYFILE(library-name/filename member-name), this field contains the name of the logical file.

```
File Name      Character (10)
Library name   Character (10)
Member name    Character (10)
```

15 The *Count* field contains the number of receivers attached or detached. The *Entry-specific data* field contains the receiver name (10 characters) and the library name (10 characters) for each receiver attached or detached.

16 The *Flag* field indicates whether a record image is present in the journal entry:

Hex F0 = before-image is not present
Hex F1 = before-image is present

If before-images are being journaled, and the *Flag* field indicates that a before-image is not present, this implies that an update or a delete operation is being requested for a record that has already been deleted.

17 The *Entry-specific data* field contains the image of the physical record. For entry types PT, PX, UP or UR, this field contains the after-image of the record. For entry types UB, DL, BR or DR, this field contains the before-image of the record if IMAGE(*BOTH) was specified and if the record was not previously deleted.

18 Refer to the *Communications and Systems Management Guide (Alerts and Distributed Systems Node Executive)*, SC41-9661, for the layout of the converted journal entry.

19 Refer to the *Security Reference*, SC41-8083, for the layout of the converted journal entry.

20 The *Entry-specific data* is the value specified on the ENTDTA parameter of the SNDJRNE command.

21 The *Entry-specific data* field contains the following save information.

```
Media type (DKT, TAP or SAV)            Character (3)
First Volume ID                         Character (6)
Date of start of save operation         Character (6)
Time of start of save operation         Zoned decimal (6,0)
Update history flag                     Character (1)
   The values in this field are:
      HEX F1 = UPDHST(*YES) specified on the save request
      HEX F0 = UPDHST(*NO) specified on the save request
Save file name                          Character (10)
    Blanks if not DEV(*SAVF)
Save file library name                  Character (10)
    Blanks if not DEV(*SAVF)
SAVACT parameter value                  Character (10)
   This is the value specified for the
   SAVACT parameter on the SAVOBJ, SAVCHGOBJ,
   or SAVLIB command.
Date of save active operation           Character (6)
   If this is a save-while-active operation,
   this is the date when checkpoint processing
   was completed for this database member.

   If this is a normal save operation, this
   is the same date as the start of the save
   operation.

Time of save active operation           Zoned decimal (6,0)
   If this is a save-while-active operation,
   this is the time when checkpoint processing
    was completed for this database member.

   If this is a normal save operation, this
   is the same time as the start of the save
   operation.

Primary receiver name                   Character (10)
   The name of the first of dual
   receivers that contains the 'F SS' entry.

Primary receiver library                Character (10)
   The name of the library containing
   the primary receiver.

Dual receiver name                      Character (10)
   The name of the second of dual
   receivers that contains the 'F SS' entry.
   This entry will be blank if only a single
   receiver was used when the 'F SS' entry was
   added.

Dual receiver library
   The name of the library containing
   the primary receiver.
   This entry will be blank if only a single
   receiver was used when the 'F SS' entry was
   added.
```

If a physical file member was saved using the save-while-active function, the saved copy of the member includes all the record-level changes found in the journal entries up to the corresponding 'F SS' save entry.

If the physical file member was not saved using the save-while-active function, the saved copy of the member includes all record-level changes found in the journal entries up to the corresponding 'F MS' member saved entry.

22 The *Entry-specific data* field for the 'F SS' contains the following save information.

```
Media type (DKT, TAP or SAV)          Character (3)
First Volume ID                       Character (6)
Date of start of save operation       Character (6)
Time of start of save operation       Zoned decimal (6,0)
Update history flag                   Character (1)
   The values in this field are:
      HEX F1 = UPDHST(*YES) specified on the save request
      HEX F0 = UPDHST(*NO) specified on the save request
Save file name                        Character (10)
    Blanks if not DEV(*SAVF)
Save file library name                Character (10)
    Blanks if not DEV(*SAVF)
SAVACT parameter value                Character (10)
   This is the value specified for the
   SAVACT parameter on the SAVOBJ, SAVCHGOBJ,
   or SAVLIB command.
Date of save active operation         Character (6)
   If this is a save-while-active operation,
   this is the date when checkpoint processing
   was completed for this database member.

   If this is a normal save operation, this
   is the same date as the start of the save
   operation.
```

If a physical file member was saved using the save-while-active function, the saved copy of the member includes all the record-level changes found in the journal entries up to the corresponding the 'F SS' save entry.

If the physical file member was not saved using the save-while-active function, the saved copy of the member includes all record-level changes found in the journal entries up to the corresponding 'F MS' member saved entry.

23 The *Entry-specific data* field for the 'L LL' entry contains the following information.

```
Product ID             Character(7)
Feature                Character(4)
License term           Character(6)
Previous usage limit   Zoned decimal(6,0)
Current usage limit    Zoned decimal(6,0)
```

24 The *Entry-specific data* field for the 'L LU' contains the following information.

```
Product ID             Character(7)
Feature                Character(4)
License term           Character(6)
Usage limit            Zoned decimal(6,0)
Request flag           Character(1)
      Hex F0 = license request was successful
      Hex F1 = license request was not successful
# of license users     Zoned decimal(4,0)
License user           Character(26) (for each license user up to 1000)
```

25 If the file is set up to reuse deleted records, then you may receive either a PT or PX journal entry for the change.

Variable-Length Portion of a Journal Entry

If you use DSPJRN with *TYPE1 or *TYPE2 specified for the OUTFILFMT parameter, or if you use RCVJRNE or RTVJRNE with *TYPE1 or *TYPE2 specified for the ENTFMT parameter, the variable length portion of the journal entry includes just the entry-specific data. The contents of the entry-specific data depends on the journal entry code and entry type as was explained in the notes to Table 2-5 on page 2-19.

If OUTFILFMT(*TYPE3) is requested on a DSPJRN command, the variable length portion of the converted journal entry will have two fields.

The first field is the *Null Value Indicators* and the second field is the *Entry-specific data*. The *Null Value Indicators* contains relevant information only for entries with journal code R. This field is meaningful only if the journaled file has at least one null-capable field. Otherwise, it contains blanks. The *Null Value Indicators* is a character string with one character for each field in the physical file that has record images appearing in the journal. Each character has the following interpretation:

Hex F0 = corresponding field in the record is not NULL.

Hex F1 = corresponding field in the record is NULL.

The second field in the variable-length portion of the journal entry is the *Entry-specific data* in the journal entry.

The *Null Value Indicators* and *Entry-Specific Data* fields have been defined as variable-length character fields in the system-supplied output file QSYS/QADSPJR3. See the DSPJRN command in the *Programming: Control Language Reference*, SC41-0030, for additional details regarding OUTFILFMT(*TYPE3).

If ENTFMT(*TYPE3) is requested on the RCVJRNE or a RTVJRNE command, the variable-length portion of the converted journal entry will include null value indicators and the entry-specific data. The RCVJRNE and RTVJRNE commands in the *Programming: Control Language Reference*, SC41-0030, have the details on the exact layout of these two fields.

User-Created Journal Entries

Use the Send Journal Entry (SNDJRNE) command to add your own entries to a journal. The system places these entries in the journal's attached journal receiver along with the system-created journal entries. To help identify your entries, you can associate each entry with a particular physical file member.

You may add entries to the journal to identify significant events (such as a checkpoint) or to help in the recovery of your applications. The data specified on the ENTDTA parameter becomes the *Entry-Specific Data* field in the journal entry, and the TYPE parameter value becomes the *entry type*.

Retrieving a Journal Entry

Use the Retrieve Journal Entry (RTVJRNE) command in a CL program to retrieve a journal entry and place it in variables in the program. You can retrieve the following:

- Sequence number
- Journal code
- Entry type
- Journal receiver name
- Library name for the journal receiver
- Journal entry

For example, you can use this command to automate your recovery procedures or to change the journal receivers and then save them. The manual *Programming: Control Language Reference*, SC41-0030, has a description of the format of the journal entry. An appendix about tips and techniques in the *Basic Backup and Recovery Guide* has an example of using the RTVJRNE command in a program.

Using Journaling to Provide an Audit Trail

You can use journal management to provide an audit trail of changes made to your database files. You can determine which program or user made changes to files by using the journal entries, and you can determine what changes were made to specific records by using the CMPJRNIMG command.

Journal management can be used to provide an audit trail because:

- The journal entries cannot be removed or changed, even by the security officer.
- The journal entries represent a chronological sequence of events.
- Each journal entry in the system is sequentially numbered without gaps until the sequence number is reset by the CHGJRN command. If the sequence number is reset, a journal entry is written. When you display the journal entries, there can be gaps in the sequence numbers because some journal entries are only used internally by the system and some entries are combined when they are displayed.
- The journal contains entries indicating when each journal receiver was changed and the name of the next journal receiver in the chain.
- Entries are written whenever journaling for a file is ended or whenever a file is restored.

Remember that the date and time recorded in the journal entries depends on the date and time entered during an IPL and therefore, may not represent the actual date and time. Also, if you use shared files, the program name that appears in the journal entry is the name of the program that first opened the shared file.

Comparing Journal Images

Use the Compare Journal Images (CMPJRNIMG) command to compare and list the differences between the before-image of a record and the after-image of that record, or the after-image of a record with the previous after-image of that record.

If the journaled files have null-capable fields, the null value indicators corresponding to the fields in the before-image of the record are compared with the null value indicators corresponding to the fields in the after-image of the record. This is done on a field-by-field basis.

The printed output from the CMPJRNIMG command shows the before- and after-images of a record followed by a line that indicates (with asterisks) the specific change in the record on a character-by-character basis. If you compare the after-images, the output shows the previous after-image of the record and the current after-image of the record, followed by a line indicating the changes.

Access Path Recovery

This section describes how to recover access paths more quickly after the system ends abnormally using access path journaling. If you use access path journaling, you do not need to completely rebuild access paths after an abnormal system end.

Unless you choose to journal your access paths, the system can spend a significant amount of time rebuilding access paths during the IPL following an abnormal system end.

Journaling Access Paths

You can use access path journaling to recover access paths during an IPL.

Files created with MAINT(*IMMED or*DLY) can have journaling started for their access paths.

Access paths for files that are specified with a parameter value of MAINT(*REBLD) cannot be journaled. Recovery for rebuilt maintenance files, created with the MAINT(*REBLD) parameter value, takes place during the next open operation that uses that access path.

Before you begin journaling access paths, consider the following:

- Before you can journal an access path, you must first journal the physical files over which that access path is built.
- All physical files over which the access path is built must be journaled to the same journal as the access path.
- Any physical file specified for access path journaling must have a keyed access path. Any logical files that use the physical file as a dependent file are not implicitly journaled. You must specify each access path that is to be journaled by naming a specific file.
- All physical files must remain journaled while the access path is being journaled.

Access Path Journaling Storage and Performance Considerations: Space requirements can increase significantly if you journal access paths. When anticipating the effect of access path journaling on system space and performance, consider the following:

- Because access paths can be shared, you can journal the same access path from several files. For example, file B shares the access path of file A. Both files can request journaling of their access paths and there is no additional overhead. If file A ends journaling its access path, the access path remains journaled until file B ends journaling the access path. Journaling of access paths must be ended through the same file through which access path journaling was started.
- Access path journaling does not cause additional synchronous write operations to the disk beyond those required to journal the underlying physical file. Access path information is written at the same time as the after-image of the underlying changed database record. The effect on system performance due to access path journaling is kept to a minimum.
- Access path journaling can significantly increase the amount of auxiliary storage used. For more information, see "Journaling Performance and Space Considerations" on page 2-3.
- To improve the performance of journaling, you should isolate the journal receiver to a user ASP. The amount of additional auxiliary storage required varies, based on the access path characteristics and the types of changes that occur. For more information on user ASPs, refer to "General Information about Auxiliary Storage Pools" on page 6-8.

  The amount of additional storage needed depends on how many access paths are being journaled, and how the journaled access paths are updated by various applications. Several journal entries that cannot be displayed are also added to the journal receiver when journaling access paths. Applications that update, delete, or add keys in an ascending or descending order use less auxiliary storage than applications that update, delete, or add keys in random order. The minimum requirement occurs when you make all changes to the same primary value of the access path (for example, when the access path uses the date-added field as its primary value and you add many records on the same day).

  Note: If you do not put journal receivers in a separate user ASP, consider saving them regularly on tape or diskette to protect them from loss due to disk failure.
- The system journals information in addition to the before-image and after-image of the database records. The system uses this additional information only after an abnormal system end to recover the access paths.

Space requirements also increase when the number of files and/or the number of file access paths being journaled increases.

Considerations for Access Path Journaling: Consider the following when using access path journaling for database files:

- Access paths for both physical and logical files can be journaled. Designate the physical and logical files that will have their access paths journaled using the Start Journal Access Path (STRJRNAP) command. When you specify journaling for database file access paths, all access paths for all members are journaled and new members added to the file automatically have their access paths journaled.
- If a database file is saved while access paths are journaled, and the file is deleted and restored, the restore operation attempts to begin journaling the access paths for the file, if the journal was restored before the file.
- All journaled access paths must have a maintenance attribute of *IMMED or *DLY specified on the MAINT parameter on the Create Physical File (CRTPF), Create Source Physical File (CRTSRCPF), and Create Logical File (CRTLF) commands as well as the Change Physical File (CHGPF), Change Logical File (CHGLF), or Change Source Physical File (CHGSRCPF) command. MAINT(*REBLD) cannot be specified.
- All access paths that are to be journaled must have a force attribute of *NO (the FRCACCPTH parameter on the CRTPF,

CRTLF, CRTSRCPF, CHGPF, CHGLF, or CHGSRCPF command).

Access Path Journal Entries

Several entry types for journal code F are used to describe journal activity for journaled access paths. These entries are:

JP Start journaling an access path. For join logical files, more than one JP entry can be created for the same logical file member: one for the primary access path and one for each of the secondary access paths. If a file member's access path is already being journaled when access path journaling is started for the file, no JP entry is created for that access path. This condition can arise if the access path is being shared by several file members, and one of the sharing files has already journaled the access path.

EP End journaling access path for member. For join logical files, more than one EP entry can be created for the same logical file member: one for the primary access path and one for each of the secondary access paths. If a file member's access path is being shared and one of the other sharing files is still journaling the access path, journaling is not ended for that access path and no EP entry is written.

PD Access path deleted. A journaled access path was deleted by deleting a file journaling its access paths or by removing a member from a file journaling its access paths.

PM The logical owning member of a journaled access path was moved to a different library.

PN The logical owning member of a journaled access path was renamed.

Other system-created entries are recorded in the journal for journaled access paths. You cannot display or use them in an apply or remove journal changes operation.

For a complete description of the system-created journal entries for a file member, see "System-Created Journal Entries" on page 2-18.

Access Path Recovery Actions

No explicit recovery actions initiated by the user are required when journaling access paths. After an abnormal system end, the system automatically uses the journal to recover journaled access paths, if necessary.

Journal Management Commands

The following commands are used for journal management. These commands are used to create objects required for journal management functions. For a complete description of these commands, see the *CL Reference*.

Journal: Use these commands for journals:

Command	Meaning	Description
CHGJRN	Change Journal	Changes the attributes of a journal and attaches new journal receivers to the journal.
CRTJRN	Create Journal	Creates a journal.
DLTJRN	Delete Journal	Deletes a journal.
WRKJRNA	Work with Journal Attributes	Displays the attributes of a journal.

Journal Receiver: Use these commands for journal receivers:

Command	Meaning	Description
CRTJRNRCV	Create Journal Receiver	Creates a journal receiver.
DLTJRNRCV	Delete Journal Receiver	Deletes a journal receiver.
DSPJRNRCVA	Display Journal Receiver Attributes	Displays the attributes of a journal receiver.

Journal Entries: Use these commands for journal entries:

Command	Meaning	Description
CMPJRNIMG	Compare Journal Images	Compares the before-images and the after-images of record-level changes in a file member and indicates where differences occur.

Command	Meaning	Description
DSPJRN	Display Journal	Displays the journal entries that are in the journal receivers associated with the specified journal.
RTVJRNE	Retrieve Journal Entry	Retrieves a journal entry and places it in CL program variables.
RCVJRNE	Receive Journal Entry	Allows a specified user program to continuously receive journal entries one at a time. Can be used to provide backup on another system.
SNDJRNE	Send Journal Entry	Places a user-created entry in the journal.

Files: Use these commands to start and end journaling a file or access path. For more information on journaling access paths, see the topic "Access Path Recovery" on page 2-28.

Command	Meaning	Description
STRJRNPF	Start Journal Physical File	Starts journaling for the physical file.
ENDJRNPF	End Journal Physical File	Ends journaling for the physical file.
STRJRNAP	Start Journal Access Path	Starts journaling access paths.
ENDJRNAP	End Journaling Access Path	Ends journaling access paths.

Database File Member: Use these commands to recover a database file member using the journaled changes:

Command	Meaning	Description
APYJRNCHG	Apply Journaled Changes	Applies the changes to the designated physical file member that were recorded in a journal receiver associated with the journal.
RMVJRNCHG	Remove Journaled Changes	Removes the changes from the designated physical file member that were recorded in a journal receiver associated with the journal (allowed only if before-images and after-images are journaled for the file).

Journal Functions: Use this command to perform journal functions, particularly, recovery functions:

Command	Meaning	Description
WRKJRN	Work with Journals	Displays a Work menu for user-selected journals from which the user can perform system-assisted recovery of journaled files, journals, and journal receivers.

Working with System-Supplied Journals and Journal Receivers

The following journals are shipped with the operating system. With some licensed programs, the OS/400 licensed program automatically adds entries to the journals. For example, OfficeVision/400, QSNADS, and QDSNX automatically add entries to the journals.

The following is a list of journals shipped with the system. These journals and their associated journal receivers are found in library QUSRSYS.

- QDSNX (distributed services network journal)
- QAOSDIAJRN (document library and directory journal)
- QSNADS (Systems Network Architecture distribution services journal)
- QLZALOG (license management)
- QSXJRN (problem database journal)

Two journals can be optionally configured. These journals and their associated journal receivers are created in library QSYS.

- QAUDJRN (security auditing journal)
- QACGJRN (job accounting journal)

For more information about what each journal is used for, see the *Basic Backup and Recovery Guide*.

The journal receivers attached to these journals can become quite large because of the activities performed by certain products. It is recommended that periodically (for example, weekly), large journal receivers should be detached, saved, and then deleted to free storage.

Before a journal receiver can be deleted, you must first detach it from the journal. The Change Journal (CHGJRN) command is used to detach a journal receiver from the journal and attach a new journal receiver.

As a new journal receiver is attached to the journal, the previously attached journal receiver is automatically detached. When the journal receiver becomes too large, you can periodically detach, save, and then delete the journal receiver without affecting normal system operations and without affecting the use of the files. However, it is recommended that this operation not be performed during the time the system is at maximum use.

When you use JRNRCV(*GEN) on the CHGJRN command, the system creates the new receiver with the same values as the currently attached receiver, and in the same library. These values include the owner, private authorities, public authority, ASP identifier, threshold, and text.

When you change from one receiver to another (when JRNRCV(*GEN) is specified), the last number in the new receiver name is one greater than the last number in the detached receiver. For example, if you detach journal receiver QAOSDI0001, the system-created journal receiver is QAOSDI0002.

Example of Working with System-Created Journals: If you know the name of the journal, you can use the Work with Journal Attributes (WRKJRNA) command to determine which receivers are currently attached, which files are being journaled, and to provide a list of all receivers associated with the journal.

To display all journals on the system, do the following:

1. Type the following on a command line and then press the Enter key:

   ```
   WRKJRN
   ```

2. The Specify Journal Name display is shown. *ALL is the default for the journal. Specify *ALL for the *Library* field and press the Enter key.
3. The Work with Journals display is shown. A list of all the journals on the system are shown.

   System-supplied journals and journal receivers start with the letter *Q*.
4. If you want to display the status of a specific system-supplied journal, type a 5 (Display journal status) in the *Opt* column and press the Enter key.
5. Use the following command (where Qxxxxxxxxx is the name of the system-supplied journal) to detach the journal receiver from the journal and create a new journal receiver.

   ```
   CHGJRN JRN(Qxxxxxxxxx) JRNRCV(*GEN)
   ```

6. It is recommended that you save the detached journal receiver. If you want to save the detached journal receiver (where QxxRCV001 is the name of the journal receiver), type the following and press the Enter key.

   ```
   SAVOBJ OBJ(QxxxRCV001) OBJTYPE(*JRNRCV) LIB(QUSRSYS)
          DEV(TAP01)
   ```

7. If you *do not* want to save the journal receiver but do want to delete it, type the following and press the Enter key.

   ```
   DLTJRNRCV JRNRCV(QUSRSYS/QxxxRCV001)
   ```

8. If the journal receiver was not saved, enter I when the following message is shown.

   ```
   Receiver not fully saved. ( C I )
   ```

9. Press the Enter key.

   This completes saving and deleting the detached journal receiver.

Chapter 3. Working with Journal Recovery Operations

Before configuring the journaling environment, you should decide whether to:

- Use one or more journals
- Use one or two journal receivers for each journal
- Journal only after-images or both before-images and after-images of database records
- Put your journal receivers in a user ASP

Isolating your journal receivers in a user ASP other than the ASP that contains the journals and database files provides:

- Better performance of the journal function
- Protection against loss of data on auxiliary storage devices

For additional information on user ASPs, refer to "General Information about Auxiliary Storage Pools" on page 6-8.

Use the following steps to configure your journal:

1. Create the journal receivers using the Create Journal Receiver (CRTJRNRCV) command.
2. Create the journals using the Create Journal (CRTJRN) command.

   Note: All journals and journal receivers on the system should have unique names. For example, try not to have JRNA in library A and library Y. If both libraries become damaged and a reclaim storage (RCLSTG command) operation is required, the duplicate journals will be renamed when they are placed in library QRCL. The system will not allow you to rename those journals or journal receivers placed in library QRCL.
3. Use the Start Journaling Physical File (STRJRNPF) command to start journaling all members of a physical file.

   If you use the recommended method of creating the library in the ASP, then the journals and the files to be journaled must be created in the same ASP. However, if you use the other method of creating the library in the system ASP and the journals, journal receivers, and save files in a user ASP, then the files to be journaled must be in the system ASP. The journals can be in either the system ASP or isolated in a user ASP.
4. Optionally, start journaling the access paths of one or more physical or logical files using the Start Journaling Access Paths (STRJRNAP) command. All members for each file have their access path journaled. See "Access Path Recovery" on page 2-28 for a complete discussion of journaling access paths and access path recovery.
5. Save the files and members after you start journaling. You cannot apply journaled changes to the restored files and members using the Apply Journaled Changes (APYJRNCHG) command if the restored files and members were not being journaled when they were saved. If you do save access paths (when saving the files), the system does not rebuild them after restoring the files and access paths. Saving access paths does not affect IPL recovery (or delayed recovery) of access paths.

Run the applications that use those journaled files when you have completed these steps.

Creating a Journal Receiver

You must create journal receivers before you create the journal to which they will be attached. To create a journal receiver, use the CRTJRNRCV command and specify the name of the previously created receiver when you create it. The following command creates a journal receiver RCVDST1:

```
CRTJRNRCV JRNRCV(DSTJRN/RCVDST1) THRESHOLD(10000)
     TEXT('RECEIVER FOR DSTJRN JOURNAL')
```

In the above example, the receiver is placed in the library DSTJRN.

To create the journal receiver in a user ASP, first create the library for the journal receiver in the user ASP, or if the desired library is in the system ASP, specify a user ASP number for the ASP parameter on the CRTJRNRCV command. If you do use the ASP parameter, only journals, journal receivers, and save files will be allowed in the user ASP. You can use the THRESHOLD parameter to have a message sent to the journal threshold message queue when the receiver

reaches a specified size. In the above example, if the size of the receiver exceeds 10 000KB (the THRESHOLD parameter value), the system sends a message to the threshold message queue specified on the CRTJRN or CHGJRN command and journaling continues. You may want to change the receiver when this occurs.

The UNIT parameter on the CRTJRNRCV command is no longer supported. The UNIT parameter is kept only for syntax compatibility with previous releases of the Operating System/400.

Creating a Journal

Journals are created using the Create Journal (CRTJRN) command after the journal receiver has been created. Specify the name of the journal and one or two journal receivers. For example, the following command creates a journal with the name JRNLA in library DSTJRN and attaches a previously created journal receiver (RCVDST1) to the journal.

```
CRTJRN JRN(DSTJRN/JRNLA) JRNRCV(DSTJRN/RCVDST1)
     MSGQ(QSYS/QSYSOPR) TEXT('Distribution Journal')
```

To specify that the journal be created in a user ASP, first create the library for the journal in the user ASP. If the library for the journal is in the system ASP, it is possible to specify a user ASP number for the ASP parameter on the CRTJRN command. However, it is not recommended.

Start Journaling Physical File

When using the Start Journal Physical File (STRJRNPF) command, you can specify the OMTJRNE (*OPNCLO) parameter to omit journal entries for open and close operations for specific files from being journaled. This can be done to reduce the number of entries in a journal. Be aware that without open and close entries in the journal, the history of file access (open for input, output, update, or delete) does not exist.

If you omit the open and close journal entries, you cannot use the TOJOBO or TOJOBC parameters on the APYJRNCHG or RMVJRNCHG command. The journal entries that specify the job's first open or close of any physical file member has not been added.

An alternative way to reduce the number of open and close entries in a journal is to specify SHARE (*YES) for the file when it is created. The system writes a single open and close entry regardless of how often the shared open data path (ODP) is opened or closed within a routing step.

The following command starts journaling for the physical file ORDENTP:

```
STRJRNPF  FILE(DSTPRODLIB/ORDENTP) JRN(DSTJRN/JRNLA)
      IMAGES(*BOTH)
```

After you have run this command, any changes made to the physical file ORDENTP are recorded in the journal JRNLA whether the changes are made directly to the physical file ORDENTP or through logical files of that physical file. Both before- and after-images will be journaled.

Start Journaling Access Paths

The Start Journal Access Path (STRJRNAP) command is used to start journaling, to a specific journal, an access path or access paths for all members of a database file. Any new members that are added to the file will also have their access paths journaled.

If the physical file is keyed, journaling can be started for the file's access path. When access path journaling is started for the physical file, only the access paths for the physical file members are journaled. Journaling for any logical files is started only when access path journaling is started for the logical files.

The journal entries created for access path journaling cannot be used in any operation that applies or removes journal changes. These entries are used only to recover access paths after the system ends abnormally.

Follow these steps to begin access path journaling:

1. Identify those access paths you wish to journal. Consider journaling your large access paths (those that would take a long time to rebuild if the system ended abnormally).
2. Use the Start Journal Physical File (STRJRNPF) command to journal all physical files associated with the access paths you want protected (if you are not already journaling the physical files).
3. Start journaling the selected access paths using the Start Journal Access Path

(STRJRNAP) command. Remember that you can journal both physical file and logical file access paths. Specify the file whose access path you want to journal on the STRJRNAP command. The following command begins journaling the access path for the logical file ORDENTL:

```
STRJRNAP FILE(DSTPRODLIB/ORDENTL) JRN(DSTJRN/JRNLA)
```

Saving Files

Save the physical files after you start journaling them and after each new member is added to the files. Logical files dependent on the physical files, should also be saved in case the physical file becomes damaged. If you save access paths (when saving the files) the system does not have to rebuild the access paths after restoring the files and access paths. You can use the Save Changed Objects (SAVCHGOBJ) command and specify OBJTYPE(*FILE) OBJJRN(*YES), or you can use the Save Object (SAVOBJ) or Save Library (SAVLIB) command to save files. Saving access paths does not affect IPL recovery.

When you start journaling a physical file, the system assigns a unique journal identifier (JID) to every member in the file. This unique journal identifier is part of every journal entry added to the journal receiver for a given file member. This identifier is the way that the journal entry is associated to the corresponding journaled object. The copy of the physical file on the save media that was saved before it was journaled does not have the journal identifier saved with it. Therefore, if this copy of the file is restored to the system, the journal entries cannot be associated with the file and cannot be applied. This is why it is critical to save the journaled file after journaling is started, and every time a member has been added to it. This ensures that the journal identifiers are saved with the file members.

Displaying Journal Status

The Work with Journal Attributes display lets you display information about the journal and related journal receivers. Use the WRKJRNA command. This display identifies:

- The journal receivers currently attached to the journal
- A directory of the journal receivers still on the system that are associated with the journal
- The names of all physical files being journaled through the journal
- The names of all database files that are having their access paths journaled

Working with Receiver Directory

Figure 3-1 shows an example of the Work with Receiver Directory display.

```
                        Work with Receiver Directory

Journal . . . . . :   JRNLA          Library . . . . . :   DSTJRN

Total size of receivers . . . . . :   155648

Type options, press Enter.
  4=Delete   8=Display attributes

                                      Attach                    Save
Opt  Receiver    Library     Number   Date      Status          Date
 _    RCVDST1     DSTJRN      02001   06/08/90  ONLINE          00/00/90
 _    RCVDSTA1    DSTJRN      03001   06/09/90  PARTIAL         06/90/90
 _    RCVDSTA2    DSTJRN      03001   06/09/90  ONLINE          00/00/90
 _    RCVDSTB1    DSTJRN      03002   06/10/90  PARTIAL         06/11/90
 _    RCVDSTB2    DSTJRN      03003   06/10/90  SAVED           06/11/90
 _    RCVDSTC1    DSTJRN      04001   06/11/90  ATTACHED        06/12/90



                                                                Bottom
F3=Exit   F5=Refresh   F11=Display size   F12=Cancel
```

Figure 3-1. Work with Receiver Directory

The *PARTIAL* status of the journal receiver on this display indicates the following:

- A journal receiver was saved while it was attached to the journal. This means that additional entries will be recorded in the journal attached to this receiver after the save operation has occurred.
- The receiver was later restored, and no complete version is available.
- A partial receiver does not contain all the entries recorded in the journal while this receiver was attached. It does contain entries recorded up to the last save operation.

You can use partial receivers to apply or remove changes from a file. If you attempt to restore a saved receiver while a more current version of the receiver is on the system, an escape message is sent to prevent you from restoring the receiver. The system makes sure the most complete version is preserved.

You can use a partial receiver as the last receiver in the receiver chain for an APYJRNCHG command only if you specify a sequence number for the TOENT parameter. You can use a partial receiver as the first receiver in the receiver chain for a RMVJRNCHG command only if you specify a sequence number for the FROMENT parameter.

The system does not allow you to delete a journal receiver from the middle of the receiver chain. This ensures logical recovery. However, if a journal receiver is damaged, you can delete it from the middle of the chain. If an attached journal receiver is damaged, you must detach the damaged receiver (CHGJRN command) before you can delete it.

The system does not prevent you from deleting a receiver that was once attached and is not saved or that is required to provide adequate recovery. If you try to delete a journal receiver that was once attached but has not been saved, the system issues an inquiry message. You can then continue or cancel the delete operation. You may use the system reply list to specify the reply the system is to send for this inquiry message (rather than explicitly responding to each inquiry message).

You must ensure that the journal receivers are not deleted until you no longer need them. You can display the journal receiver directory from the Journal Attributes display to determine which journal receivers have been saved. A date of 00/00/00 in the *Saved* column indicates that a journal receiver has not been saved.

End Journaling Physical File

Use the End Journal Physical File (ENDJRNPF) command to end journaling of changes for a specific physical file and all its members. Journaling of a member stops if you remove it from a journaled file. Journaling of a file stops if you delete it.

Ending Access Path Journaling

To end journaling the access paths for a physical or logical file, use the End Journal Access Path (ENDJRNAP) command. In addition, if you remove a member from a file that is journaling its access path, journaling for that member's access path is implicitly ended, unless the access path is shared and journaled by another file member. If a file that is journaling its access paths is deleted, journaling is implicitly ended for all the file member's access paths unless a particular access path is being shared and journaled by another file.

A physical file cannot have journaling ended for its members if the physical file or any logical file based on that physical file is journaling its access path. All physical files under a logical file must be journaled if that logical file is journaling its access path.

Work with Journal (WRKJRN) Command Options

The Work with Journal (WRKJRN) command shows a display that provides you with options to display journal status and guides you through various types of recovery involving journals, journal receivers, and journaled files. The options are shown on the following display:

```
                              Work with Journals

Type options, press Enter.
  2=Forward recovery       3=Backout recovery   5=Display journal status
  6=Recover damaged journal   7=Recover damaged journal receivers
  9=Associate receivers with journal

Opt  Journal     Library     Text
     JRNACC      DSTA1       JOURNAL FOR ACCOUNTS









                                                                      Bottom
Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Recovery Options

When it is necessary to restore an object during the following recovery procedures, the system prompts for the values needed with a prompt of the RSTOBJ command. The values known for OBJ, SAVLIB, and OBJTYPE parameters appear, but they cannot be changed. If the values for DEV, VOL, SAVF, SAVDATE and SAVTIME can be determined by the system (depending on the amount of damage), they are also filled in. If the values are not shown, you must enter the values.

If more than one object is restored, the system groups the objects according to the library specified in the SAVLIB parameter. Within each group, the system further orders the objects according to the values specified for the DEV, VOL, SAVF, SAVDATE, and SAVTIME parameters. This grouping results in a minimum number of restore object prompts as a result of information known to the system.

Work with Forward Recovery: Option 2 on the Work with Journals display shows the Work with Forward Recovery display and a list of the file members being journaled.

```
                        Work with Forward Recovery

Journal . . . . . :   JRNACC          Library . . . . . :   DSTA1

Position to . . . . .
  Library . . . . . .

Type options, press Enter.
  1=Add member to list   2=Apply journaled changes   3=Restore
  4=Remove member from list

Opt     File          Library       Member        Status
 _      PAYFILE1      PAYLIB        QTR192
 _

                                                                  Bottom
F3=Exit   F12=Cancel
```

The Work with Forward Recovery display contains a status field for each file member. The status field for each member indicates the following:

- NOT FOUND. The system cannot locate the specified database file.
- DAMAGED. The member is damaged and needs to be restored.
- NOT SYNCHRONIZED. The journal receiver used for this member is damaged and needs to be restored.
- RESTORE COMPLETE. The restore of the member is complete.
- RECOVERED. Recovery has completed successfully.
- NOT JOURNALED. The member is not journaled to any journal and cannot be recovered.
- DIFFERENT JOURNAL. The member is not journaled to the journal you are working with.
- Blank. The member and the journal are usable and everything is synchronized.

Option 1 (Add member to list) on the display allows you to add a member to the list. You may want to do this if you want to restore those members.

Option 2 (Apply journaled changes) shows the APYJRNCHG command prompt, which applies the journaled changes to the file members, and changes the status to RECOVERED (if the apply operation was successful). If the apply operation was not successful, messages appear indicating why, and the status remains the same.

If any required receivers are missing or damaged while running the APYJRNCHG command, the system displays prompts for the restore procedures for the missing or damaged receivers.

If any of the members in the list have a status of DAMAGED when option 2 is selected, you are prompted with the command necessary to recover the file member. For files that are damaged, recovery involves the restore of the last save followed by the Apply Journaled Changes (APYJRNCHG) command. The system guides you through recovery as follows:

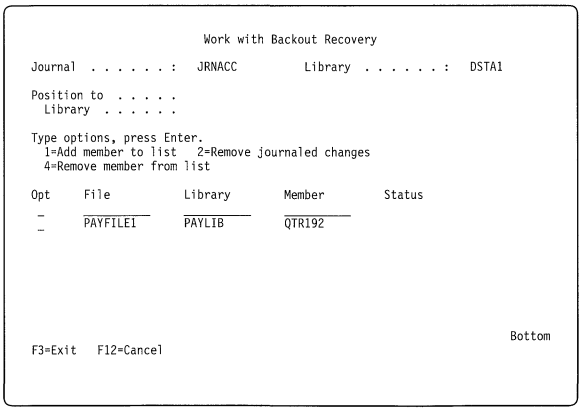
1. The system identifies all the logical files dependent on the specified damaged file. The Dependent Logical Files display appears identifying these files.
2. The dependent logical files are deleted.
3. The files to be recovered (restored) are deleted by the system.
4. The system displays prompts for the restore of files to be recovered. After all restores are completed successfully, the files to be recovered are allocated exclusively to prevent any other processing. This allocation is maintained until the recovery procedures are complete.
5. The system displays prompts for the restores of the dependent logical files.
6. An APYJRNCHG command is prompted with FROMENT(*LASTSAVE) and TOENT(*LASTRST).
7. If the APYJRNCHG command encounters a required journal receiver that is not on-line, the system prompts for the restore of the required receiver and again starts the APYJRNCHG command.

When the recovery process is complete, the status field for the member indicates RECOVERED (if the operation was successful). If the operation failed, the status field remains unchanged, and messages appear indicating why the operation failed.

Option 3 (Restore) prompts you for the files to restore. Use this option if any members have a status of NOT FOUND. Members that are restored successfully have a status of RESTORE COMPLETE. Members that are not restored keep their old status. A message is sent indicating that the restore did not complete successfully. All members restored are included in the list of members to recover.

Option 4 (Remove member from list) on the display removes file members from the list of members to be recovered.

Work with Backout Recovery: The same options on the Work with Forward Recovery display are available on the Work with Backout Recovery display (option 3 on the Work with Journals display). In addition, the option to restore the file is not valid for backout recovery. The status field shown on the Work with Backout Recovery display is either blank or it indicates the same status as for forward recovery, except for RESTORE COMPLETE.



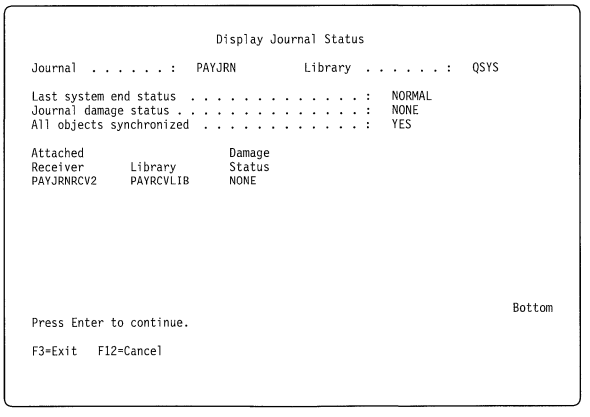
Option 1 (Add member to list) on the display allows you to add a member to the list.

Option 2 (Remove journaled changes) on this display shows the Remove Journaled Changes (RMVJRNCHG) command prompt, removes the journaled changes, and changes the status to RECOVERED (if the operation was successful). If any required journal receivers are missing or damaged while the RMVJRNCHG command is running, the system displays prompts for the necessary restore procedures for the missing or damaged receivers. If the remove operation was not successful, messages appear indicating why the status remains the same.

If any members in the list have a status of NOT FOUND or DAMAGED when on the Work with Backout Recovery display, the operation is not allowed. These members must be recovered in a forward fashion after they have been restored. Forward recovery of specific files must be used for this type of recovery.

Option 4 (Remove member from list) on this display allows you to remove file members from the list.

Display Journal Status: Option 5 on the Work with Journal display shows the current status of the journal. It shows if the last system end was NORMAL or ABNORMAL, and if the journal is damaged. The damage status is NONE or FULL.



If the last system end was abnormal, this display indicates whether objects being journaled were synchronized (that is, whether each object in use during the abnormal end was corrected to match the entries in the attached journal receivers during the previous IPL).

If the last system end was normal, the display indicates that all objects are synchronized with the journal. If the journal is damaged, the display indi-

cates that the system was unable to determine whether or not all objects are synchronized.

The display also presents information about the currently attached receivers and their damage status. The damage status of the receivers can be NONE, PARTIAL, or FULL. If the journal is damaged to the extent that the status of the attached journal receivers cannot be determined, there are no attached receivers shown on the display.

If some objects are not synchronized or damage has been detected, a message appears indicating the form of recovery that you should perform.

Recover Damaged Journal: Option 6 on the Work with Journals display verifies that the journal is damaged before proceeding with recovery. If the journal is not damaged, an information message appears.

Recovery for a damaged journal guides you through the following steps:

1. The system attempts to determine which files are currently being journaled to the indicated journal. If the system cannot successfully build this list, a message appears before the recovery operation begins.
2. Journaling is ended for all access paths currently being journaled to the specified journal.
3. Journaling is ended for all files currently being journaled to the specified journal.
4. The system deletes the journal.
5. The system presents the Recover Damaged Journal display, which asks you whether to restore or create the journal:
   a. If the journal will be restored, the system prompts for the values needed for the restore operation.
   b. If the journal will be created, the system prompts for the receiver names and attributes with the CRTJRNRCV command prompt. The system prompts for values needed to create the journal with the CRTJRN command prompt, with known values shown.
6. The list of files for which journaling is to be started again is shown. When you press the Enter key, journaling is started for all files listed.
7. The list of files containing access paths for which journaling is to be started again appears. When you press the Enter key, journaling for the access paths is started for the files listed.
8. The system associates all applicable receivers with the re-created or restored journal so that a restore of these receivers is not necessary.

   A journal receiver is associated with a journal if the journal appears in the journal receiver directory. A receiver that was previously attached to a journal, but is not currently associated with a journal cannot be used. For example, the receiver cannot be used with the DSPJRN, APYJRNCHG, or RMVJRNCHG commands.

As the recovery of a damaged journal proceeds, the Display Journal Recovery Status display appears. The information on this display is updated as the operation progresses to indicate which steps have been completed, which steps have been bypassed, and which step will be run next. Whenever a user action is required, the status display is replaced by the appropriate prompt display.

The status field indicates the following operation status:

- PENDING. The step has not been started.
- NEXT. The step will be performed next (after the Enter key is pressed).
- BYPASSED. The step was not performed. (It was not necessary).
- COMPLETE. The step has been performed.

The first display you usually see after the first status display is the Recover Damaged Journal display. Use this display to choose whether the journal is to be created or restored.

When the last step of the recovery process is complete, a message appears indicating that all files for which journaling was started should be saved to establish a new recovery point.

Recover Damaged Journal Receivers:

Option 7 on the Work with Journals display checks to determine which journal receivers associated with the specified journal are damaged. If none are damaged, a message appears.

If there are damaged journal receivers associated with the specified journal, the Recover Damaged Journal Receivers display appears and lists those receivers.

The status fields initially show a value of DAMAGED. After recovery has been successfully completed, the status shows a value of RECOVERED (receiver recovered).

Recovery for a damaged journal receiver guides you through the following steps:

1. If the attached receivers are damaged, a CHGJRN command must be run before they can be recovered.

   Indicate whether existing (empty) receivers will be used or if new ones will be created. If new receivers are to be created, the CRTJRNRCV command prompts are presented for receiver name and attributes. After the new receivers are created, the CHGJRN command prompt is shown.

   If the attached receiver is not damaged, the preceding step is omitted.
2. The damaged journal receiver is deleted.
3. Prompts for the restore of the damaged journal receivers are shown. Any of the values on the prompt can be changed except the receiver name. Save information in the prompt is provided by the system.

Associate Receivers with Journal:

Option 9 on the Work with Journals display should be used if the journal was restored or created again. The system associates all applicable receivers with the restored or recreated journal so that a restore of these receivers is not necessary.

A journal receiver is associated with a journal if the journal appears in the journal receiver directory. A receiver that was previously attached to a journal but is not currently associated with a journal cannot be used. For example, the receiver cannot be used with the DSPJRN, APYJRNCHG, or RMVJRNCHG commands.

Recovery of a Physical File Using Journaled Changes

In addition to the file recovery information that follows, you can use the Work with Journals (WRKJRN) command to help recover journaled files.

You can recover from many types of damage to database file members using journaled changes. For example, a file member is damaged and becomes unusable, an error in an application program caused records to be improperly updated, or incorrect data was used to update a file. In each of these instances, simply restoring a saved version of the file may result in the loss of a significant amount of data. If you use the Apply Journaled Changes (APYJRNCHG) command to apply journaled changes, significantly less data may be lost. You can use the Remove Journaled Changes (RMVJRNCHG) command to recover from improperly updated records or incorrect data if before-images have been journaled. This command removes (or backs out) changes made to a file.

Note: To apply or remove journaled changes to or from a saved copy of the file, the file must have been saved while it was being journaled.

For an explanation of why files must be saved, see "Saving Files" on page 3-3.

Recovery after Abnormal System End

If the system abnormally ends while files are being journaled, the system does the following:

1. Brings all journals, journal receivers, and files being journaled to a usable and predictable condition during the IPL, including any access paths being journaled and in use at the time the system abnormally ended.
2. Checks all recently recorded entries in the journal receivers that were attached to a journal.
3. Places an entry in the journal to indicate that an abnormal system end occurred. When the system completes the IPL, all entries are available for processing.

4. Checks that the journal receivers attached to journals can be used for normal processing of the journal entries. If some of the files being journaled could not be synchronized with the journal, the system sends a message (CPF3172) to the history log (QHST) that identifies the journals that could not be synchronized. If a journal or a journal receiver is damaged, the system sends a message to the history log identifying the damage that occurred (CPF3171 indicates that the journal is damaged, and the message CPF3173 or CPF3174 indicates that the journal receiver is damaged).
5. Recovers each physical file member that was in use at the time the system ended abnormally, using the normal system recovery procedures for database files. See the manual *Database Guide*, SC41-9659, for complete information on database files.

   In addition, if a physical file being journaled (or a logical file defined over that physical file) was opened for output, update, or delete operations, the system performs the following functions so changes to that file will not be lost:

   a. Ensures that the changes appearing in the journal receiver after the IPL also appear in the database file. Changes that do not appear in the journal receiver are not in the database file.

   b. Places an entry in the journal receiver indicating whether the file was synchronized with the journal. If the file could not be synchronized with the journal, the system places a message (CPF3175) in the history log identifying the failure, and you must correct the problem.

   A synchronization failure can occur if the data portion of the member is damaged, a journal receiver required to perform the synchronization is damaged, or the journal is inoperable.

Procedure for Abnormal System End Recovery: After an abnormal system end, perform the following:

1. Perform a manual IPL.
2. Check the history log to determine if there are any damaged files, files that are not synchronized, or any damaged journals or journal receivers.

   An alternative is to use the WRKJRN command. See "Work with Journal (WRKJRN) Command Options" on page 3-4 for more information.
3. If necessary, recover the damaged journals or journal receivers as described in "Recovering When a Journal Is Damaged" on page 3-10 and "Recovering When a Journal Receiver Is Damaged" on page 3-10.
4. If there is a damaged file:

   a. Delete the file.

   b. Restore the file from the latest saved version.

   c. Allocate the file so no one else can access it.

   d. Restore the needed journal receivers from newest to oldest, if they are not on-line.

   e. Use the APYJRNCHG command to apply the changes to the file.

   f. Deallocate the file.

   WRKJRN can be used to recover damaged files. See "Work with Journal (WRKJRN) Command Options" on page 3-4.
5. If a file could not be synchronized, use the information in the history log and in the journal to determine why the file could not be synchronized and how to proceed with recovery. For example, you may need to use the DFU or a user-written program to bring the file to a usable condition.
6. Determine which applications or programs were active, and determine where to restart the applications from the information in the history log and in the journal.

If a journaled access path is in use during an abnormal system end, that access path does not appear on the Edit Rebuild Access Path display.

If the maintenance for the access path is immediate, the system automatically recovers the access path during IPL. A status message appears for each immediate maintenance access path as it is being recovered during IPL.

The system places a message (CPF3208) in the system history log for each access path recovered through the journal during IPL. If the maintenance for the access path is delayed, the system auto-

matically recovers the access path during the next database file open operation using the access path entries recorded in a journal.

Recovering When a Journal Is Damaged

If a journal becomes damaged, the system sends the message CPF8135 to the system operator and to the job log. Use the Work with Journal (WRKJRN) command to help you in the recovery of a damaged journal. Select option 6 (Recover damaged journal) on the Work with Journals display to recover a damaged journal. For a description of the Work with Journals display, see "Work with Journal (WRKJRN) Command Options" on page 3-4, or the WRKJRN command in the manual *Programming: Control Language Reference*, SC41-0030.

It is recommended that you use the Work with Journals (WRKJRN) command to recover a damaged journal. The WRKJRN command performs all the steps described below except for saving the physical files and logical files. The WRKJRN command associates the receivers with the recovered journals without you having to delete and restore the receivers.

Use the following steps to recover a damaged journal without using the WRKJRN command:

1. End journaling for all access paths associated with the journal by using the ENDJRNAP command.
2. End journaling for all physical files associated with the journal by using the ENDJRNPF command.
3. Delete the damaged journal by using the DLTJRN command.
4. Create a journal receiver (CRTJRNRCV command) and create a journal with the same name and in the same library as the damaged journal (CRTJRN command), or restore the journal from a previously saved version.
5. Start journaling the physical files and, if desired, the access paths by using the Start Journal Physical File (STRJRNPF) and Start Journal Access Path (STRJRNAP) commands for the files, or by deleting and restoring all the files that were being journaled. Physical files and access paths that were journaled at the time of their save automatically begin journaling at restore time if the journal is on-line.
6. Save the physical files and all associated logical files to allow for later recovery.
7. Associate the old journal receivers with the new journal. Save the journal receiver that was attached to the damaged journal. Delete it and restore it and any previously attached journal receivers you need. You must delete and then restore the receivers after the journal is restored or re-created in order to associate the journal receivers with the journal. To display the names of the associated journal receivers, use the Work with Journal Attributes (WRKJRNA) command and take the option to Work with Receiver Directory.

Each time a journal is restored, a new receiver chain is started because the last journal receiver on the chain that existed prior to the restore process did not have the newly created receivers as its next receivers.

Recovering When a Journal Receiver Is Damaged

If a journal receiver becomes damaged, the system sends the message CPF8136 or message CPF8137 to the system operator and the job log. To recover from a damaged receiver, do the following:

- If the damaged receiver is currently attached to a journal, use the Change Journal (CHGJRN) command to attach a new receiver and detach the damaged receiver. (If dual receivers exist, the system continues to place entries on the receiver that is not damaged until a CHGJRN command is run.)
- If the journal receiver is not currently attached to a journal, delete the journal receiver using the Delete Journal Receiver (DLTJRNRCV) command and restore a previously saved copy.
- If the journal receiver was never attached to a journal, delete the receiver and create it again or restore it.

If the journal receiver is partially damaged, all journal entries except those in the damaged portion of the journal receiver can be listed using the Display Journal (DSPJRN) command. Using

this list, you can determine what you need to do to recover your files. Applying or removing journal changes cannot be done with a partially damaged journal receiver.

It is recommended that you use the Work with Journals (WRKJRN) command to recover a damaged journal receiver. For a description of the Work with Journals display, see "Work with Journal (WRKJRN) Command Options" on page 3-4, or the WRKJRN command in the manual *CL Reference*. For a description of the journal menus, see the on-line information.

Applying and Removing Journaled Changes

The Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG) commands are used to apply and remove changes made to a file member.

Note: To use the APYJRNCHG or RMVJRNCHG command to recover a physical file member, the member must be currently journaled. The journal entries must have the same journal identifier as the member. To ensure the journal identifiers are the same, save the physical file immediately after journaling is started for the file and each time a member is added to the file. "Saving Journaled Files" on page 2-7 has more information about saving journaled files.

Applying Journaled Changes

If a file member becomes damaged or is not usable, you can recover the file using the Apply Journaled Changes (APYJRNCHG) command directly, or by using the Work Journal (WRKJRN) command and following the prompts. For a description of the journal menus, see the on-line information or refer to the topic "Work with Journal (WRKJRN) Command Options" on page 3-4. You must first reestablish the physical file member to a condition that you know is undamaged.

- To reestablish the member, restore the last saved copy of the file. The file must have been saved while it was being journaled and the member existed.
- Use the WRKJRN command options to help in the restore or apply operation.
  - If you saved the file using the CPYF command, use the CPYF command to restore the member.
  - If the member was just initialized, initialize the member again using the Initialize Member (INZPFM) command or a user-created application program.
  - If a member was just reorganized, reorganize the member again using the Reorganize Physical File Member (RGZPFM) command.

For more information about the CPYF, INZPFM, and RGZPFM commands, see *CL Reference*.

If the journal receivers are deleted or saved with their storage freed since the file was last saved (or since some other point), you must restore the needed journal receiver, from the newest to oldest. The system applies the changes to the file in the same order as they were originally made. When you use the APYJRNCHG command, the file cannot be in use by anyone else.

When the condition of the member has been established, use the APYJRNCHG command to apply the changes recorded in the journal to the file. On the APYJRNCHG command, specify the first journal entry to be applied to the member. This entry can be selected from any of the following points:

- After the last save of the member
- From the first journal entry
- From an identified sequence number that corresponds to a date and time stamp
- From an identified sequence number that corresponds to the start or end of a particular job's use of the member (if you did not specify OMTJRNE(*OPNCLO) when starting journaling for the file)

You can stop applying the journal entries at:

- The end of the data in the last journal receiver in the receiver range
- A particular entry in the journal
- A date and time stamp
- A commitment boundary
- The start or end of a particular job's use of the data in the member (provided you did not specify OMTJRNE (*OPNCLO))

- The journal entry indicating when the member was last restored

You can ensure that commitment transaction boundaries are honored on the apply journaled changes operations by using the commit boundary (CMTBDY) parameter on these commands.

Use the Display Journal (DSPJRN) command to identify the desired starting and ending points. If you use a control language (CL) program for your recovery procedures, use the Retrieve Journal Entry (RTVJRNE) command to retrieve a journal entry and place it in program variables. The Receive Journal Entry (RCVJRNE) command can also be used for recovery. See the *CL Reference* for more information.

Apply Journaled Changes (APYJRNCHG) Command Examples

The following command applies the changes in journal JRNA to the first member of all files in the library DSTPRODLIB that are being journaled to journal JRNA:

```
APYJRNCHG JRN(JRNLIB/JRNA) FILE((DSTPRODLIB/*ALL))
```

Because the RCVRNG parameter is not specified, the system determines the range of journal receivers to use as a result of the save information for the files. Because the FROMENT parameter is not specified, the system applies the changes beginning with the first journal entry after the member was last saved.

If the file was last saved with the save-while-active function, the saved copy of each file member includes all record-level changes in the journal entries up to the corresponding F SS journal entry. In this case, the system applies changes beginning with the first journal entry following the F SS entry.

If the file was last saved when it was not in use (normal save), the saved copy of each member includes all record-level changes in the journal entries up to the corresponding F MS member saved journal entry. In this case, the system applies changes beginning with the first journal entry following the F MS entry.

The following command applies the changes to the file from the journal receivers currently attached to the journal:

```
APYJRNCHG  JRN(JRNLIB/JRNA) FILE((LIBA/FILEA MBR1))
     RCVRNG(*CURRENT) FROMENT(*FIRST) TOENT(*LAST)
```

The system applies the changes from the first journal entry in the currently attached receiver to the last journal entry in the currently attached receiver. Changes are applied to member MBR1 of the file FILEA.

The following command applies the changes in the journal JRNA to all members of the file FILEA beginning with the first journal entry after the file member was last saved:

```
APYJRNCHG  JRN(JRNLIB/JRNA)   FILE((LIBA/FILEA *ALL))
     TOJOBC(000741/USERP/WORKSTP)
```

The operation continues until the specified job closes any of the members in the file that it opened. The operation is *not restricted* only to those journal entries recorded by the specified job.

Note: This example works only if you do not specify OMTJRNE (*OPNCLO) when starting journaling for the file.

Removing Journaled Changes

Depending on the type of damage to the physical file and the amount of activity since the file was last saved, removing changes from the file can be easier than applying changes to the file. Use the Remove Journaled Changes (RMVJRNCHG) command directly or the Work with Journal (WRKJRN) command and follow the prompts to remove (or back out) changes from a file member if before-images were journaled. For a description of the journal menus, see the topic "Work with Journal (WRKJRN) Command Options" on page 3-4, or the online information for the WRKJRN command. The changes are removed in reverse chronological order from the order in which they were originally made to the file.

On the RMVJRNCHG command, you identify the first journal entry to be removed from the file member. This entry can be from:

- The last journal entry contained within the range of journal receivers specified
- The entry corresponding to the last save of the member
- An identified sequence number

You can control the changes that are removed from the file. For example, assume an application updated entries incorrectly for a period of time. In this case, you could remove the changes from the member until that application first opened the member.

You can stop removing journaled changes at:

- The end of data in the journal receivers. (This corresponds to the first journal entry that was recorded on the range of journal receivers specified.)
- An identified sequence number that corresponds to a particular entry in the journal.
- The start of a particular job's use of the member (if you did not specify OMTJRNE (*OPNCLO)when starting journaling for the file).

You can ensure that commitment transaction boundaries are honored on the remove journaled changes operations by using the CMTBDY parameter on these commands.

Use the Display Journal (DSPJRN) command to identify the desired starting and ending points for removing the changes. If you use a control language (CL) program for your recovery procedures, use the Retrieve Journal Entry (RTVJRNE) command to retrieve a journal entry and place it in program variables. The Receive Journal Entry (RCVJRNE) command can also be used for recovery. See the *CL Reference* for more information.

Remove Journaled Changes (RMVJRNCHG) Command Examples

The following command removes the changes in journal JRNA from the first member of FILEA:

```
RMVJRNCHG JRN(JRNLIB/JRNA) FILE(DSTPRODLIB/FILEA)
     RCVRNG(*CURRENT)
```

The system starts removing the changes beginning with the latest entry for that member on the currently attached journal receiver and continues to the earliest entry for that member on the currently attached journal receiver.

The following command removes the changes in journal JRNA from the first member of FILEA:

```
RMVJRNCHG JRN(JRNLIB/JRNA) FILE(DSTPRODLIB/FILEA)
     RCVRNG(JRNLIB/RCVA10 JRNLIB/RCVA8)
```

The system starts removing the changes beginning with the last entry (the latest entry) for that member in journal receiver RCVA10 and continues to the first entry (the earliest entry) for that member on journal receiver RCVA8.

The following removes the changes in JRNA from all members in FILEA from the last save entry to entry number 1003.

```
RMVJRNCHG JRN(JRNLIB/JRNA) FILE(DSTPRODLIB/FILEA)
  RCVRNG(*CURRENT) FROMENT(*LASTSAVE) TOENT(1003)
```

If the last save operation used the save-while-active function, the system starts removing changes from the entry preceding the last F SS start of save entry. If the last save operation was a normal save operation, the system starts removing changes from the entry preceding the last F SS member saved entry. In the example, journaled changes are removed back to entry 1003.

Actions of the APYJRNCHG or RMVJRNCHG Command by Journal Code

Table 3-1 on page 3-13 shows the actions taken by the APYJRNCHG or RMVJRNCHG command by journal code and entry type. If All is specified for the *Entry Type*, it indicates that all entry types for that journal code have the specified actions taken by the APYJRNCHG or RMVJRNCHG command.

Table 3-1 (Page 1 of 3). Actions by Journal Code and Entry Type

Journal Code	Entry Type	Operation	APYJRNCHG	RMVJRNCHG
A	All		Ignores	Ignores
C	All		Ignores	Ignores
A	All		Ignores	Ignores

Table 3-1 (Page 2 of 3). Actions by Journal Code and Entry Type

Journal Code	Entry Type	Operation	APYJRNCHG	RMVJRNCHG
F	AY	Journaled changes applied	Ends	Ends
F	CE	Change end of data	Member end of data changed	Ends
F	CL	Member closed	Ignores	Ignores
F	CR	Member cleared	Member cleared of all records	Ends
F	EJ	End journaling	Ends	Ignores
F	EP	End journaling access paths	Ignores	Ignores
F	FD	Member forced to auxiliary storage	Ignores	Ignores
F	IU	Object synchronized (see note)	Ignores	Ignores
F	IU	Object not synchronized (see note)	Ends	Ends
F	IZ	Member initialized	Initialized records inserted in member	Initialized records deleted from member
F	JM	Start journaling member	Ignores	Ends
F	JP	Start journaling access paths	Ignores	Ignores
F	MD	Member deleted	Ends	Ends
F	MF	Member saved with storage freed	Ends	Ends
F	MM	Member moved	Ignores	Ignores
F	MN	Member renamed	Ignores	Ignores
F	MR	Member restored	Ends	Ends
F	MS	Member saved	Ignores	Ignores
F	OP	Member opened	Ignores	Ignores
F	PD	Access path deleted	Ignores	Ignores
F	PM	Logical owning member of access path moved	Ignores	Ignores
F	PN	Logical owning member of access path renamed	Ignores	Ignores
F	RC	Journaled changes removed	Ends	Ends
F	RG	Member reorganized	Ends	Ends
F	SA	Start of APYJRNCHG	Ends	Ends
F	SR	Start of RMVJRNCHG	Ends	Ends
F	SS	Start of save active	Ignores	Ignores
J	All		Ignores	Ignores
L	All		Ignores	Ignores
P	All		Ignores	Ignores
R	BR	Before-image updated for rollback operation	Ignores	Record updated with before-image
R	DL	Record deleted	Record deleted	Record updated with before-image
R	DR	Record deleted for rollback operation	Record deleted	Record updated
R	PT	Record written to member	Record written to member	Record deleted from member
R	PX	Record added directly to member	Record added	Record deleted from member
R	UB	Record updated (before-image)	Ignores	Record updated with before-image

Table 3-1 (Page 3 of 3). Actions by Journal Code and Entry Type

Journal Code	Entry Type	Operation	APYJRNCHG	RMVJRNCHG
R	UP	Record updated (after-image)	Record updated with after-image	Ignores
R	UR	After-image updated for rollback operation	Record updated with after-image	Ignores
S	All		Ignores	Ignores
T	All		Ignores	Ignores
U	User-specified	User entry	Ignores	Ignores

Note: The *Flag* field in the journal entry indicates whether the file is synchronized (hex F0 = file was synchronized; hex F1 = file was not synchronized).

In addition to the entries that cause the command to end, the system ends the Apply Journaled Changes (APYJRNCHG) or Remove Journaled Changes (RMVJRNCHG) command if any format error (such as an undefined entry for that file member) or logical error (such as updating a record that has not been inserted or a duplicate key exception) is encountered when the command is run.

For entries that end the APYJRNCHG or RMVJRNCHG command, a message identifying the reason for the system end is placed in the job log, and the corresponding change is not made to the file member. The message contains the sequence number of the journal entry on which the failing condition was detected. Analyze the error, make the necessary correction, and then start applying journal changes again using the appropriate sequence number.

For example, if the entry that caused the APYJRNCHG command to end is entry code F of type RG, you must reorganize the physical file member referred to in the journal entry. Use the same options that were originally specified on the reorganize request when the journal entry was recorded in the journal receiver. Resume applying journal changes starting with the journal entry following 'F RG' reorganize physical file member journal entry.

The APYJRNCHG and RMVJRNCHG commands send an escape message and end the journal operation if any required journal receiver defined by the RCVRNG parameter is not on the system and associated with the journal. Use the WRKJRNA command to select the journal receiver directory display, to see which journal receivers are on the system and associated with the journal. The escape message contains the name of the required journal receiver if the reason code of message CPF7053 is 1 or if message CPF9801 is sent.

When the processing of the APYJRNCHG or RMVJRNCHG command ends with an escape message, the members can be partially changed. To determine how many changes were applied or removed for each member, review the diagnostic messages in the job log prior to the final escape message for each member, or use the DSPJRN command to display the journal entries with an F journal code and an entry type of AY or RC. The *Count* field in the journal entry contains the number of journal entries applied or removed.

For more information on the journal codes and the journal entries, see "System-Created Journal Entries" on page 2-18.

Displaying and Printing Journal Entries

Use the Display Journal (DSPJRN) command to display journal entries. These entries can be displayed at a work station, printed, or written to a database file. (You cannot directly access the journal entries in the form in which they are contained in the journal receivers.)

You can use the list of the journal entries to:

- Prepare for the recovery of a particular file. The list contains the information you need to specify the starting and ending points for applying and removing journaled changes.

- Determine the functions that have been performed on the physical files being journaled (such as save and restore, clear, reorganize).
- Determine the functions that have been performed on the journal (such as attaching new journal receivers).
- Determine the functions that have been performed on the associated journal receivers (such as save and restore).
- Review the activity that has occurred on a file.
- Analyze journal entries for debugging or problem analysis.
- Analyze journal entries for an audit trail.
- Provide a report of a saved journal receiver.
- List the events that have occurred in the functioning of the journal (for example, when the files have been saved).

The DSPJRN command can selectively list journal entries for a particular member of a file, or the entries for all files within a particular library. You can further identify journal entries to be listed by specifying only:

- User-created entries
- Journal entries created to control the journal (journal codes F and J)
- Journal entries for specific entry types or journal codes
- Journal entries for a particular job, program, or file
- Any combination of these

Output for Journal Entries Directed to a Work Station

If you direct the output from the DSPJRN command to the requesting work station, basic information about the journal entries appears. Use the roll key to display the next sequential set of entries. When the receiver range includes an attached journal receiver, TOENT(*LAST) is specified on the command, and the last journal entries in the journal are displayed, press the Page Down key to display any new journal entries added to the attached receiver since the last time the Page Down key was pressed.

Output for Journal Entries Directed to a Database File

If you direct the output from the DSPJRN command to a database output file, you can further restrict the journal entries you want to process by creating logical files over the database output file.

Each journal entry occupies one record in the output file. Each has a fixed length portion for standard files. Before-images and after-images occupy separate records. The ENTDTALEN parameter controls the length of the field used to contain the record image. If the journal entry is smaller than the output file record, the journal entry is padded with blanks. If the journal entry is larger than the output file record, the remainder of the journal entry is truncated, and the system issues a warning message. To avoid truncation, specify the maximum record length in your files for the ENTDTALEN parameter on the DSPJRN command or specify *CALC for the ENTDTALEN parameter to allow the system to calculate the length of the specific data field so no entry is truncated.

If you write journal entries to a database output file, you can write applications programs that will process the data to:

- Write your own apply program.
- Correct data that has been incorrectly updated.
- Remove or review all changes made by a particular program.

  However, if you remove all changes made by a particular program, you could remove some valid updates. For example, assume that two work station users are using the same program to update a file, and one user enters some data that is not valid. If you remove all changes made by that program to remove the data that is not valid, you also remove the valid data entered by the other work station user.

Format of Database Output Files:

When you direct the output of the DSPJRN command to a database file, the system creates the output file records in a standard format. The system creates the database file in one of three standard formats determined by the value specified for the OUTFILFMT parameter:

Type	Format
***TYPE1**	The system uses the QJORDJE format in the model output file QADSPJRN to create the output file for the converted journal entries.
***TYPE2**	The system uses the QJORDJE2 format in the model output file QADSPJR2 to create the output file for the converted journal entries. In addition to all the fields included in the *TYPE1 format, this format includes two fields: one for the user profile which the job was running under when the journal entry was added, and one for the name of the system on which the output file was created.
***TYPE3**	The system uses the QJORDJE3 format in the model output file QADSPJR3 to create the output file for the converted journal entries. In addition to all the fields included in the *TYPE2 format, this format includes a field for the null value indicators that correspond to the record image in the journal entry.

You can create an output file to hold the output from the DSPJRN command but the format has to match the format of one of the IBM-supplied output files. See the DSPJRN command in the manual *Programming Reference Summary*, for a list of the IBM-supplied output files.

To process individual fields in the entry-specific data, you can use your high-level language (HLL) to subdivide the fields into subfields, or you can use the Retrieve Journal Entry (RTVJRNE) command and the substring built-in function.

Analyzing Your Journal Activity: You can use the output file created by the DSPJRN command to help analyze your journal entries. For example, you could determine how many of each type of entry (such as add or update) was done for a specific file or by a specific user.

Chapter 4. Commitment Control

Prerequisite: To understand the information in the following topic, it is assumed that you have a basic understanding of the journal management function on the AS/400 system. For more information about journal management, see Chapter 2, "Journal Management" on page 2-1.

Commitment Control Introduction

Commitment control is a function that allows you to define and process a group of changes to resources, such as database files or tables, as a single transaction. Commitment control allows you to design an application so that it can be started again if a job, an activation group within a job, or the system ends abnormally. The application can be started again with assurance that no partial updates are in the database due to incomplete transactions from a prior failure.

Commitment control is started and ended using the Start Commitment Control (STRCMTCTL) and the End Commitment Control (ENDCMTCTL) commands.

Commitment control allows you to:

- Ensure that all changes within a transaction are completed for all resources affected.
- Ensure that all changes within a transaction are removed if processing is interrupted.
- Remove changes made during a transaction when the workstation user application determines that a transaction is in error.

Throughout this topic, the term transaction is used. A **transaction** is defined as a group of individual changes to objects on the system that should appear as a single atomic change to the user. Commitment control ensures that either the entire group of individual changes occur on the system, or none of the individual changes occur on the system. An example of changes that can be grouped together is the transfer of funds from a savings to a checking account. To the user, this is a single transaction. However, more than one change occurs to the database because both savings and checking accounts are updated.

User transactions can be:

- Inquiries in which no database file changes occur.
- Simple transactions in which one database file is changed each time the Enter Key is pressed.
- Complex transactions in which one or more database files are changed each time the Enter key is pressed.
- Complex transactions in which each time the Enter key is pressed one or more database files are changed, but these changes represent only a part of a logical group of transactions.
- Simple or complex transactions that involve database files on a remote system. A transaction must be defined entirely to a single system, either to the local system or to a single remote system.
- Simple or complex transactions on the local system that involve objects other than database files.

Commit and Rollback Operations

Two operations affect changes made under commitment control:

- Commit

  All changes made under commitment control since the previous commit or rollback operation are made permanent and all locks related to the commitment transaction are released.

- Rollback

  All changes made since the previous commit or rollback operation are removed and all locks related to the commitment transaction are released.

The commit and rollback operations outlined are available in several AS/400 programming languages including RPG/400*, COBOL/400*, C/400*, PL/I, control language (CL), and Structured Query Language/400 (SQL/400*).

Terms Used with Commitment Control

Throughout the commitment control topics, the terms committable resource, committable change, commitment boundary, source system, and target system are used. The following information provides descriptions of these terms.

A **committable resource** is a local or remote AS/400 object that can be placed under commitment control. The following are different types of committable resources:

- Local and Remote AS/400 Database Files.

  See "Local and Remote AS/400 Database Files" on page 4-9 for more information about these committable resources.

- Local and Remote Objects Accessed by SQL.

  See "Local and Remote Objects Accessed by SQL" on page 4-10 for more information about these committable resources.

- Local API Commitment Resources.

  Applications may place other local objects under commitment control using the Add Commitment Resource (QTNADDCR) API. These are known as **API commitment resources**. See "Local API Commitment Resources" on page 4-10 for more information about these committable resources.

A **committable change** is a change to a committable resource that can be committed or rolled back. A committable change is referred to as *pending* or *uncommitted* until the change is committed or rolled back.

The following are committable changes that can be made into committable resources:

- Record-level changes made to local database files.
- Record-level changes made to remote database files using DDM or SQL.
- Object-level changes made to local or remote objects accessed using SQL. This includes Data Definition Language (DDL) changes such as Create SQL Package, Create SQL Table, and Drop SQL Table.

  Object-level changes to local database files and DDM files not accessed using SQL can *not* be committed.

- Changes made to local objects using the Add Commitment Resource (QTNADDCR) API.

A **commitment boundary** is established by performing a commit or rollback operation. For a particular commitment definition, a commitment boundary is defined to be any time there are no outstanding uncommitted changes for the objects being changed under commitment control. When a committable change is made to a committable resource, the current transaction is no longer at a commitment boundary.

A commit operation establishes a *new* commitment boundary. A rollback operation brings the objects being changed for the current transaction back to the *previous* commitment boundary.

The **source system** is the AS/400 system that asks for a change to an object or a record on a different system. The other system can be another AS/400 system or a different system. The source system is also referred to as the application requester.

The **target system** is the system that acts as the server when a source system requests a change to a remote object. The target system is the system on which the object actually resides. The target system is also referred to as the application server.

Commitment Definitions and Activation Groups

When commitment control is started, the system creates a **commitment definition.** Each commitment definition is known only to the job that started commitment control and is not known by any other job. The commitment definition contains information pertaining to the resources being changed under commitment control within that job. The commitment control information in the commitment definition is maintained by the system as the commitment resources change, until the commitment definition is ended.

A commitment definition generally includes:

- The parameters on the Start Commitment Control (STRCMTCTL) command.

- The current status of the commitment definition.
- Information about database files and other committable resources that contain changes made during the current transaction.

Multiple commitment definitions can be started and used by programs running within a job. Each commitment definition for a job identifies a separate transaction that has committable resources associated with it. These transactions can be committed or rolled back independently from transactions associated with other commitment definitions started for the job.

Scope for a Commitment Definition:

The **scope** for a commitment definition is used to indicate which programs that run within the job will use that commitment definition. The default scope for a commitment definition is to the activation group of the program that starts commitment control. Only programs that run within that activation group will use that commitment definition. Commitment definitions that are scoped to an activation group are referred to as **activation-group-level** commitment definitions. Many activation-group-level commitment definitions can be active for a job at one time. However, each activation-group-level commitment definition can be associated only with a single activation group. The programs that run within that activation group can associate their committable changes only with that activation-group-level commitment definition.

The alternative scope for a commitment definition is to the job. A commitment definition with this scope value is referred to as the **job-level** (*JOB) commitment definition. Any program running in an activation group that does not have an activation-group-level commitment definition started uses the job-level commitment definition, if it has already been started by another program for the job. Only a single job-level commitment definition can be started for a job.

In addition to the job-level commitment definition and commitment definitions that are scoped to a particular activation group, a job can have other commitment definitions active that are started by some system functions. These types of commitment definitions do the following:

- Function independently from any other commitment definition active for a job.
- Do not have any particular scope.
- Are used only by the system function that started the commitment definition.

An example of a function that starts these types of commitment definitions is OfficeVision/400.

For a given activation group, only a single commitment definition can be used by the programs that run within that activation group. Therefore, programs that run within an activation group can either use the job-level or the activation-group-level commitment definition, *but not both at the same time.*

Even when the job-level commitment definition is active for the job, a program can still start the activation-group-level commitment definition if no commitment control requests or operations have been performed for the job-level commitment definition by any program running within that activation group. Otherwise, the job-level commitment definition would first have to be ended before the activation-group-level commitment definition could be started. Commitment control requests or operations for the job-level commitment definition that would prevent the activation-group-level commitment definition from being started include:

- An open, full or shared, of a database file under commitment control.
- Adding an API commitment resource using the QTNADDCR API.
- A commit operation.
- A rollback operation.

Likewise, if the programs within an activation group are currently using the activation-group-level commitment definition, it must first be ended before the job-level commitment definition could be used by the programs running within that same activation group.

When opening a database file, the open scope for the opened file may be either to the activation group or to the job with one restriction. If the file is being opened under commitment control and is being scoped to the job, then the program making the open request must be using the job-level commitment definition.

Commitment Definition Names: The system gives names to all commitment definitions started for a job. Table 4-1 on page 4-4 shows various commitment definitions and their associated names for a particular job.

Table 4-1. Commitment Definition Names for a Job

Activation Group	Commit Scope	Commitment Definition Name
<any>	Job	*JOB
Default activation group	Activation group	*DFTACTGRP
User-named activation group	Activation group	Activation group name (for example, PAYROLL)
System-named activation group	Activation group	Activation group number (for example. 0000000145)
<none>	<none>	QDIR001 (example of a system-defined commitment definition for system use only)

Note

Only Integrated Language Environment (ILE)-compiled programs can start commitment control for activation groups other than the default activation group. Therefore, a job can use multiple commitment definitions only if the job is running one or more ILE-compiled programs. For more information about the Integrated Language Environment, see the *ILE* C/400 Programmer's Guide and Reference*.

Original Program Model (OPM) programs run in the default activation group, and by default use the *DFTACTGRP commitment definition. In a mixed OPM and ILE environment, the job-level commitment definition should be used if all committable changes made by all programs are to be committed or rolled back together.

An opened database file scoped to an activation group can be associated with either an activation-group-level or job-level commitment definition. An opened database file scoped to the job can be associated only with the job-level commitment definition. Therefore, any program, OPM or ILE, that opens a database file under commitment control scoped to the job needs to use the job-level commitment definition.

Application programs *do not* use the commitment definition name to identify a particular commitment definition when making a commitment control request. Commitment definition names are primarily used in messages to identify a particular commitment definition for a job. Instead, the system determines which commitment definition to use based on which activation group the requesting program is running in. This is possible because only a single commitment definition can be used by the programs that run within an activation group at any point in time.

Jobs with Multiple Commitment Definitions Example: Figure 4-1 on page 4-5 shows an example of a job that uses multiple commitment definitions.

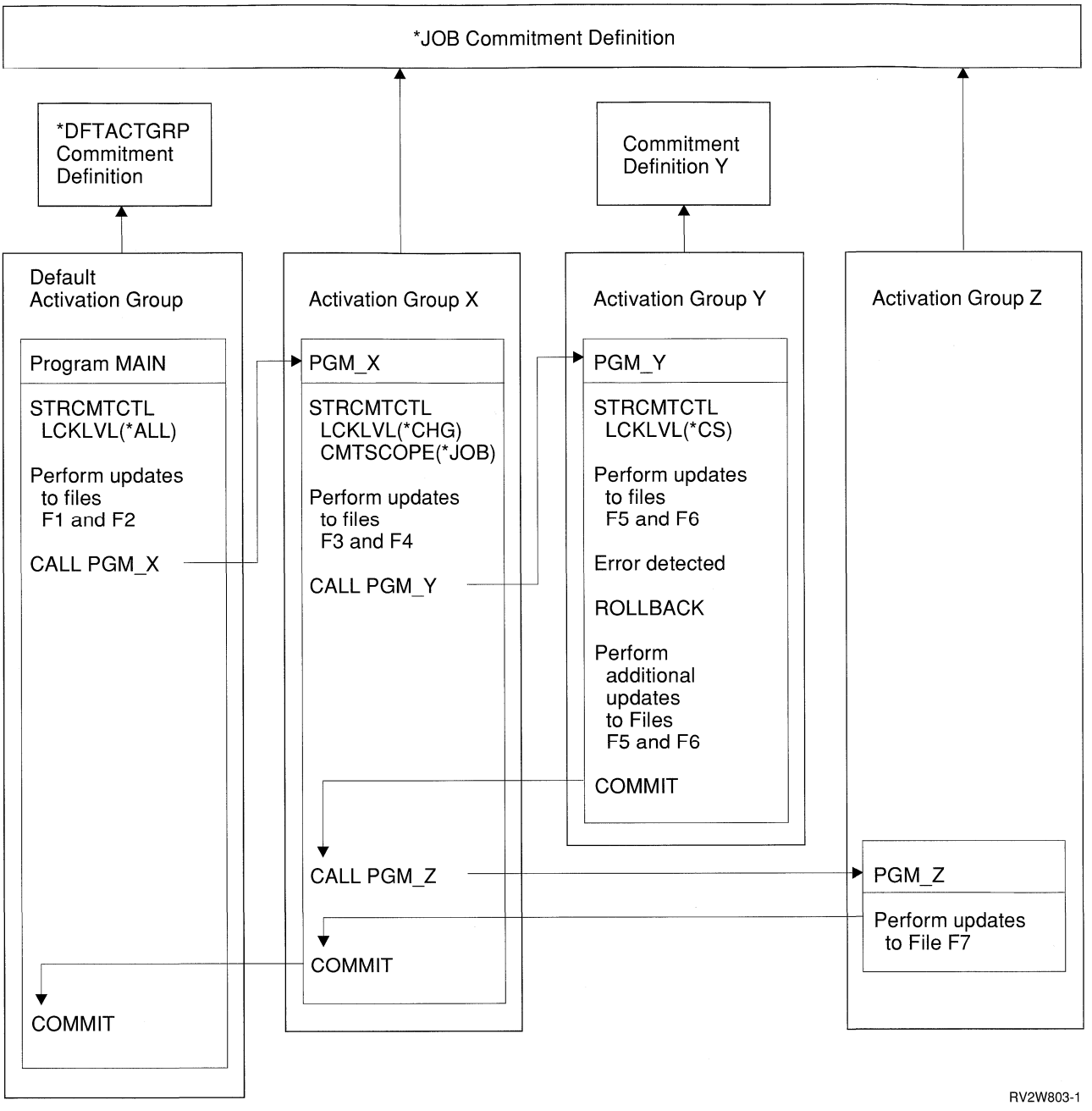


Figure 4-1. Using multiple commitment definitions in a job

The example assumes that all of the updates made to the database files by all of the programs are made under commitment control.

1. The MAIN program starts the *DFTACTGRP commitment definition and performs updates to files *F1* and *F2*. The updates to files *F1* and *F2* are associated with the *DFTACTGRP commitment definition.
2. After the updates are complete, but still pending:
   a. PGM_X is called that runs in activation group X.
   b. PGM_X starts the job-level commitment definition and makes updates to files *F3* and *F4*.
   c. The updates to files *F3* and *F4* are associated with the *JOB commitment definition.
3. The updates to files *F3* and *F4* are still pending when PGM_Y is called, which runs in activation group Y:

a. PGM_Y starts the activation-group-level commitment definition named Y.

b. Updates are made to files *F5* and *F6*.

However, the program then detects an error condition and performs a rollback operation. The rollback operation is performed only for commitment definition Y. Therefore, only the updates that were made to files *F5* and *F6* are backed out.

c. PGM_Y then makes the correct updates to files *F5* and *F6* and performs a commit operation.

As with the previous rollback operation, the commit operation is performed for commitment definition Y. Therefore, only the updates to files *F5* and *F6* are committed to the database.

d. At this point in time, the updates to files *F1, F2, F3,* and *F4* are still pending.

4. Control is then returned to PGM_X, which then:

a. Calls PGM_Z running in activation group Z. PGM_Z does not start commitment control.

b. The updates made to file *F7* are associated with the *JOB commitment definition because it is active.

5. Control then returns to PGM_X, which performs a commit operation.

The commit operation is performed for the job-level commitment definition which commits the updates made to files *F3*, *F4*, and *F7* to the database.

The updates made to files *F1* and *F2* are still pending.

6. Control then returns to the MAIN program, which performs a commit operation that makes the changes to files *F1* and *F2* in the database permanent.

Notes:

1. If the commit operation performed by PGM_X is changed to a rollback operation, then only the updates to files *F3*, *F4*, and *F7* are backed out from the database. The updates to files *F5* and *F6* remain in the database, and the updates for files *F1* and *F2* remain pending.

2. If a commit operation is performed by PGM_Z after the updates to file *F7*, then that commit operation commits all of the updates for files *F3*, *F4*, and *F7* to the database. When control returns to PGM_X, the updates to files *F3* and *F4* are already committed to the database.

3. If PGM_X is not making its changes under commitment control, and thus, does not start the job-level commitment definition, an error is signaled to PGM_Z when it attempts to make its changes under commitment control without first running a STRCMTCTL command. Programs running within an activation group can make changes under commitment control *without* first running a STRCMTCTL command only if one of the following is true:

   - Some other program running within the job has started the job-level commitment definition.
   - A program running within the same activation group has started the activation-group-level commitment definition.

4. PGM_Z can run a STRCMTCTL command to start the activation-group-level commitment definition for activation group Z instead of using the job-level commitment definition if it is desirable to have the updates for file F7 committed independently from the updates made for files F3 and F4. However, notice that the activation-group-level commitment definition must be started *before* making any changes under commitment control that become associated with the job-level commitment definition.

5. The default level of record locking (LCKLVL parameter) for database files opened under commitment control is specified when starting commitment control with the STRCMTCTL command. Notice that all of the commitment definitions are using different levels of default record locking. Any valid level of default record locking can be specified for a particular commitment definition, regardless of the default record locking value specified for any other commitment definition being used in the job.

   In the example, the updates performed by PGM_Z are using a default lock level of *CHG. This was established when the job-level commitment definition was started by PGM_X. For more information about lock

levels, see the topic "Lock-Level Parameter" on page 4-11.

6. For simplicity, the example shows only a single program running within each activation group. However, all programs that run in an activation group all use the same single commitment definition.
7. This example uses only local database files as the committable resources. However, different types of committable resources could be:
   a. Used and substituted for the database file resources.
   b. Added in addition to the database file resources already in the example.

Implicit Commit and Rollback Operations

Usually, a commit or rollback operation is initiated from an application program using one of the available programming languages that supports commitment control. These types of commit and rollback operations are known as ***explicit commit and rollback requests.*** However, in some instances the system initiates a commit or rollback operation for a commitment definition. Commit and rollback operations initiated by the system are known as ***implicit commit and rollback requests.***

The system performs an implicit commit or rollback for a commitment definition for the following cases:

- Activation group end

  Pending changes are implicitly committed by the system:

  - If pending changes exist for an activation-group-level commitment definition associated with an activation group that is ending normally, and
  - If no errors are encountered by the system when closing any files opened under commitment control scoped to the activation group during the activation group end processing.

  Pending changes are implicitly rolled back by the system:

  - If pending changes exist for an activation-group-level commitment definition associated with an activation group that is ending abnormally, or
  - If errors were encountered by the system when closing any file opened under commitment control scoped to the activation group during the activation group end processing.

  However, an implicit commit or rollback operation is *never* performed during activation group end processing for the *JOB or *DFTACTGRP commitment definitions. This is because the *DFTACTGRP and *JOB commitment definitions are never ended due to an activation group ending. These commitment definitions are explicitly ended by the ENDCMTCTL command or when the job ends.

- Abnormal job end

  If pending changes exist for any commitment definition when a job ends abnormally, those pending changes are rolled back implicitly by the system. However, if a job ends abnormally during a user's explicitly requested commit operation for a commitment definition, then the commit operation is completed before ending the job.

- Normal job end

  If pending changes exist for any commitment definition when a job ends normally, those pending changes are rolled back implicitly by the system.

- Abnormal system end

  If pending changes exist for any commitment definition for any job when the system ends abnormally, those pending changes are rolled back implicitly by the system as part of the IPL recovery processing during the next IPL. However, if the system ends abnormally during a user's explicitly requested commit operation for a commitment definition, then the commit operation is completed during the next IPL.

Note: With respect to performing an implicit commit or rollback operation, a read performed for a database file opened under commitment control constitutes a pending change for that particular commitment definition. This is because the file position for each file opened under commitment control is brought back to the last commitment boundary when a rollback operation is performed.

Performing a read under commitment control changes the file position, and, therefore, a pending change then exists for the commitment definition. A commitment definition with an API commitment resource added is always considered to have pending changes. This is because the system does not know when a real change is made for an object or objects associated with an API commitment resource.

The flag value for the 'CM' (changes committed) and 'RB' (changes rolled back) journal entries indicate whether a commit or rollback operation for a particular commitment transaction was performed explicitly by an end-user program or implicitly by the system. See Table 2-5 on page 2-19 for more information about the content for these journal entry types.

Files Being Journaled under Commitment Control

The system requires that a database file be journaled if the record changes to that file are to be made under commitment control. The following items apply to files being journaled for commitment control purposes:

- An error message is sent by the system at open time if an attempt is made to open a database file under commitment control, but the file is not currently journaled. Likewise, an error message is also sent if no commitment definition is started that can be used by the file being opened.
- If the database file resides on a remote system, then that file must be journaled to a journal on the remote system prior to the open being performed for the remote file.
- All database files that are to be opened under commitment control at the same time for a commitment definition for a particular commitment transaction must be journaled to the *same, single* journal.

  However, a different journal can be used from transaction to transaction for a particular commitment definition. A file can be opened under commitment control that is journaled to a different journal than the previous open if:

  – No file opened under commitment control currently exists for this particular commitment definition that is journaled to a different journal, and

  – No closed files exist for this particular commitment definition with pending changes that are journaled to a different journal.

- If only the after images are being journaled for a database file when that file is opened under commitment control, the system automatically starts journaling both the before and after images. The before record images will be captured on the journal only for those programs that are making record changes to the file under commitment control. The record changes *not* made under commitment control for the same file (most likely being made by other programs) still have only the after images captured on the journal for those record changes.

Only record-level committable changes are automatically written to a journal by the system. The journal is then used by the system, if necessary, for recovery purposes. Object-level committable changes and committable changes for API commitment resources are *not* written to a journal. Recovery for object-level commitment resources is performed by the system using a mechanism other than a journal. Recovery for API commitment resources is accomplished by calling the commit and rollback exit program associated with each particular API commitment resource.

Commit Cycle Identifier: The ***commit cycle identifier*** within each journal entry is used to associate all of the journal entries for a particular commitment transaction together. The 'SC' (commit transaction started) journal entry type denotes the beginning of a commitment transaction. The 'CM' (changes committed) and 'RB' (changes rolled back) journal entries denote the end of a commitment transaction. The 'SC' journal entry, the 'CM' or 'RB' journal entry, and all of the other journal entries deposited in the journal on behalf of record-level changes for the same commitment transaction are all identified with the same unique commit cycle identifier. The commit cycle identifier for a particular commitment transaction is the journal sequence number of the 'SC' (commit transaction started) journal entry. For a complete list of journal entry types that can be

associated with a commitment transaction, see "Entry Types by Journal Code" on page 2-19.

NOTE

The journal entries that are written to a journal on behalf of commitment control requests are written only if at least one local database file has ever been opened under commitment control for the commitment definition. If no local database files have ever been opened under commitment control for the commitment definition, then the commitment transactions are not recorded in any journal on the local system.

If at least one remote database file is opened under commitment control for the commitment definition, then the commitment transactions are recorded in a journal on the remote system.

For data recovery purposes, the Apply Journaled Changes (APYJRNCHG) command can be used to apply record changes to database files. When performing an apply operation, CMTBDY(*YES) can be specified so that commitment boundaries are enforced when the system applies the record-level changes to the database files. The same is also true for the Remove Journaled Changes (RMVJRNCHG) command when removing record-level changes from database files. For database record changes made to remote files, the apply and remove commands must be issued on the remote system.

However, because object-level changes made under commitment control and committable changes made to objects using the QTNADDCR API are *not* written to a journal, those committable changes cannot be applied or removed using the APYJRNCHG and RMVJRNCHG commands. Commitment transactions involving these types of committable changes must be recovered with user-written programs.

See Chapter 2, "Journal Management" on page 2-1 for more information about the journal management functions.

Changes Made to Resources under Commitment Control

Various resources can be placed under commitment control. Depending upon the resource, different kinds of committable changes can be made for those resources. The following discusses these different commitment resources and how changes are made to them under commitment control.

Local and Remote AS/400 Database Files:
Specification is made at the time a database file is opened indicating whether the changes are to be made under commitment control. For details on how to place a database file under commitment control, refer to the appropriate language reference manual.

All database files that are opened under commitment control during a commitment transaction must be journaled to the same journal. The journal to be used by an associated commitment definition is established at the time of the first database file open made under commitment control that is to be associated with the commitment definition. If the database file resides on a remote system, then that file must be journaled to a journal on the remote system prior to the remote file being opened.

Note: If all database files that were opened under commitment control to a particular commitment definition are closed and a commitment boundary is established, then the next file to be opened under commitment control *may* be journaled to a journal other than the one previously used.

After the database file has been opened under commitment control, record-level changes made to the file are recorded in the journal, along with the commit and rollback operations that denote the end of each individual commitment transaction. After the record-level changes are complete, the database file can be closed. The file may be closed with or without pending changes remaining for the file. If pending changes remain for the file at the time it is closed, those pending changes are either committed or rolled back during the next commit or rollback operation performed for the commitment definition.

No more than 256 distinct database file members can be opened under commitment control, or closed with pending changes, associated with the same commitment definition. The maximum number of records that can be locked in a transaction is 32768.

Local or remote record-level changes can be made to database files or tables using DDM or SQL. However, commitment control for DDM files is supported only between AS/400 systems. For information on how to place DDM files under commitment control, see the *DDM Guide*. For information on how to place database files accessed by local and remote SQL under commitment control, see the *Distributed Database Guide*.

Local and Remote Objects Accessed by SQL: Local and remote object-level changes can be made using SQL. This includes Data Definition Language (DDL) changes such as Create SQL Package, Create SQL Table, and Drop SQL Table. For a complete list of possible object-level changes and more information on how to place object-level changes under commitment control using local and remote SQL, see the *Distributed Database Guide*.

Object-level changes to local database files and DDM files not accessed using SQL are *not* committable changes.

Local API Commitment Resources: Applications may place other local objects under commitment control using the Add Commitment Resource (QTNADDCR) API. The application must provide an exit program (referred to as the commit and rollback exit program) to perform the commit and rollback operations for these resources. The exit program is called during commit and rollback operations to perform whatever processing is necessary for the object, or objects, that are associated with the API commitment resource. More than one API commitment resource may be added to a commitment definition. API commitment resources are always considered local resources.

Table 4-2 shows the three application programming interfaces provided by the system for commitment control purposes.

Table 4-2. Commitment Control Application Programming Interfaces

API Program	API Name	Description
QTNADDCR	Add Commitment Resource	Adds an API commitment resource to a particular commitment definition.
QTNRMVCR	Remove Commitment Resource	Removes an API commitment resource from a particular commitment definition.
QTNRCMTI	Retrieve Commitment Information	Retrieves information about commitment control.

For more information about API commitment resources and these API interfaces, see the *System Programmer's Interface Reference*.

Start Commitment Control Command

Commitment control is started either at the activation-group level or at the job-level using the Start Commitment Control (STRCMTCTL) command.

The following can be specified on the Start Commitment Control (STRCMTCTL) command:

- The default level of record locking (LCKLVL parameter) for the database files opened under commitment control. For more information about lock levels, see the topic "Lock-Level Parameter" on page 4-11.
- The notify object (NFYOBJ) parameter to which an entry (the commit identification) is sent to identify the last transaction that was successfully committed for the commitment definition. The system places the commit identification in the notify object for a commitment definition used for local resources if:
  - an abnormal job end or an abnormal system end occurs.
  - pending changes exist during a normal job end.

- a rollback operation is performed during an activation group end.

For more information about notify objects, see the topic "Notify Object Parameter" on page 4-13.

- The scope (CMTSCOPE parameter) for the commitment definition.
- The Text (TEXT parameter) to help describe the intended use of the commitment definition.

Lock-Level Parameter: The value you specify for the LCKLVL parameter on the Start Commitment Control (STRCMTCTL) command becomes the default level of record locking for database files that are opened and placed under commitment control for the commitment definition. The default level of record locking cannot be overridden when opening local database files. However, database files accessed by SQL can be opened with a different level of record locking than what is specified as the default lock level on the STRCMTCTL command.

The lock level should be specified according to your needs, the wait periods allowed, and the release procedures used most often. The following descriptions apply only to files opened under commitment control.

***CHG:** This value is used if you want only changed records to be protected from changes by other jobs running at the same time.

When updating records using files not opened under commitment control, the lock on the record is held only from the time the record is read until the update operation is complete. However, under commitment control, the lock is held for the duration of the transaction.

***CS:** This value is used if you want both changed and retrieved records to be protected from changes by other jobs running at the same time. Retrieved records that are not changed are protected only until they are released, or a different record is retrieved.

This value ensures that other jobs are not able to read a record for update that has been read under commitment control by this job. In addition, the program cannot read records for update that have been locked with a record lock type of *UPDATE in another job until that job accesses a different record.

***ALL:** This value is used if you want changed records and retrieved records that are under commitment control to be protected from changes by other jobs running under commitment control at the same time. Records that are retrieved or changed are protected until the next commit or rollback operation.

This value ensures that other jobs are not able to access a record for update that has been read under commitment control by this job. This is different from normal locking protocol. When the lock level is specified as *ALL, even a record that is not read for update cannot be accessed if it is locked with a record lock type of *UPDATE in another job.

Table 4-3 shows the duration of record locks for files under and not under commitment control.

Table 4-3 (Page 1 of 2). Lock Duration by Lock-Level Parameter

Request	LCKLVL Parameter	Duration of Lock	Lock Type
Read only	No commitment control	No lock	
	LCKLVL(*CHG)	No lock	
	LCKLVL(*CS)	From read to next read, commit, or rollback	*READ
	LCKLVL(*ALL)	From read to commit or rollback	*READ

Table 4-3 (Page 2 of 2). Lock Duration by Lock-Level Parameter

Request	LCKLVL Parameter	Duration of Lock	Lock Type
Read for update then update or delete	No commitment control	From read to update or delete	*UPDATE
	LCKLVL(*CHG)	From read to update or delete From update or delete to next commit or rollback	*UPDATE *UPDATE
	LCKLVL(*CS)	From read to update or delete From update or delete to next commit or rollback	*UPDATE *UPDATE
	LCKLVL(*ALL)	From read to update or delete From update or delete to next commit or rollback	*UPDATE *UPDATE
Read for update then release	No commitment control	From read to release	*UPDATE
	LCKLVL(*CHG)	From read to release, commit, or rollback	*UPDATE
	LCKLVL(*CS)	From read to release, commit, or rollback	*UPDATE
	LCKLVL(*ALL)	From read to release, commit or rollback	*UPDATE
Add	No commitment control	No lock	
	LCKLVL(*CHG)	From add to commit or rollback	*UPDATE
	LCKLVL(*CS)	From add to commit or rollback	*UPDATE
	LCKLVL(*ALL)	From add to commit or rollback	*UPDATE
Write direct	No commitment control	For duration of write direct	*UPDATE
	LCKLVL(*CHG)	From write direct to commit or rollback	*UPDATE
	LCKLVL(*CS)	From write direct to commit or rollback	*UPDATE
	LCKLVL(*ALL)	From write direct to commit or rollback	*UPDATE

A record lock type of *READ is obtained on records that are not read for update when the lock level is *CS or *ALL. This type of lock prevents other jobs from reading the records for update but does not prevent the records from being accessed from a read-only operation.

A record lock type of *UPDATE is obtained on records that are updated, deleted, added, or read for update. This type of lock prevents other jobs from reading the records for update, and prevents jobs running under commitment control with a record lock level of *CS or *ALL from accessing the records for even a read-only operation.

Programs that are not using commitment control can read records locked by another job, but cannot read records for update, regardless of the value specified for the LCKLVL parameter.

The lock level specified for a commitment definition when commitment control is started for an activation group or for the job applies only to opens associated with that particular commitment definition.

Note: The *CS and *ALL lock-level values protect you from retrieving a record that currently has a pending change from a different job. However, the *CS and *ALL lock-level values *do not* protect you from retrieving a record using a program running in one activation group that currently has a pending change from a program running in a different activation group within the same job.

Within the same job, a program can change a record that has already been changed within the current transaction as long as the record is accessed again using the same commitment definition. When using the job-level commitment definition, the access to the changed record can be made from a program running within any activation group that is using the job-level commitment definition.

Notify Object Parameter

A **notify object** is a message queue, data area, or database file that contains information identifying the last successful transaction completed for a particular commitment definition if that commitment definition did not end normally. The information used to identify the last successful transaction for a commitment definition is given by the **commit identification** that associates a commit operation with a specific set of committable resource changes.

The commit identification of the last successful transaction for a commitment definition is placed in the notify object only if the commitment definition does not end normally. This information can be used to help determine where processing for an application ended so that the application can be started again.

If local database files are under commitment control and a commit operation is performed with a commit identification, the commit identification is placed in the commit journal entry (journal entry type of 'CM') that identifies that particular transaction as being committed.

You can specify the commit identification on the following:

- CL - COMMIT command
- RPG/400 - COMIT operation code
- PL/I - PLICOMMIT subroutine
- C/400 - _Rcommit function

The commit identification is not supported on the COBOL COMMIT verb or the SQL COMMIT statement.

The maximum size of the commit identification within the notify object for the RPG/400 COMIT operation code, the C/400 _Rcommit function, and the PLICOMMIT subroutine is 4000 characters. If the commit identification is larger than 4000 characters in length, the commit identification is truncated. The maximum size of the commit identification within the notify object for the CL COMMIT command is 3000 characters. If the commit identification is larger than 3000 characters, the commit identification is truncated.

Note: Data areas are limited to 2000 characters.

When a notify object is updated with the commit identification, it is updated as follows:

- Database file

  If a database file is used as the notify object, the commit identification is added to the end of the file. Any existing records are left in the file. Because several users or jobs can be changing records at the same time, each commit identification in the file should contain unique information to associate the data with the job and commitment definition that failed. The file serving as a notify object can be journaled.

- Data area

  If a data area is used as the notify object, the entire content of the data area is replaced when the commit identification is placed in the data area. If more than one user or job is using the same program, only the commit identification from the last commitment definition that did not end normally will be in the data area. Consequently, a single data area

notify object may not produce the correct information for starting the application programs again. To solve this problem, use a separate data area for each commitment definition for each workstation user or job.

- Message queue

  If a message queue is used as a notify object, message CPI8399 is sent to the message queue. The commit identification is placed in the second-level text for message CPI8399. As with using a database file for the notify object, the contents of each commit identification should uniquely identify a particular commitment definition for a job so that an application program can be started again.

Updates Made to the Notify Object:

Notify objects are useful only for the recovery of local resources. Regardless of how a commitment definition ends, the system does *not* update the notify object if remote resources have pending changes for the transaction for which the interruption occurred. The notify object is *not* useful for recovering remote resources.

With respect to updating the notify object, a read performed for a database file opened under commitment control constitutes a pending change for that particular commitment definition. This is because file position is brought back to the last commitment boundary when a rollback operation is performed. Performing a read under commitment control changes the file position, and therefore, a pending change then exists for the commitment definition.

A commitment definition with an API commitment resource added is always considered to have pending changes. This is because the system does not know when a real change is made for an object or objects associated with an API commitment resource.

Otherwise, updates to the notify object are made by the system based upon the following ways a commitment definition can end:

- If a job ends normally and no uncommitted changes exist, the system does *not* place the commit identification of the last successful commit operation in the notify object.
- If an implicit commit operation is performed for an activation-group-level commitment definition when the activation group is ended, the system does *not* place the commit identification of the last successful commit operation in the notify object.

  Note: Implicit commit operations are *never* performed for the *DFTACTGRP or *JOB commitment definitions.
- If the system, job, or an activation group ends abnormally before the first successful commit operation for a commitment definition, the system does *not* update the notify object because there is no last commit identification. To differentiate between this condition and a normal program completion, your program must update the notify object with a specific entry prior to completing the first successful commit operation for the commitment definition.
- If an abnormal job end or an abnormal system end occurs after at least one successful commit operation, the system places the commit identification of that commit operation in the notify object. If the last successful commit operation did not specify a commit identification, then the notify object is not updated. For an abnormal job end, this notify object processing is performed for each commitment definition that was active for the job. For an abnormal system end, this notify object processing is performed for each commitment definition that was active for all jobs on the system.
- The system updates the notify object with the commit identification of the last successful commit operation for that commitment definition if all of the following occur:
  - A non-default activation group ends.
  - An implicit rollback operation is performed for the activation-group-level commitment definition.
  - At least one successful commit operation has been performed for that commitment definition.

  If the last successful commit operation did not specify a commit identification, then the notify object is not updated. An implicit rollback operation is performed for an activation-group-level commitment definition if the activation group is ending abnormally or errors occurred when closing the files opened under commit-

ment control that were scoped to that activation group. For more information about scoping database files to activation groups and how activation groups can be ended, see the *ILE* C/400 Reference Summary*.

- If uncommitted changes exist when a job ends normally and at least one successful commit operation has been performed, the commit identification of the last successful commit operation is placed in the notify object and the uncommitted changes are rolled back. If the last successful commit operation did not specify a commit identification, then the notify object is not updated. This notify object processing is performed for each commitment definition that was active for the job when the job ended. For more information on the functions done during the normal ending of a job, see the topic "Commitment Control during Normal Routing Step End" on page 4-20.
- If uncommitted changes exist when the ENDCMTCTL command is run where stated, the notify object is updated only if the last successful commit operation specified a commit identification:
  - For a batch job, the uncommitted changes are rolled back and the commit identification of the last successful commit operation is placed in the notify object.
  - For an interactive job, if the response to inquiry message CPA8350 is to rollback the changes, the uncommitted changes are rolled back and the commit identification of the last successful commit operation is placed in the notify object.
  - For an interactive job, if the response to inquiry message CPA8350 is to commit the changes, the system prompts for a commit identification to use and the changes are committed. The commit identification that is entered on the prompt display is placed in the notify object.
  - For an interactive job, if the response to inquiry message CPA8350 is to cancel the ENDCMTCTL request, the pending changes remain and the notify object is *not* updated.

Commit Scope Parameter

The commit scope parameter identifies the scope for the commitment definition. The default is to scope the commitment definition to the activation group of the program making the start commitment control request. The alternative scope is to the job. See "Commitment Definitions and Activation Groups" on page 4-2 for more information regarding the scope for a commitment definition.

Commit Text Parameter

The commit text parameter identifies the specific text to be associated with a commitment definition when displaying information about the commitment definitions started for a job. If no text is specified, the system provides a default text description.

Commit Operations

A commit operation performed for a commitment definition makes the committable changes that have been made since the last commit or rollback operation for the same commitment definition permanent. A single commit operation makes permanent the committable resource changes for a single commitment definition within the job. The system determines the commitment definition for which to perform the commit operation based on the activation group in which the program making the commit request is running.

Use the following to perform a commit operation:

- CL - COMMIT command
- RPG/400 - COMIT operation code
- COBOL/400 - COMMIT verb
- C/400 - _Rcommit function
- ILE C/400 - _Rcommit function
- PL/I - PLICOMMIT subroutine
- SQL/400 - COMMIT statement

When a successful commit operation runs, the system does the following:

- If local resources are under commitment control, the system saves the commit identification, if one is provided, for use at recovery time.
- Records are written to the file prior to the commit operation being performed, if:

- Records were added to a local or remote database file under commitment control, and
- SEQONLY(*YES) was specified when the file was opened so that blocked I/O feedback is used by the system and a partial block of records exists.

Otherwise, the I/O feedback area and I/O buffers are not changed.

- If a file has been opened since the commitment definition was established, a journal entry type of 'CM' is added to the journal to indicate that changes made under commitment control have been committed. The commit identification, if one is specified, is used as the entry-specific data for the 'CM' journal entry. Before the commit operation was requested, the before-images and the after-images of changed records were placed in the journal.
- Pending object-level changes are made permanent.
- Record and object locks that were acquired and kept for commitment control purposes are unlocked and those resources are made available to other users.
- A call is made to the commit and rollback exit program for each API commitment resource that is present in the commitment definition. The exit programs are called in the order that the API commitment resources were added to the commitment definition.
- Information is changed in the commitment definition to show that the current transaction has been ended.

All of the above must perform correctly for the commit operation to be successful.

When remote resources are under commitment control, a commit operation on the source system causes the system to send a commit request to the target system. At the target system, the commit operation works the same as it would on a local system. Any errors are sent back to the source system where they are signaled to the user. If the commit operation fails for remote resources and pending changes have been made to remote resources using SQL, the system sends a rollback request to the target system and the connection is ended.

Rollback Operations

A rollback operation performed for a commitment definition removes the committable changes that have been made since the last commit or rollback operation for the same commitment definition. A single rollback operation removes the committable resource changes for a single commitment definition within the job. The system determines the commitment definition on which to perform a rollback operation based on the activation group in which the program making the rollback request is running.

Use the following to perform the rollback operation:

- CL - ROLLBACK command
- RPG/400 - ROLBK operation code
- COBOL/400 - ROLLBACK verb
- C/400 - _rRollback function
- ILE C/400 - _Rrollback function
- PL/I - PLIROLLBACK subroutine
- SQL/400 - ROLLBACK statement

When a successful rollback operation runs, the system does the following:

- For database files under commitment control:
  - If a record was deleted from a file, the record is added back to the file.
  - Any changes to records that have been made during this transaction are removed, and the original records (the before-images) are placed back into the file.
  - If any records were added to the file during this transaction, they remain in the file as deleted records.
  - If any record changes were made during the transaction, a journal entry of 'RB' is added to the journal indicating that a rollback operation occurred. The journal also contains images of the record changes that were rolled back. Before the rollback operation was requested, the before-images and after-images of changed records were placed in the journal.
  - The system positions the open files under commitment control at the last record accessed in the previous transaction or at

the open position if no commit operation has been performed for the file using this commitment definition. This is an important consideration if you are doing sequential processing.

- Records are cleared from the I/O buffer:
  - If records were added to a local or remote database file under commitment control, and
  - If SEQONLY(*YES) was specified when the file was opened so that blocked I/O is used by the system and a partial block of records exists that has not yet been written to the database.

  Otherwise, the I/O feedback area and I/O buffers remain unchanged.
- Non-committable changes for database files are not rolled back. For example, opened files are not closed, and cleared files are not restored. Any files that were closed during this transaction are *not* reopened or repositioned.
- Record locks that were acquired for commitment control purposes are unlocked and those records are made available to other users.

- The commit identification currently saved by the system remains the same as the commit identification provided with the last commit operation for the same commitment definition.
- Object-level committable changes made during this transaction are reversed, or rolled back.
- Object locks that were acquired for commitment control purposes are unlocked and those objects are made available to other users.
- A call is made to the commit or rollback exit program for each API commitment resource that is present in the commitment definition. The exit programs are called in the *reverse* order of that by which the API commitment resources were added to the commitment definition.
- The previous commitment boundary is established as the current commitment boundary.
- Information is changed in the commitment definition to show that the current transaction has been ended.

All of the above must perform correctly for the rollback operation to be successful.

When remote resources are under commitment control, a rollback operation on the source system causes the system to send a rollback request to the target system. At the target system, the rollback operation works the same as it would on a local system. Any errors are sent back to the source system where they are signaled to the user. If the rollback operation fails, then the connection to the target system is ended.

Commitment Control Status

You can select an option on the Work with Job (WRKJOB command) or Display Job (DSPJOB command) display to show the status of commitment control for a particular job. When you select the *Display Commitment Control Status* option, a list of commitment definitions currently established for the job are shown. When you select the Display option for a commitment definition, the Display Commitment Definition Status display is shown. It contains the following information that can help in debugging a program.

- Activation group number associated with the commitment definition.

  This field is blank for the *JOB commitment definition and any commitment definition started by the system for system provided functions, such as OfficeVision/400.
- Journal to which the files under commitment control are being journaled.
- System-generated identifier of the current commit cycle, if local database files are opened under commitment control.
- Whether there are any resources under commitment control for the commitment definition, and if so, whether they are local or remote.
- Default lock-level specified on the STRCMTCTL command.
- Name of the notify object.
- Total number of commit operations performed for the commitment definition.

- Total number of rollback operations performed for the commitment definition.
- Whether API commitment resources have been added to the commitment definition.

If remote resources are not under commitment control for the commitment definition, you can select the *Display Record Level Status* option on the Display Commitment Control Status display. This display shows the following:

- Files currently under commitment control for the commitment definition.
- Total number of committed changes that were committed since the last time the file was opened and associated with this commitment definition.
- Total number of changes to the files that were rolled back since the last time the file was opened and associated with this commitment definition.
- Total number of committable record-level changes made to the files waiting for a commit or rollback operation.
- Level of record locking for the file.
- Status of the file (either open or closed).

If remote resources are not under commitment control for the commitment definition, you can select the *Display Object Level Status* option on the Display Commitment Control Status display. This display shows the following:

- Name and type of the object that has had an object-level change made for it during this commitment transaction.
- Description of the change that was made to the object.

If remote resources are under commitment control for the commitment definition, you can select the *Display Remote RDB Status* option on the Display Job Commitment Control Status display to show information about remote SQL relational databases. This display shows the following:

- Remote location name of the system containing the database.
- Name of the remote relational database under commitment control for this commitment definition.
- Whether or not changes have been made to the remote relational database during the current transaction.

If remote resources are under commitment control for the commitment definition, you can select the *Display Remote File Status* option on the Display Commitment Control Status display to show information about remote DDM files. This display shows the following:

- Remote location name of the system containing the DDM files currently opened under commitment control for this commitment definition.
- Names of the DDM files currently opened under commitment control for this commitment definition.
- Level of record locking for the file.
- Status of the file (open or closed).
- Remote file name for the DDM file.

For further description of the commitment control status displays, use the on-line help information.

On the Work with Active Jobs (WRKACTJOB) display, the *Status* column may show a value of CMTW (commit wait) for one or more jobs. This value indicates that the job is being prevented from moving beyond a commitment boundary due to a save-while-active operation. During a save-while-active operation, the system delays every job with committable resources in the save request at a commitment boundary until those objects have reached a checkpoint. This allows the save-while-active operation to save the objects at commitment boundaries so that no partial transactions are saved to the save media.

End Commitment Control (ENDCMTCTL) Command

Commitment control may be ended for either the job-level or activation-group-level commitment definition using the End Commitment Control (ENDCMTCTL) command. Issuing the ENDCMTCTL command indicates to the system that the commitment definition in use by the program making the request is to be ended. The ENDCMTCTL command ends only one commitment definition for the job and all other commitment definitions for the job remain unchanged.

If the activation-group-level commitment definition is ended, then programs running within that activation group can no longer make changes under commitment control, unless the job-level commitment definition is already started for the job. If the job-level commitment definition is active, then it is immediately made available for use by the programs running within the activation group that just ended commitment control.

If the job-level commitment definition is ended, then any program running within the job that was using the job-level commitment definition can no longer make changes under commitment control without first starting commitment control again with the STRCMTCTL command.

Before issuing the ENDCMTCTL command, the following must be satisfied for the commitment definition to be ended:

- All files opened under commitment control for the commitment definition to be ended must first be closed. When ending the job-level commitment definition, this includes all files opened under commitment control by any program running in any activation group that is using the job-level commitment definition.
- All API commitment resources for the commitment definition to be ended must first be removed using the QTNRMVCR API. When ending the job-level commitment definition, this includes all API commitment resources added by any program running in any activation group that is using the job-level commitment definition.
- A remote database associated with the commitment definition to be ended must be disconnected.

If commitment control is being ended in an interactive job and one or more committable resources associated with the commitment definition have pending changes, inquiry message CPA8350 is sent to the user asking whether to commit the pending changes, roll back the pending changes, or cancel the ENDCMTCTL request.

If commitment control is being ended in a batch job, and one or more closed files associated with the commitment definition to be ended still have pending changes, the changes are rolled back and message CPF8356 (for local resources) or CPF835C (for remote resources) is sent.

If a notify object is defined for the commitment definition being ended it may be updated. See "Updates Made to the Notify Object" on page 4-13 for more information regarding the updating of a notify object by the system.

After the commitment definition has successfully ended, all the necessary recovery, if any, has been performed. No additional recovery is performed for the commitment resources associated with the commitment definition just ended.

After the commitment definition is ended, the job-level or activation-group-level commitment definition can then be started again for the programs running within the activation group. The job-level commitment definition may be started only if it is not already started for the job.

Although commitment definitions can be started and ended many times by the programs that run within an activation group, the amount of system resources required for the repeated start and end operations can cause a decrease in job performance and overall system performance. Therefore, it is recommended that a commitment definition be left active if a program to be called later will use it.

Commitment Control during Activation Group End

The system automatically ends an activation-group-level commitment definition when an activation group ends and the job is not ending at the same time. If pending changes exist for an activation-group-level commitment definition and the activation group is ending normally, the system performs an implicit commit operation for the commitment definition before it is ended. Otherwise, an implicit rollback operation is performed for the activation-group-level commitment definition before being ended if the activation group is ending abnormally, or if errors were encountered by the system when closing any files opened under commitment control scoped to the activation group.

NOTE

An implicit commit or rollback operation is *never* performed during activation-group end processing for the *JOB or *DFTACTGRP commitment definitions. This is because the *JOB and *DFTACTGRP commitment definitions are never ended due to an activation group ending. Instead, these commitment definitions are either explicitly ended with an ENDCMTCTL command or ended by the system when the job ends.

The system automatically closes any files scoped to the activation group when the activation group ends. This includes any database files scoped to the activation group opened under commitment control. The close for any such file occurs before any implicit commit operation that may be performed for the activation-group-level commitment definition. Therefore, any records that reside in an I/O buffer are first forced to the database before any implicit commit operation is performed.

As part of the implicit commit or rollback operation that may be performed, a call is made to the API commit and rollback exit program for each API commitment resource associated with the activation-group-level commitment definition. After the API commit and rollback exit program is called, the system automatically removes the API commitment resource.

If an implicit rollback is performed for a commitment definition that is being ended due to an activation group being ended, then the notify object, if one is defined for the commitment definition, may be updated. See "Updates Made to the Notify Object" on page 4-14 for more information regarding the updating of a notify object by the system.

Commitment Control during Normal Routing Step End

The system ends *all* commitment definitions for a job when a routing step is normally ended. A routing step ends normally by one of the following:

- A normal end for a batch job.
- A normal sign-off for an interactive job.
- The Reroute Job (RRTJOB), Transfer Job (TFRJOB), or Transfer Batch Job (TFRBCHJOB) command ends the current routing step and starts a new routing step.

Any other end of a routing step is considered abnormal and is recognized by a nonzero completion code in the job completion message (CPF1164) in the job log.

Prior to ending a commitment definition during routing step end, the system performs an implicit rollback operation if the commitment definition has pending changes. This includes calling the API commit and rollback exit program for each API commitment resource associated with the commitment definition. After the API commit and rollback exit program is called, the system automatically removes the API commitment resource.

If a notify object is defined for the commitment definition, it may be updated. See "Updates Made to the Notify Object" on page 4-14 for more information about the updating of a notify object by the system.

Commitment Control during Abnormal System or Job End

The system ends *all* commitment definitions for a job when the job ends abnormally. These commitment definitions are ended during the end job processing. If the system ends abnormally, the system ends *all* commitment definitions that were started and being used by *all* active jobs at the time of the abnormal system end. These commitment definitions are ended as part of the database recovery processing that is performed during the next IPL after the abnormal system end.

The system performs the following for commitment definitions being ended during an abnormal job end or during the next IPL after an abnormal system end:

- Prior to ending a commitment definition, the system performs an implicit rollback operation if the commitment definition has pending changes, unless processing for the commitment definition was interrupted in the middle of a commit operation. If ended in the middle of a commit operation, the commit operation is completed. The processing to perform the implicit rollback operation or to complete the

commit operation includes calling the API commit and rollback exit program for each API commitment resource associated with the commitment definition. After the API commit and rollback exit program is called, the system automatically removes the API commitment resource.

- If a notify object is defined for the commitment definition, it may be updated. See "Updates Made to the Notify Object" on page 4-14 for more information about the system updating a notify object.

NOTE

The recovery for commitment definitions described in this section refers to an abnormal end for the system or a job due to a power failure, a hardware failure, or a failure in the operating system or licensed internal code. However, forcing a job to end abnormally with the End Job Abnormal (ENDJOBABN) command *may* result in pending changes for any currently active transactions for the job being ended to be partially committed or rolled back. The next IPL *may* attempt recovery for any partial transactions for the job ended with the ENDJOBABN command.

However, any commitment control recovery that may be performed during an IPL for a job that is ended with the ENDJOBABN command may or may not be desirable. All locks for commitment resources are released when the job is ended abnormally and any pending changes due to partial transactions are made available to other jobs. These pending changes may then cause other application programs to make additional erroneous changes to the database. Likewise, any ensuing IPL recovery that would be performed later may adversely effect the changes made by any applications after the job was ended abnormally. For example, an SQL table may be dropped during IPL recovery as the rollback action for a pending create table. However, other applications may have already inserted several rows into the table after the job was ended abnormally.

Commitment Control during a Save-While-Active Operation

The save-while-active checkpoint processing waits until all committable resources in the save request are at a commitment boundary (with respect to all jobs making committable changes to the objects being saved) prior to actually saving the objects to the save media. This is done so that no partial transactions are saved to the save media by a save-while-active operation.

When commitment control is active for any job on the system, the system performs the following during the save-while-active checkpoint processing:

- Identifies all jobs that have one or more commitment definitions with pending committable changes related to the objects being saved by the save-while-active operation.
- For identified jobs, allows additional committable changes to be made for *any* commitment definitions already started, or to be started. Additional committable changes are allowed for the objects so that all pending changes for the objects saved by the save-while-active operation can be committed or rolled back.
- Delays any job that attempts to make a committable change to an object being saved by the save-while-active operation and *all* commitment definitions for the job do not have any pending changes for any objects being saved by the save-while-active operation. The job is delayed only until the checkpoint processing is completed for the save-while active operation.

Any job not making committable changes to the objects being saved by the save-while-active operation is not delayed, except for the following cases:

- Object-level changes made to local resources using SQL.

  Any job that attempts to make an object-level committable change within a library that is having objects saved from it by a save-while-active operation.
- API commitment resources.

Any job that attempts to add an API commitment resource and all other commitment definitions for the job do not have any pending changes for any objects being saved by the save-while-active operation.

In addition, any job with an added API commitment resource is held during a commit or rollback request for the commitment definition with the API commitment resource added. This is because the system does not know which objects are being changed for an API commitment resource.

Therefore, for a save-while-active request, the system delays all jobs during the checkpoint processing that have a commitment definition with an API commitment resource. Because the job is held during the commit or rollback request, and because a commit or rollback request can be performed only for a single commitment definition at a time, *jobs with more than one commitment definition with API commitment resources added always prevent a save-while-active operation from completing.*

For all the cases mentioned previously that delay a job due to a save-while-active operation, the job is delayed only if all other commitment definitions for the job do not have any pending changes for any objects being saved by the save-while-active operation. The job is delayed only until the checkpoint processing is completed for the save-while-active operation.

The save active wait (SAVACTWAIT) parameter value on the save commands can be used to control the amount of time allowed for jobs to reach, and be delayed at, a commitment boundary.

Other considerations and restrictions apply when performing a save operation from a job that has active commitment definitions. For more information, see "Miscellaneous Considerations and Restrictions for Commitment Control" on page 4-26.

For more information about commitment control and save-while-active operations, see Chapter 5, "Save-While-Active Function" on page 5-1.

Commitment Control Considerations and Restrictions

The following sections describe programming considerations and restrictions when using commitment control.

The Size of a Transaction: For this discussion, a transaction is interactive. (Commitment control can also be used for batch applications, which often can be considered a series of transactions. Many of the same considerations apply to batch applications, which are discussed in "Commitment Control for Batch Applications" on page 4-25.)

The maximum number of records that can be locked in a transaction is 32 768.

When choosing the lock level for your records, consider the size of your transactions. Size should determine how long records are locked before a transaction ends. You have to decide if a commit or rollback operation for commitment control is limited to a single use of the Enter key, or if the transaction consists of many uses of the Enter key.

Note: The shorter the transaction, the earlier the job waiting to start save-while-active checkpoint processing can continue and complete.

For example, for an order entry application, a customer might order several items in a single order requiring an order detail record and an inventory master record update for every item in the order. If the transaction is defined as the entire order and each use of the Enter key orders an item, all records involved in the order are locked for the duration of the entire order. Therefore, often-used records (such as inventory master records) could be locked for long periods of time, preventing other work from progressing. If all items are entered with a single Enter key using a subfile, the duration of the locks for the entire order is minimized.

In general, the number and duration of locks should be minimized so several workstation users can access the same data without long waiting periods. You can do this by not holding locks while the user is entering data on the display. Some applications may not require more than one workstation user accessing the same data. For

example, in a cash posting application with many open item records per customer, the typical approach is to lock all the records and delay them until a workstation user completes posting the cash for a given receipt.

If the workstation user presses the Enter key several times for a transaction, it is possible to perform the transaction in a number of segments. For example:

- The first segment is an inquiry in which the workstation user requests the information.
- The second segment is a confirmation of the workstation user's intent to complete the entire transaction.
- The third segment is retrieval and update of the affected records.

This approach allows record locking to be restricted to a single use of Enter.

This inquiry-first approach is normally used in applications where a decision results from information displayed. For example, in an airline reservation application, a customer may want to know what flight times, connecting flights, and seating arrangements are available before making a decision on which flight to take. Once the customer makes a decision, the transaction is entered. If the transaction fails (the flight is now full), the rollback function can be used and a different request entered. If the records were locked from the first inquiry until a decision is made, another reservation clerk would be waiting until the other transaction is complete.

Record Locking: When a job holds a record lock and another job attempts to retrieve that record for update, the requesting job waits and is removed from active processing until one of the following occurs:

- The record lock is released.
- The specified wait time ends.

More than one job can request a record to be locked by another job. When the record lock is released, the first job to request the record receives that record. When waiting for a locked record, specify the wait time in the WAITRCD parameter on the following create, change, or override commands:

Create Physical File (CRTPF)
Create Logical File (CRTLF)
Create Source Physical File (CRTSRCPF)
Change Physical File (CHGPF)
Change Logical File (CHGLF)
Change Source Physical File (CHGSRCPF)
Override Database File (OVRDBF)

When specifying wait time, consider the following:

- If you do not specify a value, the program waits the default wait time for the process.
- If the record cannot be allocated within the specified time, a notify message is sent to the high-level language program.
- If the wait time for a record is exceeded, the message sent to the job log gives the name of the job holding the locked record that caused the requesting job to wait. If you experience record lock exceptions, you can use the job log to help determine which programs to alter so they will not hold locks for long durations. Programs keep record locks over long durations for one of the following reasons:
  - The record remains locked while the workstation user is considering a change.
  - The record lock is part of a long commitment transaction. Consider making smaller transactions so a commit operation can be performed more frequently.
  - An undesired lock has occurred. For example, assume a file is defined as an update file with unique keys, and the program updates and adds additional records to the file. If the workstation user wants to add a record to the file, the program may attempt to access the record to determine whether the key already exists. If it does, the program informs the workstation user that the request made is not valid. When the record is retrieved from the file, it is locked until it is implicitly released by another read operation to the same file, or until it is explicitly released.

    Note: For more information about how to use each high-level language interface to release record locks, see the appropriate high-level language reference manual.

    The duration of the lock is much longer if LCKLVL(*ALL) is specified because the record that was retrieved from the file is locked until the next commit or rollback

operation. It is not implicitly released by another read operation and cannot be explicitly released.

Minimizing Locks: A typical way to minimize record locks is to release the record lock. (This technique does not work if LCKLVL(*ALL) has been specified.) For example, a single file maintenance application typically does the following:

- Displays a prompt for a record identification to be changed.
- Retrieves the requested record.
- Displays the record.
- Allows the workstation user to make the change.
- Updates the record.

In most cases, the record is locked from the access of the requested record through the update. The record wait time may be exceeded for another job that is waiting for the record. To avoid locking a record while the workstation user is considering a change, release the record after it is retrieved from the database (before the record display appears). You then need to access the record again before updating. If the record was changed between the time it was released and the time it was accessed again, you should inform the workstation user. The program can determine if the record was changed by saving one or more fields of the original record and comparing them to the fields in the same record after it is retrieved, as follows:

- Use an update count field in the record and add 1 to the field just before an update. The program saves the original value and compares it to the value in the field when the record is retrieved again. If a change has occurred, the workstation user is informed and the record appears again. The update count field is changed only if an update occurs. The record is released while the workstation user is considering a change. If you use this technique, you must use it in every program that updates the file.
- Save the contents of the entire data record and compare it to the record the next time it is retrieved.

In both cases above, the sequence of operations prevents the simple use of externally described data in RPG where the same field names are used in the master record and in the display file. Using the same field names (in RPG) does not work because the workstation user's changes are overlaid when the record is retrieved again.

You can solve this problem by moving the record data to a data structure or continue to use externally described data if you use the DDS keyword RTNDTA. The RTNDTA keyword allows your program to reread data on the display without the operating system having to move data from the display to the program. This allows the program to do the following:

1. Prompt for the record identification.
2. Retrieve the requested record from the database.
3. Release the record.
4. Save the field or fields used to determine if the record was changed.
5. Display the record and wait for the workstation user to respond.

If the workstation user changes the record on the display, the program uses the following sequence:

1. Retrieves the record from the database again.
2. Compares the saved fields to determine if the database record has been changed. If it has been changed, the program releases the record and sends a message when the record appears.
3. Retrieves the record from the display by running a read operation with the RTNDTA keyword and updates the record in the database record.
4. Proceeds to the next logical prompt because there are no additional records to be released if the workstation user cancels the request.

LCKLVL(*CHG) and LCKLVL(*CS) work in this situation. If LCKLVL(*ALL) is used, you must release the record lock by using a commit or rollback operation.

Commitment Control for Batch Applications: Batch applications may or may not need commitment control. In some cases, a batch application can perform a single function of reading an input file and updating a master file. However, you can use commitment control for this type of application if it is important to start it again after an abnormal end.

The input file is an update file with a code in the records to indicate that a record was processed. This file and any files updated are placed under commitment control. When the code is present in the input file, it represents a completed transaction. The program reads through the input file and bypasses any records with the completed code. This allows the same program logic to be used for normal and starting again conditions.

If the batch application contains input records dependent on one another and contains switches or totals, a notify object can be used to provide information on starting again. The values held in the notify object are used to start processing again from the last committed transaction within the input file. See the topic "Notify Object Parameter" on page 4-13.

If input records are dependent on one another, they can be processed as a transaction. A batch job is not permitted to exceed the 32 768 record lock limit. Any commit cycle that exceeds 2000 locks will probably slow down system performance noticeably. Otherwise, the same locking considerations exist as for interactive applications, but the length of time records are locked in a batch application may be less important than in interactive applications.

Performance Considerations for Commitment Control: Using commitment control requires resources that can affect system performance. Several factors affect system performance regarding commitment control:

- Journaling

  Journaling a file requires system resources. If you specify only after-images, commitment control changes this to both before- and after-images while commitment control is in effect. Usually this is a space, not a performance, consideration.

- Commit operation

  If a file has been opened under commitment control, each commit of a transaction adds two entries to the journal whether or not the user has made changes to the database. The number of entries added can increase significantly for a large volume of small transactions. You may want to place the journal receivers in a user auxiliary storage pool (ASP). For detailed information on user ASPs, see "General Information about Auxiliary Storage Pools" on page 6-8.

- Rollback operation

  Because commitment control must rollback the pending changes recorded in the database, additional system resources are required whenever a rollback occurs. Also, if record changes are pending, a rollback operation causes additional entries to be added to the journal.

- Start Commitment Control (STRCMTCTL) and End Commitment Control (ENDCMTCTL) commands

  Additional overhead is incurred by the system each time a commitment definition is started and ended using the STRCMTCTL and ENDCMTCTL commands respectively. Avoid using the STRCMTCTL and ENDCMTCTL commands for each transaction. Use them only when necessary. You can establish a commitment definition at the beginning of an interactive job and use it for the duration of the job.

- Opening a file

  If you open a file without specifying the commit open option, no additional system resource is used even if a commitment definition has been started. For more information about specifying the commit open option, see the appropriate high-level language reference manual.

- Using the journal for commitment control transactions

  The same journal must be used for all files under commitment control within the same transaction for a commitment definition. Files that are journaled to use a different journal can be placed under commitment control without ending the commitment definition:

- If all the database files that were opened under commitment control for a particular commitment definition are now closed with no pending changes, or
- If all database files are now closed, and a commit or rollback operation is first performed to complete or back out any pending database file changes.

If files are open under commitment control, or a transaction has not been completed, a different journal cannot be used.

- Record locking

  Record locking can affect other applications. The number of records locked within a particular job increases the overall system resources used for the job. Applications needing to access the same record must wait for the transaction to end.

- SEQONLY

  If you request the SEQONLY(*YES) option (by using the OVRDBF command or the application program implicitly attempts to use SEQONLY(*YES)) and the file is opened for input only under commitment control, the option is changed to SEQONLY(*NO). This option can affect the performance of input files because records will not be blocked.

- Requesting a record-level change for a database file

  A request to make a record-level change under commitment control for a database file may be delayed if the commitment definition is at a commitment boundary and a save-while-active operation is running in a different job. This can happen when a file is journaled to the same journal as some of the objects on the save request.

  Note: The *Status* column on the Work with Active Jobs (WRKACTJOB command) display shows CMTW (commit wait) when a job is being held due to save-while-active checkpoint processing.

- Committing or rolling back changes.

  A commit or rollback operation may be delayed at a commitment boundary when a save-while-active operation is running in a different job. This can happen when an API commitment resource was previously added to the commitment definition.

- Requesting an object-level change

  A request to make an object-level change under commitment control may be delayed if the commitment definition is at a commitment boundary and a save-while-active operation is running in a different job. This can happen when an object-level change is made while the save-while-active operation is running against the library the object is in. For example, the create SQL table operation under commitment control for table MYTBL in library MYSQLLIB may be delayed when a save-while-active operation is running against library MYSQLLIB.

  Note: If the wait time exceeds 60 seconds, inquiry message CPA8351 is sent to ask the user whether to continue waiting or cancel the operation.

- Adding an API resource using the QTNADDCR API

  A request to add an API commitment resource using the QTNADDCR API may be delayed if all commitment definitions for the job are at a commitment boundary and a save-while-active operation is running in a different job.

  Note: If the wait time exceeds 60 seconds, inquiry message CPA8351 is sent to ask the user whether to continue waiting or cancel the operation.

Miscellaneous Considerations and Restrictions for Commitment Control

- If you specify that a shared file be opened under commitment control, all subsequent uses of that file must be opened under commitment control.
- If SEQONLY(*YES) is specified for the file opened for read only (either implicitly or by the high-level language program, or explicitly by the OVRDBF command), then SEQONLY(*YES) is ignored and SEQONLY(*NO) is used.
- Record-level changes made under commitment control are recorded in a journal. These changes can be applied to or removed from the database with the Apply Journaled Changes (APYJRNCHG) command or the Remove Journaled Changes (RMVJRNCHG) command.

Images of the files before (before-image) and after (after-image) they are changed are journaled under commitment control. If you specified only to journal the after-images of the files, the system also automatically journals the before-image of the file changes that occurred under commitment control. However, because the before-images are not captured for all changes made to the files, the RMVJRNCHG command cannot be used to remove these before-images from the files in the database.

- Changes that are made under commitment control that are not record-level changes are not recorded in any journal. Therefore, these changes cannot be applied to or removed from the database using the APYJRNCHG and RMVJRNCHG commands.
- Object-level and record-level changes made under commitment control using SQL will use the commitment definition that is currently active for the activation group that the requesting program is running in. If neither the job-level nor the activation-group-level commitment definition is active, SQL will start an activation-group-level commitment definition. See the SQL/400 User's Guide for more information regarding changes made under commitment control using SQL.
- All remote resources under commitment control must have the same remote location name, local location name, device, mode, remote network ID, transaction program name (TPN) and user ID. If a commitment boundary is established and all resources are removed, the location can be changed.
- Do not use the Submit Remote Command (SBMRMTCMD) command to start commitment control or perform any other commitment control operations.
- The COMMIT and ROLLBACK CL commands will fail if SQL is in the call stack and the remote relational database is not on an AS/400 system. If SQL is not on the call stack, the COMMIT and ROLLBACK commands will not fail.
- Commitment control must be started on the source system before making committable changes to remote resources. The system automatically starts commitment control for distributed database SQL on the source system at connection time if the SQL program is running with the commitment control option other than *NONE. When the first remote resource is placed under commitment control, the system starts commitment control on the target system.
- A save operation is prevented if *the job performing the save* has one or more active commitment definitions that are not at a commitment boundary. This prevents pending changes due to a partial transaction from being saved to the save media.

  Note: Object locks and record locks prevent pending changes from commitment definitions in *other* jobs from being saved to the save media. This is true only for API commitment resources if locks are acquired when changes are made to the object or objects associated with the API commitment resource.

  If the job attempting the save operation has one or more commitment definitions with an added API resource, then the save operation is prevented.

Commitment Control Errors

When you use commitment control, it is important to understand which conditions cause errors and which do not. In general, errors occur when commitment control functions are used inconsistently, such as running an End Commitment Control (ENDCMTCTL) command when files that use the commitment definition are still open.

Error Conditions: The following are some typical errors related to commitment control:

If an error occurs, an escape message is sent that you can monitor for in a CL program.

- Consecutive STRCMTCTL commands are run without an intervening ENDCMTCTL command.
- Files are opened under commitment control, but no STRCMTCTL command was run.

  This is not an error condition for programs that run within an activation group that are to use the job-level commitment definition. The job-level commitment definition can be started only by a single program, but once started by a program, the job-level commitment definition

is used by any program running in any activation group that is not using an activation-group-level commitment definition. Programs that run within an activation group and are to use the activation-group-level commitment definition must first start the activation-group-level commitment definition with the STRCMTCTL command.

- Files opened under commitment control within the same transaction are not journaled.
- Files opened under commitment control are not journaled, within the same transaction, to the same journal.
- More than 256 distinct database file members opened under commitment control, or closed with pending changes, are associated with the same commitment definition.
- The first open operation of a shared file places the file under commitment control, but subsequent open operations of the same shared file do not.
- The first open operation of a shared file does not place the file under commitment control, but subsequent open operations of the same shared file do.
- More than 32768 records locked in a single transaction.
- The program issues a read operation, a commit operation, and a change to the same record. The read operation must be issued again after the commit operation because the commit operation has freed the lock on the record.
- Resources placed under commitment control do not reside at the same location as resources already under commitment control for the commitment definition.
- Uncommitted changes exist when an ENDCMTCTL command is issued.

  This is not an error condition for the ENDCMTCTL command if all files are closed, any remote database is disconnected, and no API commitment resource is still associated with the commitment definition to be ended.
- A commit, rollback, or ENDCMTCTL command is run, and a STRCMTCTL command was not run.

  This is not an error condition for programs that run within an activation group and the job-level commitment definition is active. The job-level commitment definition can be started only by a single program, but once started by a program, the job-level commitment definition is used by any program running in any activation group that is not using an activation-group-level commitment definition. Programs that run within an activation group and are to use the activation-group-level commitment definition must first start the activation-group-level commitment definition with the STRCMTCTL command.
- An ENDCMTCTL command is run with files still open under commitment control for the commitment definition.
- A job performing a save operation has one or more commitment definitions that are not at a commitment boundary.
- A save-while-active operation ended because other jobs with committable resources did not reach a commitment boundary in the time specified for the SAVACTWAIT parameter.
- A save-while-active process was not able to continue because of API-committable resources being added to more than one commitment definition for a single job.
- More than 1023 commitment definitions for a single job.

Non-Error Conditions: The following are some situations for commitment control in which no errors occur:

- A commit or rollback operation is run and no resources are under commitment control. This allows you to include commit or rollback operations in your program without considering whether there are resources under commitment control. It also allows you to specify a commit identification before making any committable changes.
- A commit or rollback operation is run and there are no uncommitted resource changes. This allows you to include commit and rollback operations within your program without considering whether there are uncommitted resource changes.
- A file under commitment control is closed and uncommitted records exist. This situation

allows another program to be called to perform the commit or rollback operation. This occurs regardless of whether or not the file is shared. This function allows a subprogram to make database changes that are part of a transaction involving multiple programs.

- A job ends, either normally or abnormally, with uncommitted changes for one or more commitment definitions. The changes for all commitment definitions are rolled back.
- An activation group ends with pending changes for the activation-group-level commitment definition. If the activation group is ending normally and there are no errors encountered when closing any files opened under commitment control scoped to the same activation group that is ending, an implicit commit is performed by the system. Otherwise, an implicit rollback is performed.
- A program accesses a changed record again that has not been committed. This allows a program to:
  - Add a record and update it before specifying the commit operation.
  - Update the same record twice before specifying the commit operation.
  - Add a record and delete it before specifying the commit operation.
  - Access an uncommitted record again by a different logical file (under commitment control).
- You specify LCKLVL(*CHG or *CS) on the STRCMTCTL command and open a file with a commit operation for read only. In this case, no locks occur on the request. It is treated as if commitment control is not in effect, but the file does appear on the WRKJOB menu option of files under commitment control.
- You issue the STRCMTCTL command and do not open any files under commitment control. In this situation, any record-level changes made to the files are not made under commitment control.

Monitoring for Errors after a CALL Command:

When a program that uses commitment control is called, you should monitor for unexpected errors and perform a rollback operation if an error occurs. For example, uncommitted records can exist when a program encounters an unexpected error such as an RPG divide-by-zero error. Depending on the status of the inquiry message reply (INQMSGRPY) parameter for a job, the program sends an inquiry message or performs a default action. If the operator response or the default action ends the program, uncommitted records still exist waiting for a commit or rollback operation.

If another program is called and causes a commit operation, the partially completed transaction from the previous program is committed.

To prevent partially completed transactions from being committed, monitor for escape messages after the CALL command. For example, if it is an RPG program, use the following coding:

```
CALL RPGA
MONMSG MSGID(RPG9001)
     EXEC(ROLLBACK) /*Rollback if pgm is canceled*/
```

If it is a COBOL program:

```
CALL COBOLA
MONMSG MSGID(CBE9001)
       EXEC(ROLLBACK) /*Rollback if pgm is canceled*/
```

Error Messages to Monitor for Commitment Control:

Several different error messages can be returned by the commit or rollback operations depending on when the error occurred. Some of the messages related to commitment control to look for are:

CPD8351 Changes may not have been committed.

CPD8353 Changes to relational database &1 may not have been committed.

CPD8354 Changes to DDM file &1 may not have been committed.

CPD8355 Changes to DDL object &1 may not have been committed.

CPD8356 Rolled back changes may have been committed.

CPD8358 Changes to relational database &1 may not have been rolled back.

CPD8359 Changes to DDM file &1 may not have been rolled back.

CPD835A Changes to DDL object &3 may not have been rolled back.

CPD8361 API exit program &1 failed during commit.

CPD8362 API exit program &1 failed during roll back.

CPD8363 API exit program &1 ended after &4 minutes during commit.

CPD8364 API exit program &1 ended after &4 minutes during rollback.

CPF8359 Rollback operation failed.

CPF835B Errors occurred while ending commitment control.

CPF8363 Commit operation failed.

CPF8367 Cannot perform commitment control operation.

CPF8369 Cannot place API commitment resource under commitment control; reason code &1.

These messages can occur during:

- Normal commit or rollback processing
- Commit or rollback processing during job process end
- Commit or rollback processing during activation group end

Note: None of the above messages can be monitored for during activation group end or job process end. Any errors encountered when ending an activation-group-level commitment definition during activation group end or any commitment definition during job end are left in the job log as diagnostic messages. The following sections show you when you can expect which messages.

Normal Commit or Rollback Processing:

Errors may occur at any time during commit or rollback processing. The following table divides this processing into four situations. The middle column describes the actions taken by the system when it encounters errors during each situation. The third column suggests what you or your application should do in response to the messages. These suggestions are consistent with the way commitment control processing is handled by the system.

Table 4-4. Commit and Rollback Processing for Failures.

Situation	Commit or Rollback Processing	Suggested Action
Record- level I/O **commit** fails	• Returns CPD8351 • Returns CPF8363 at end of processing • Exits commit processing without attempting any object-level or API commitment resource (no implicit rollback occurs)	Either: • Do explicit rollback if possible or, • Notify application to do an explicit rollback
Object-level or commit and rollback exit program for API commitment resource fails during commit	1. Returns one of the following messages depending on the commitment resource type: • CPD8353 • CPD8354 • CPD8355 • CPD8361 2. Continues processing 3. Returns CPF8363 at end of processing	Monitor for messages; handle as desired
Record-level I/O **rollback** fails	1. Returns CPD8356 2. Attempt to continue processing to rollback object-level or API commitment resources 3. Returns CPF8359 at end of processing	Monitor for messages; handle as desired
Object-level or commit and rollback exit program for API commitment resources fails during rollback	1. Returns one of the following messages depending on the commitment resource type: • CPD8358 • CPD8359 • CPD835A • CPD8362 2. Continues processing 3. Returns CPF8359 at end of processing	Monitor for messages; handle as desired

Commit or Rollback Processing During Job End: All of the situations described in Table 4-4 also apply when a job is ending except that message CPF835B is sent instead of message CPF8359 or CPF8363. In addition, one of two messages may appear specific to job completion if a commit and rollback exit program for an API-committable resource has been called. If the commit and rollback exit program does not complete within 5 minutes, the program is canceled, a diagnostic message CPD8363 (for commit) or CPD8364 (for rollback) is sent, and the remainder of the commit or rollback processing continues.

Commit or Rollback Processing During IPL: All of the situations described in Table 4-4 also apply during IPL recovery for commitment definitions except that message CPF835F is sent instead of message CPF8359 or CPF8363. Messages that get sent for a particular commitment definition may appear in the job log for one of the QDBSRVxx jobs or the QHST log. In the QHST log, message CPI8356 indicates the beginning of IPL recovery for a particular commitment definition. Message CPC8351 indicates the end of IPL recovery for a particular commitment definition and any other messages regarding the recovery of that commitment definition is found between those two messages.

One of two messages may appear specific to a commitment definition if a commit and rollback exit program for an API-committable resource has been called. If the commit and rollback exit program does not complete within 5 minutes, the program is canceled, a diagnostic message CPD8363 (for commit) or CPD8364 (for rollback) is sent, and the remainder of the commit or rollback processing continues.

Example of Using Commitment Control

Note: The following example does not include the Integrated Language Environment C/400.

The advantages of using commitment control are illustrated in the following example. Assume that

the following application program does not use commitment control. The records read for updating are locked by the system. The following steps describe how the application program transfers funds from a savings account to a checking account:

- Program A locks and retrieves the savings record. (This action could require a wait if the record is locked by another program.)
- Program A locks and retrieves the checking record. (This could also require a wait.) Program A now has both records locked, and no other program can change them.
- Program A updates the savings record. This causes the record to be released so it is now available to be read for update by any other program.
- Program A updates the checking record, which causes the record to be released so it is now available to be read for update by any other program.

Without using commitment control, a problem needs to be solved to make this program work properly in all circumstances. For example, a problem occurs if program A does not update both records because of a job or system failure. In this case, the two files are not consistent — funds are removed from the savings account, but they are not added to the checking account. Using commitment control allows you to ensure that all changes involved in the transaction are completed or that the files are returned to their original state if the processing of the transaction is interrupted.

If commitment control is used, the preceding example is changed as follows:

1. Commitment control is started.
2. Program A locks and retrieves the savings record. (This action could require a wait if the record is locked by another program.)
3. Program A locks and retrieves the checking record. (This could also require a wait.) Program A now has both records locked, and no other program can change them.
4. Program A updates the savings record, and commitment control keeps the lock on the record.
5. Program A updates the checking record, and commitment control keeps the lock on the record.
6. Program A commits the transaction. The changes to the savings record and the checking record are made permanent in the files. The changes are recorded in the journal, which assumes they will appear on disk. Commitment control releases the locks on both records. The records are now available to be read for update by any other program.

Because the locks on both records are kept by commitment control until the transaction is committed, a situation cannot arise in which one record is updated and the other is not. If a routing step or system failure occurs before the transaction is committed, the system removes (rolls back) the changes that have been made so that the files are updated to the point where the last transaction was committed.

When using commitment control, ensure that all database files to be placed under commitment control in a single environment are first journaled to the same journal.

For each routing step in which files are to be under commitment control, the steps shown in Figure 4-2 on page 4-33 occur:

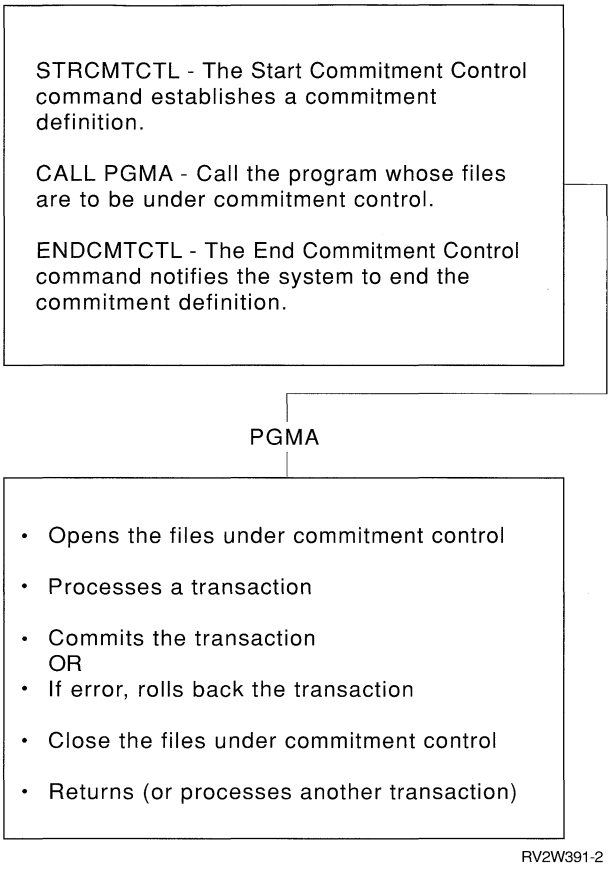


Figure 4-2. Routing Steps with Files under Commitment Control

The operations that are performed under commitment control are journaled to the journal. Using the transfer of funds from a savings account to a checking account as an example, the major entries shown in Figure 4-3 are placed in the journal. (For a description of all the journal entries and their specific content, see the manual *System Operator's Guide*, SC41-8082.)

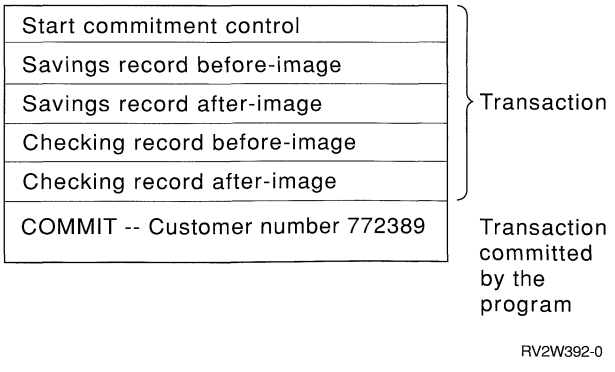


Figure 4-3. Journal Entries for Transfer of Funds from Savings to Checking

The start commitment control journal entry appears after the first file open entry under commitment control. This is because the first file open entry determines what journal is used for commitment control. The journal entry from the first open operation is then used to check subsequent open operations to ensure all files are using the same journal.

When a job failure or system failure occurs, the resources under commitment control are updated to a commitment boundary. If a transaction is started but is not completed before a routing step ends, that transaction is rolled back by the system and does not appear in the file after the routing step ends. If the system abnormally ends before a transaction is completed, that transaction is rolled back by the system and does not appear in the file after a subsequent successful initial program load (IPL) of licensed internal code. Anytime a rollback occurs, reversing entries are placed in the journal.

For example, assume you have a balance of $100.00 and a customer takes out $20.00, for a new balance of $80.00. The database update causes both before-image ($100.00) and after-image ($80.00) journal entries. The journal entries appear as in the example in Figure 4-4

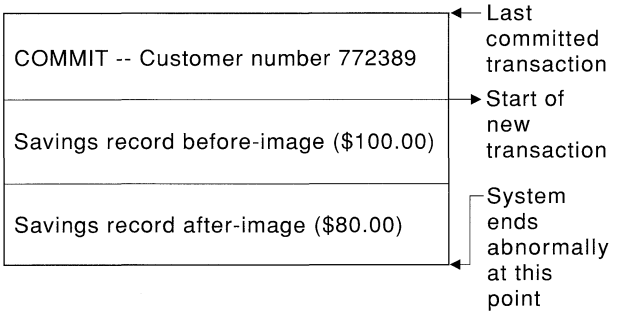


Figure 4-4. Journal Entries for a System That Ended Abnormally

Assume the system abnormally ended after journaling the entries, but before reaching the commitment point or rollback point. After the IPL, the system reads the journal entry and updates the corresponding database record. This update produces two journal entries that reverse the update: the first entry is the before-image ($80.00) and the second entry is the after-image ($100.00) as shown in Figure 4-5 on page 4-34.

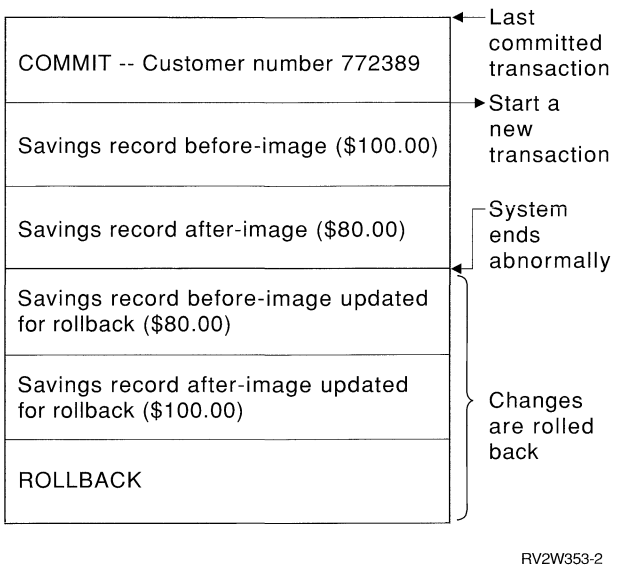


Figure 4-5. Journal Entries for Rollback Changes

When the IPL is successfully completed after the abnormal end, the system removes (or rolls back) any database changes that are not committed. In the preceding example, the system removes the changes from the savings record because a commit operation is not in the journal for that transaction. In this case, the before-image of the savings record is placed in the file. The journal contains the rolled back changes, and an indication that a rollback operation occurred, as in the example shown in Figure 4-5.

Example of Transaction Logging File

A **transaction logging file** is used to start an application again after a system or job failure when a notify object is not used. A transaction logging file is often used in interactive applications to summarize the effects of a transaction.

For example, in an order entry application, a record is usually written to a transaction logging file for each item ordered. The record contains the item ordered, the quantity, and the price. In an accounts payable application, a record is written to a transaction logging file for each account number that is to receive a charge. This record normally contains such information as the account number, the amount charged, and the vendor.

In many of the applications where a transaction logging file already exists, a workstation user can request information about the last transaction entered. By adding commit control to the applications in which a transaction logging file already exists, you can:

- Ensure that the database files are updated to a commitment boundary.
- Simplify the starting of the transaction again.

You must be able to uniquely identify the workstation user if you use a transaction logging file for starting applications again under commit control. If unique user profile names are used on the system, that profile name can be placed in a field in the transaction logging record. This field can be used as the key to the file.

The following examples (Figure 4-6 through 4-10) assume that an order inventory file is being used to issue transactions and that a transaction logging file already exists. The program does the following:

1. Prompts the workstation user for a quantity and item number.
2. Updates the quantity in the production master file (PRDMSTP).
3. Writes a record to the transaction logging file (ISSLOGL).

If the inventory quantity on hand is insufficient, the program rejects the transaction. The workstation user can ask the program where the data entry was interrupted, since the item number, description, quantity, user name, and date are written to the transaction logging file.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R PRDMSTR                 TEXT('Master record')
  2.00      A            PRODCT         3       COLHDG('Product' 'Number')
  3.00      A            DESCRP        20       COLHDG('Description')
  4.00      A            ONHAND         5  0    COLHDG('On Hand' 'Amount')
  5.00      A                                    EDTCDE(Z)
  6.00      A          K PRODCT
```

RV2W358-0

Figure 4-6. DDS for Physical File PRDMSTP

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R ISSLOGR                 TEXT('Product log record')
  2.00      A            PRODCT         3       COLHDG('Product' 'Number')
  3.00      A            DESCRP        20       COLHDG('Description')
  4.00      A            QTY            3  0    COLHDG('Quantity')
  5.00      A                                    EDTCDE(Z)
  6.00      A            USER          10       COLHDG('User' 'Name')
  7.00      A            DATE           6  0    EDTCDE(Y)
  8.00      A                                    COLHDG('Date')
```

RV2W359-0

Figure 4-7. DDS for Physical File ISSLOGP Used by ISSLOGP

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A                                    LIFO
  2.00      A          R ISSLOGR                 PFILE(ISSLOGP)
  3.00      A          K USER
```

RV2W360-0

Figure 4-8. DDS for Logical File ISSLOGL

Example of Transaction Logging File

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

 1.00       A                                      REF(ISSLOGP)
 2.00       A          R PROMPT
 3.00       A                                      CA03(98 'End of program')
 4.00       A                                      CA02(97 'Where am I')
 5.00       A                                    1 20'ISSUES PROCESSING'
 6.00       A                                    3  2'Quantity'
 7.00       A            QTY       R        I    +1
 8.00       A  62                                  ERRMSG('Not enough +
 9.00       A                                      Qty' 62)
 10.00      A                                   +6'Product'
 11.00      A            PRODCT    R        I    +1
 12.00      A  61                                  ERRMSG('No Product +
 13.00      A                                      record found' 62)
 14.00      A  55                               15  2'No Previous record exists'
 15.00      A                                   24  2'CF2 Last transaction'
 16.00      A          R RESTART
 17.00      A                                    1 20'LAST TRANSACTION +
 18.00      A                                        INFORMATION'
 19.00      A                                    5  2'Product'
 20.00      A            PRODCT    R             +1
 21.00      A                                    7  2'Description'
 22.00      A            DESCRP    R             +1
 23.00      A                                    9  2'Qty'
 24.00      A            QTY       R            +1REFFLD(QTY)
                                                                    RSLS848-1
```

Figure 4-9. DDS for Display File PRDISSD Used in the Program

This process is outlined in Figure 4-10 on page 4-37.

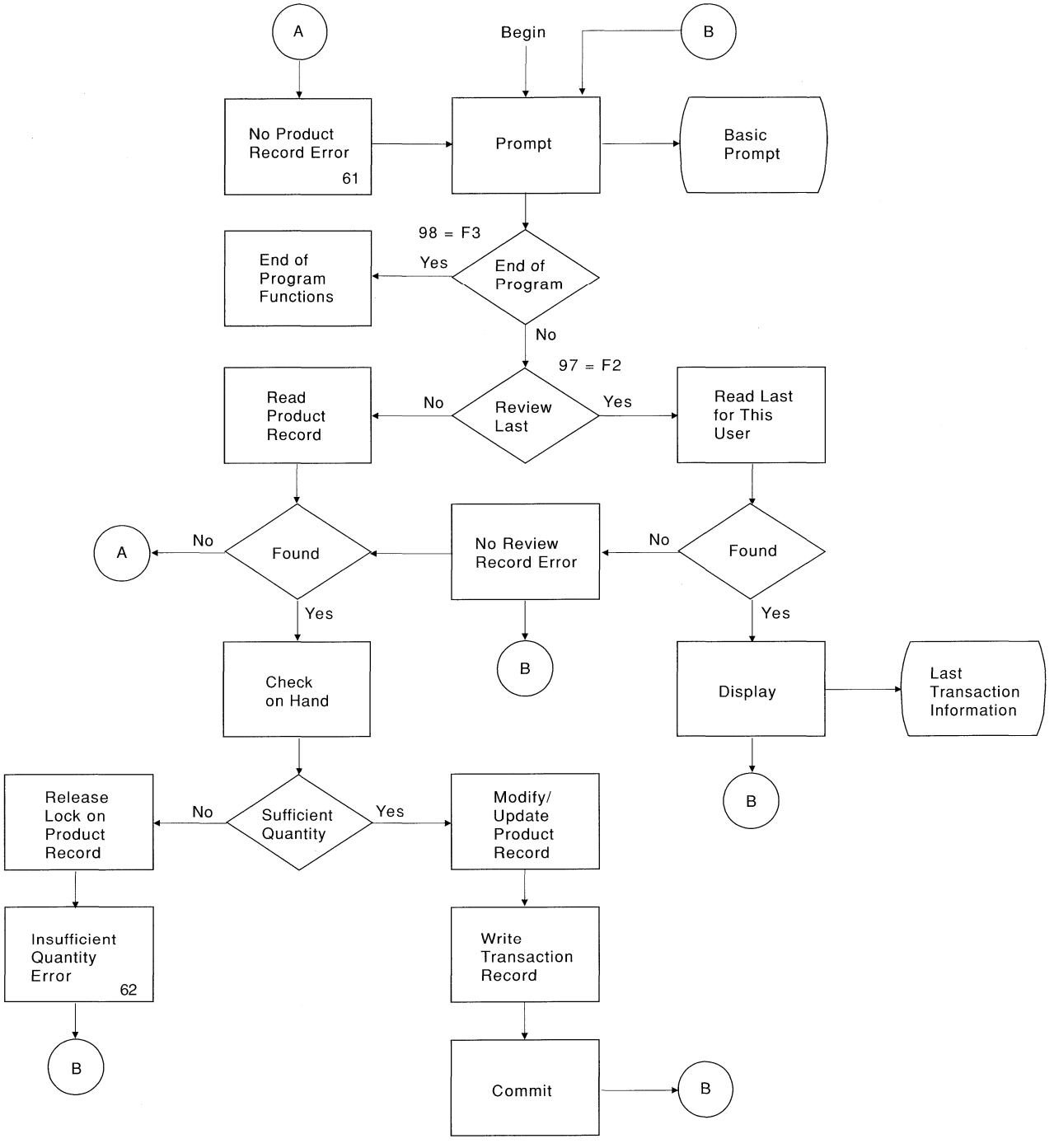


Figure 4-10. Program Flow

The RPG COMMIT operation code is specified after the PRDMSTP file is updated and the record is written to the transaction logging file. Since each prompt to the operator represents a boundary for a new transaction, the transaction is considered a single Enter transaction.

The user name is passed to the program when it is called. The access path for the transaction logging file is defined in last-in-first-out (LIFO) sequence so the program can easily access the last record entered.

The workstation user can start the program again after a system or job failure by using the same function that identified where data entry was stopped. No additional code needs to be added to the program. If you are currently using a trans-

action logging file but are not using it to find out where you are, add the user name to the transaction logging file (assuming user names are unique) and use this approach in the program.

Figure 4-11 shows the RPG program used. Statements required for commitment control are shown in boxes.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00      FISSLOGL IF  E           K        DISK         KCOMIT      A
  3.00      FPRDISSD CF  E                    WORKSTN
  4.00      C           *ENTRY    PLIST
  5.00      C                     PARM           USER   10
  6.00      C*
  7.00      C*  Initialize fields used in Trans Log Rcd
  8.00      C*
  9.00      C                     MOVE UDATE     DATE
 10.00      C*
 11.00      C*  Basic processing loop
 12.00      C*
 13.00      C           LOOP      TAG
 14.00      C                     EXFMTPROMPT
 15.00      C   98                GOTO END                        End of pgm
 16.00      C   97                DO                              Where am I
 17.00      C                     EXSR WHERE
 18.00      C                     GOTO LOOP
 19.00      C                     END
 20.00      C           PRODCT    CHAINPRDMSTR             61     Not found
 21.00      C   61                GOTO LOOP
 22.00      C           ONHAND    SUB QTY        TEST    50   62  Less than
 23.00      C   62                DO                              Not enough
 24.00      C                     EXCPTRLSMST                     Release lck
 25.00      C                     GOTO LOOP
 26.00      C                     END
 27.00      C*
 28.00      C*  Update master record and output the Transaction Log Record
 29.00      C*
 30.00      C                     Z-ADDTEST      ONHAND
 31.00      C                     UPDATPRDMSTR
 32.00      C                     WRITEISSLOGR
 33.00      C                     COMIT
 34.00      C                     GOTO LOOP
 35.00      C*
 36.00      C*  End of program processing
 37.00      C*
 38.00      C           END       TAG
 39.00      C                     SETON                     LR
 40.00      C*
 41.00      C*  WHERE subroutine for 'Where am I' requests
 42.00      C*
 43.00      C           WHERE     BEGSR
 44.00      C           USER      CHAINISSLOGL              55     Not found
 45.00      C  N55                EXFMTRESTART
 46.00      C                     ENDSR
 47.00      OPRDMSTR E                RLSMST
```

RSLS813-2

Figure 4-11. RPG Program

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00            PGM
  2.00            DCL         &USER *CHAR LEN(10)
  3.00            STRCMTCTL   LCKLVL(*CHG)
  4.00            RTVJOBA     USER(&USER)
  5.00            CALL        PRDISS PARM(&USER)
  6.00            MONMSG      MSGID(RPG9001) EXEC(ROLLBACK)
  7.00            ENDCMTCTL
  8.00            ENDPGM
```

RSLS827-0

Figure 4-12. CL Program Used to Call RPG Program PRDISS

To use commitment control in this program, a lock level of *CHG would normally be specified. The record is locked by the change until a commit operation is run. Note that if there is an insufficient quantity of inventory, the record is explicitly released. (If the record were not explicitly released in the program, it would be released when the next record is read for update from the file.)

In this example, there is no additional advantage to using the lock level *ALL. If *ALL were used, a rollback or commit operation would have to be used to release the record when an insufficient quantity existed.

Figure 4-12 is a CL program that calls the RPG program PRDISS. Note the use of STRCMTCTL/ENDCMTCTL commands. The unique user name is retrieved (RTVJOBA command) and passed to the program. The use of the MONMSG command to cause a rollback is described in "Standard Commit Processing Program" on page 4-48.

Starting Application Programs Using a Notify Object

When a program is started after an abnormal end, it can look for an entry in the notify object. If one exists, the program can start a transaction again. After the transaction has been started again, the notify object is cleared by the program to prevent it from starting the same transaction yet another time.

Following are ways you can use a notify object:

- If the commit identification is placed in a database file, query this file to determine where to start each application or workstation job again.
- If the commit identification is placed in a message queue for a particular workstation, a message can be sent to the work station users when they sign on to inform them of the last transaction committed.
- If the commit identification is placed in a database file that has a key or user name, the program can read this file when it is started. If a record exists in the file, start the program again. The program can send a message to the workstation user identifying the last transaction committed. Any recovery is performed by the program. If a record existed in the database file, the program deletes that record at the end of the program.
- For a batch application, the commit identification can be placed in a data area that contains totals, switch settings, and other status information necessary to start the application again. When the application is started, it accesses the data area and verifies the values stored there. If the application ends normally, the data area is set up for the next run.
- For a batch application, the commit identification can be sent to a message queue. A program that is run when the application is started can retrieve the messages from the queue and start the programs again.

There are several techniques for starting your applications again depending on your application needs. In choosing the technique, consider the following:

- When there are multiple users of a program at the same time, a single data area cannot be used as the notify object because after an abnormal system end, the commit identification for each user would overlay each other in the data area.

- Your design for deleting information in the notify object should handle the situation when a failure occurs immediately following use of the information:
  - If information is deleted immediately, it would not exist if another failure occurs before processing the interrupted transaction.
  - The information in the notify object should not be deleted until the successful processing of the interrupted transaction. In this case, more than one entry will exist in the notify object if it is a database file or message queue.
  - The program should access the last record if there is more than one entry.
- A notify object cannot be used to provide the work station user with the last transaction committed because the notify object is updated only if a system or job failure occurs or if uncommitted changes exist at the normal end of a job.
- If information is displayed to the workstation user, it must be meaningful. To accomplish this may require that the program translate codes kept in the notify object into information that will help the user start again.
- Information for starting again should be displayed if the work station user needs it. Additional logic in the program is required to prevent information from being displayed again when it is no longer meaningful.
- A single notify object and a standard processing program can provide a starting again function if the notify object is a database file. This standard processing program is called by the programs that require the ability to start again to minimize the changes to each individual program.

Using a Unique Notify Object for Each Program: Using a single, unique notify object for each job allows use of an externally described commit identification even though there may be multiple users of the same program. In the following example (Figure 4-13 on page 4-41 through Figure 4-16 on page 4-43), a database file is used as a notify object and it is used only by this program.

The program has two database files (PRDMSTP and PRDLOCP) that must be updated for receipts to inventory. The display file used by the program is named PRDRCTD. A database file, PRDRCTP, is used as the notify object. This notify object is defined to the program as a file and is also used as the definition of a data structure for the notify function.

The DDS for the physical file PRDMSTP is shown in Figure 4-6 on page 4-35.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R PRDLOCR                     TEXT('Location record')
  2.00      A            PRODCT         3           COLHDG('Product' 'Number')
  3.00      A            LOCATN         6           COLHDG('Location')
  4.00      A            LOCAMT         5  0        COLHDG('Location' 'Amount')
  5.00      A                                       EDTCDE(Z)
  6.00      A          K PRODCT
  7.00      A          K LOCATN
                                                                      RV2W361-0
```

Figure 4-13. DDS for Physical File PRDLOCP

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A                                       REF(PRDMSTP)
  2.00      A          R PROMPT
  3.00      A                                       CA03(98 'End of program')
  4.00      A                                       SETOFF(71 'RESTART')
  5.00      A                                   1  20'PRODUCT RECEIPTS'
  6.00      A                                   3   2'Quantity'
  7.00      A            QTY           3  0I     +1
  8.00      A                                     +6'Product'
  9.00      A            PRODCT   R       I     +1
 10.00      A  61                                   ERRMSG('No record +
 11.00      A                                       found in the +
 12.00      A                                       master file' 62)
 13.00      A                                     +6'Location'
 14.00      A            LOCATN   R       I     +1REFFLD(LOCATN PRDLOCP)
 15.00      A  62                                   ERRMSG('No record +
 16.00      A                                       found in the +
 17.00      A                                       location file' 62)
 18.00      A                                   9   2'Last Transaction'
 19.00      A  71                                 +6'This is restart +
 20.00      A                                        information'
 21.00      A                                       DSPATR(HI BL)
 22.00      A                                  12   2'Quantity'
 23.00      A                                  12  12'Product'
 24.00      A                                  12  23'Location'
 25.00      A                                  12  35'Description'
 26.00      A            LSTPRD   R         14  15REFFLD(PRODCT)
 27.00      A            LSTLOC   R         14  26REFFLD(LOCATN *SRC)
 28.00      A            LSTQTY   R         14   5REFFLD(QTY *SRC)
 29.00      A                                        EDTCDE(Z)
 30.00      A            LSTDSC   R         14  35REFFLD(DESCRP)
                                                                      RSLS849-0
```

Figure 4-14. DDS for Display File PRDRCTD

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A                                      LIFO
  2.00      A                                      REF(PRDMSTP)
  3.00      A          R PRDRCTR
  4.00      A            USER          10
  5.00      A            PRODCT     R
  6.00      A            DESCRP     R
  7.00      A            QTY             3  0
  8.00      A            LOCATN     R           REFFLD(LOCATN PRDLOCP)
  9.00      A          K USER
                                                                    RSLS850-0
```

Figure 4-15. DDS for Notify Object and Externally Described Data Structure (PRDRCTP)

The program processes the notify object as follows:

- At the beginning, the program randomly processes the notify object and displays a record if it exists for the specific key:
  - If multiple records exist, the last record for this key is used because the PRDRCTP file is in LIFO sequence.
  - If no record exists, a transaction was not interrupted so it is not necessary to start again.
  - If the program fails before the first successful commit operation, it does not consider that starting again is required.
- The routine to clear the notify object occurs at the end of the program:
  - If there were multiple failures, the routine can handle deletion of multiple records in the notify object.
  - Although the system places the commit identification in a database file, the commit identification must be specified as a variable in the RPG program.
  - Because RPG allows a data structure to be externally described, a data structure is a convenient way of specifying the commit identification. In this example, the data structure uses the same external description that the database file used as the notify object.

The processing for this program prompts the user for a product number, a location, and a quantity:

- Two files must be updated:
  - Product master file (PRDMSTP)
  - Product location file (PRDLOCP)
- A record in each file must exist before either is updated.
- The program moves the input fields to corresponding last fields after each transaction is successfully entered. These last fields are displayed to the operator on each prompt as feedback for what was last entered.
- If information for starting again exists, it is moved to these last fields and a special message appears on the display.

This process is outlined in Figure 4-16 on page 4-43. The user name is passed to the program to provide a unique record in the notify object.

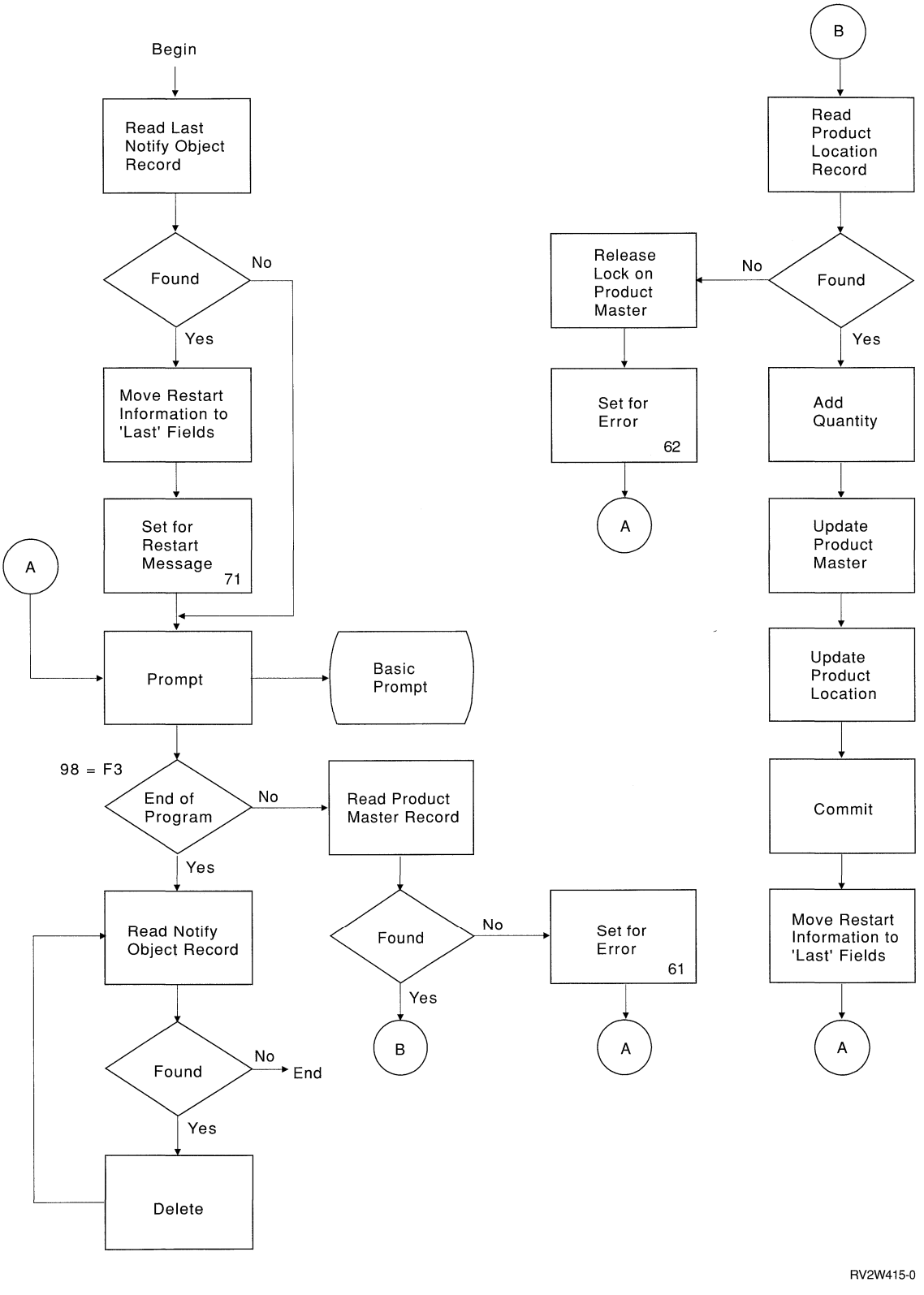


Figure 4-16. Program Flow

Figure 4-17 on page 4-44 shows the RPG source code for this example. The notify object (file PRDRCTP) is used as a normal file at the beginning and end of the program, and is also specified as the notify object in the CL (STRCMTCTL command) before calling the program.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00     FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00     FPRDLOCP UF  E           K        DISK         KCOMIT
  3.00     FPRDRCTD CF  E                    WORKSTN
  4.00     F*
  5.00     F* The following file is the specific notify object for this pgm.
  6.00     F*      It is accessed only in a restart situation and at the
  7.00     F*        end of the program to delete any records.  The records
  8.00     F*        are written to the notify object by Commitment Control.
  9.00     F*
 10.00     FPRDRCTP UF  E           K        DISK
 11.00     ICMTID     E DSPRDRCTP
 12.00     C           *ENTRY    PLIST
 13.00     C                     PARM           USER10 10
 14.00     C                     MOVE USER10    USER
 15.00     C*
 16.00     C*  Check for restart information - get last rcd per user
 17.00     C*    PRDRCTP file access path is in LIFO sequence
 18.00     C*
 19.00     C           USER      CHAINPRDRCTR             20     Not found
 20.00     C  N20                DO                               Restart
 21.00     C                     EXSR MOVLST                      Move to last
 22.00     C                     SETON                    71     Restart
 23.00     C                     END
 24.00     C*
 25.00     C*  Basic processing loop
 26.00     C*
 27.00     C           LOOP      TAG
 28.00     C                     EXFMTPROMPT
 29.00     C   98                GOTO END                         End of pgm
 30.00     C           PRODCT    CHAINPRDMSTR             61     Not found
 31.00     C   61                GOTO LOOP
 32.00     C           KEY       KLIST
 33.00     C                     KFLD           PRODCT
 34.00     C                     KFLD           LOCATN
 35.00     C           KEY       CHAINPRDLOCR             62     Not found
 36.00     C   62                DO
 37.00     C                     EXCPTRLSMST                      Release lck
 38.00     C                     GOTO LOOP
 39.00     C                     END
 40.00     C                     ADD  QTY       ONHAND            Add
 41.00     C                     ADD  QTY       LOCAMT
 42.00     C                     UPDATPRDMSTR                     Update
 43.00     C                     UPDATPRDLOCR                     Update
 44.00     C*
 45.00     C*  Commit and move to previous fields
 46.00     C*
 47.00     C           CMTID     COMIT
 48.00     C                     EXSR MOVLST                      Move to last
 49.00     C                     GOTO LOOP
 50.00     C*
 51.00     C*  End of program processing
 52.00     C*
 53.00     C           END       TAG
 54.00     C                     SETON                    LR
 55.00     C*
 56.00     C*  Delete any records in the notify object
 57.00     C*
 58.00     C           DLTLP     TAG
 59.00     C           USER      CHAINPRDRCTR             20     Not found
 60.00     C  N20                DO
 61.00     C                     DELETPRDRCTR                     Delete
 62.00     C                     GOTO DLTLP
 63.00     C                     END
 64.00     C*
 65.00     C*  Move to -Last Used- fields for operator feedback
 66.00     C*
 67.00     C           MOVLST    BEGSR
 68.00     C                     MOVE PRODCT    LSTPRD
 69.00     C                     MOVE LOCATN    LSTLOC
 70.00     C                     MOVE QTY       LSTQTY
 71.00     C                     MOVE DESCRP    LSTDSC
 72.00     C                     ENDSR
 73.00     OPRDMSTR E                RLSMST                                  RSLW153-0
```

Figure 4-17. RPG Source

Using a Single Notify Object for All Programs:

Using a single notify object for all programs is advantageous since all information required to start again is in the same object and a standard approach to the notify object can be used in all programs. In this situation, use a unique combination of user and program identifications to make sure that the program accesses the correct information when it starts again.

Because the information required to start again may vary from program to program, an externally described data structure for the commit identification should not be used. If a single notify object is used, the preceding program could describe the data structure within the program rather than externally. For example:

```
 1   10       USER
11   20       PGMNAM
21   23       PRODCT
24   29       LOCATN
30   49       DESC
50   51  0    QTY
52  220       DUMMY
```

In each program that uses this notify object, the information specified for the commit identification would be unique to the program (the user and program names are not unique). The notify object must be large enough to contain the maximum information that any program would place in the commit identification.

Using a Standard Processing Program:

A standard processing program is one way to start your application again using one database file as the notify object for all applications. This approach assumes that user profile names are unique by user for all applications using the standard program.

For this approach, the physical file NFYOBJP is used as the notify object and defined as:

```
Unique user profile name   10 characters
Program identification     10 characters
Information for
     starting again        Character field
                           (This should be large
                           enough to contain the maximum
                           amount of information for starting
                           programs again that require
                           information for starting again.
                           This field is required by
                           the application programs.
                           In the example, it is
                           assumed to be a length of 200.)
```

The file is created with SHARE(*YES). The first two fields in the file are the key to the file. (This file can also be defined as a data structure in RPG programs.)

Processing Flow: The standard program is called from applications that must start again. The application programs pass this parameter list to the standard program:

- Request code
- Return code
- Data structure name (the contents of the notify object)

Request codes do the following:

- R (Read)

  Retrieves the last record added to the notify object with the same key. The return code is set as:

  0 No record is available (no start again required).

  1 Record returned in the information field for starting again (start again required).

- W (Write)

  Writes a record to the file. This code could be used if you use a notify object for your own purposes. For example, if the program determines that the transaction needs to be started again, the program could write a record to the notify object to simulate what the system will do if a job or the system fails.

- D (Delete)

  Deletes all records in the notify object with the same key. The return code is set as:

  0 No records exist to be deleted.

  1 One or more records were deleted.

- O (Open)

  The O request code is optional and is used to avoid having to start the processing program each time it is called.

- C (Close)

  After the open request code is used, using the close request code ensures the file is closed.

- S (Search)

  Returns the last record for this user. The program name is not used. This code can be

used in an initial program to determine whether starting again is required.

Application Program Example: Following is an example of using a standard processing program. The application shown in Figure 4-18 on page 4-47 performs as follows:

1. The application program receives the user name in a parameter and uses it with the program name as a unique identifier in the notify object.
2. The application program passes a request code of R to the standard commit processing program, which determines if a record exists in the notify object.
3. If the standard commit processing program returns a code of 1, a record was found and the application program presents the information needed to start again to the user.
4. The application program proceeds with normal processing.
5. When a transaction is completed, values are saved for reference so the workstation user can see what was done for the previous transaction.

   The information saved is not provided by the notify object because the notify object is updated only if a job or system failure occurs.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..
  1.00      FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00      FPRDLOCP UF  E           K        DISK         KCOMIT
  3.00      FPRDRCTD CF  E                    WORKSTN
  4.00      F*
  5.00      F* The following is a compile time array which contains the
  6.00      F*    restart information used in the next example
  7.00      F*
  8.00      E                    RTXT   50  50  1               Restart text
  9.00      I*
 10.00      I* Data structure used for info passed to notify object
 11.00      I*
 12.00      ICMTID       DS
 13.00      I                                        1  10 USER
 14.00      I                                       11  20 PGMNAM
 15.00      I                                       21  23 PRODCT
 16.00      I                                       24  29 LOCATN
 17.00      I                                       30  49 DESCRP
 18.00      I                                     P 50  51OQTY
 19.00      I                                       52 170 DUMMY
 20.00      I                                      171 220 RSTART
 21.00      C           *ENTRY    PLIST
 22.00      C                     PARM          USER10 10
 23.00      C*
 24.00      C*  Initialize fields used to communicate with std program
 25.00      C*
 26.00      C                     MOVE USER10   USER
 27.00      C                     MOVEL'PRDRC2' PGMNAM
 28.00      C                     MOVE 'R'      RQSCOD          Read Rqs
 29.00      C                CALL 'STDCMT'
 30.00      C                     PARM          RQSCOD  1
 31.00      C                     PARM          RTNCOD  1
 32.00      C                     PARM          CMTID 220       Data struct
 33.00      C           RTNCOD    IFEQ '1'                      Restart
 34.00      C                     EXSR MOVLST                   Move to last
 35.00      C SETON                                       71    Restart
 36.00      C                     END
 37.00      C*
 38.00      C*  Initialize fields used in notify object
 39.00      C*
 40.00      C                     MOVEARTXT,1   RSTART          Move text
 41.00      C*
 42.00      C* Basic processing loop
 43.00      C*
 44.00      C           LOOP      TAG
 45.00      C                     EXFMTPROMPT
 46.00      C   98                GOTO END
 47.00      C           PRODCT    CHAINPRDMSTR            61    Not found
 48.00      C   61                GOTO LOOP
 49.00      C           KEY       KLIST
 50.00      C                     KFLD          PRODCT
 51.00      C                     KFLD          LOCATN
```

RSLW154-0

Figure 4-18 (Part 1 of 2). Application Program Example

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..
 52.00      C           KEY       CHAINPRDLOCR              62    Not found
 53.00      C   62                DO
 54.00      C                     EXCPTRLSMST                     Release lck
 55.00      C                     GOTO LOOP
 56.00      C                     END
 57.00      C                     ADD  QTY       ONHAND           Add
 58.00      C                     ADD  QTY       LOCAMT
 59.00      C                     UPDATPRDMSTR                    Update
 60.00      C                     UPDATPRDLOCR                    Update
 61.00      C*
 62.00      C*  Commit and move to previous fields
 63.00      C*
 64.00      C           CMTID     COMIT
 65.00      C                     EXSR MOVLST                     Move to last
 66.00      C                     GOTO LOOP
 67.00      C*  End of program processing
 68.00      C*
 69.00      C           END       TAG
 70.00      C                     MOVE 'D'       RQSCOD           Dlt Rqs
 71.00      C                     CALL 'STDCMT'
 72.00      C                     PARM           RQSCOD
 73.00      C                     PARM           RTNCOD
 74.00      C                     PARM           CMTID
 75.00      C                     SETON                     LR
 76.00      C*
 77.00      C*  Move to -Last Used- fields for operator feedback
 78.00      C*
 79.00      C           MOVLST    BEGSR
 80.00      C                     MOVE PRODCT    LSTPRD
 81.00      C                     MOVE LOCATN    LSTLOC
 82.00      C                     MOVE DESCRP    LSTDSC
 83.00      C                     MOVE QTY       LSTQTY
 84.00      C                     ENDSR
 85.00     OPRDMSTR E                 RLSMST
 86.00 ** RTXT       Restart Text
 87.00 Inventory Menu - Receipts Option
```

RSLW155-0

Figure 4-18 (Part 2 of 2). Application Program Example

Standard Commit Processing Program: The standard commit (STDCMT) processing program performs the functions required to communicate with a single notify object used by all applications. While the commitment control function automatically writes an entry to the notify object, a user-written standard program must process the notify object. The standard program simplifies and standardizes the approach.

The program is written to verify the parameters that were passed and perform the appropriate action as follows:

O=Open

The calling program requests the notify object file be kept open on return. Because the notify object is opened implicitly by the RPG program, the program should not close it. Indicator 98 is set so the program returns with LR off to keep the program's work areas and leaves the notify object open so it can be called again without excess overhead.

C=Close

The calling program has determined it no longer needs the notify object and requests a

close. Indicator 98 is set off to allow a full close of the notify object.

R=Read — The calling program requests that a record with matching key fields be read and passed back. The program uses the passed key fields to attempt to retrieve a record from NFYOBJP. If duplicate records exist for the same key, the last record is returned. The return code is set accordingly and, if the record existed, it is passed back in the data structure CMTID.

W=Write — The calling program requests a record to be written to the notify object to allow the calling program to start again the next time it is called. The program writes the contents of the passed data as a record in NFYOBJP.

D=Delete — The calling program requests that records for this matching key be deleted. This function is usually performed at the successful completion of the calling program to remove any information on starting again. The program attempts to delete any records for passed key fields. If no records exist, a different return code is passed back.

S=Search — The calling program requests a search for a record for a particular user regardless of which program wrote it. This function is used in the program for sign-on to indicate that starting again is required. The program uses only the user name as the key to see if records exist. The return code is set appropriately, and the contents of the last record for this key (if it exists) are read and passed back.

Figure 4-19 on page 4-50 shows the standard commit processing program, STDCMT.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FNFYOBJP UF  E           K        DISK                    A
  2.00      ICMTID       DS
  3.00      I                                       1  10 UNQUSR
  4.00      I                                      11  20 UNQPGM
  5.00      I                                      21 220 BIGFLD
  6.00      C            *ENTRY    PLIST
  7.00      C                      PARM           RQSCOD  1
  8.00      C                      PARM           RTNCOD  1
  9.00      C                      PARM           CMTID 220
 10.00      C             UNQUSR   CABEQ*BLANKS   BADEND         H1 Invalid
 11.00      C             UNQPGM   CABEQ*BLANKS   BADEND         H2 Invalid
 12.00      C*
 13.00      C*  'O' for Open
 14.00      C*
 15.00      C             RQSCOD   IFEQ 'O'                        Open
 16.00      C                      SETON                     98    End LR
 17.00      C                      GOTO END
 18.00      C                      END
 19.00      C*
 20.00      C*  'C' for Close
 21.00      C*
 22.00      C             RQSCOD   IFEQ 'C'                        Close
 23.00      C                      SETOF                     98
 24.00      C                      GOTO END
 25.00      C                      END
 26.00      C*
 27.00      C*  'R' for Read - Get last record for the key
 28.00      C*
 29.00      C             RQSCOD   IFEQ 'R'                      Read
 30.00      C             KEY      KLIST
 31.00      C                      KFLD           UNQUSR
 32.00      C                      KFLD           UNQPGM
 33.00      C             KEY      CHAINNFYOBJR              51    Not found
 34.00      C   51                 MOVE '0'       RTNCOD
 35.00      C   51                 GOTO END
 36.00      C                      MOVE '1'       RINCOD           Found
 37.00      C             LOOP1    TAG
 38.00      C             KEY      READENFYOBJR                 20 EOF
 39.00      C   20                 GOTO END
 40.00      C                      GOTO LOOP1
 41.00      C                      END
 42.00      C*
 43.00      C*  'W' FOR Write
 44.00      C*
 45.00      C             RQSCOD   IFEQ 'W'                        Write
 46.00      C                      WRITENFYOBJR
 47.00      C                      GOTO END
 48.00      C                      END
 49.00      C*
 50.00      C*  'D' for Delete - Delete all records for the key
 51.00      C*
```

RSLW156-0

Figure 4-19 (Part 1 of 2). Standard Commit Processing Program

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

52.00      C           RQSCOD    IFEQ 'D'                          Delete
53.00      C           KEY       CHAINNFYOBJR              51      Not found
54.00      C   51                MOVE '0'       RTNCOD
55.00      C   51                GOTO END
56.00      C                     MOVE '1'       RTNCOD             Found
57.00      C           LOOP2     TAG
58.00      C                     DELETNFYOBJR
59.00      C           KEY       READENFYOBJR                  20 EOF
60.00      C  N20                GOTO LOOP2
61.00      C                     GOTO END
62.00      C                     END
63.00      C*
64.00      C*  'S' for Search for the last record for this user
65.00      C*              (Ignore the -Program- portion of the key)
66.00      C*
67.00      C           RQSCOD    IFEQ 'S'                          Search
68.00      C           UNQUSR    SETLLNFYOBJR                  20 If equal
69.00      C  N20                MOVE '0'       RTNCOD
70.00      C  N20                GOTO END
71.00      C                     MOVE '1'       RTNCOD             Found
72.00      C           LOOP3     TAG
73.00      C           UNQUSR    READENFYOBJR                  20 EOF
74.00      C  N20                GOTO LOOP3
75.00      C                     GOTO END
76.00      C                     END
77.00      C*
78.00      C*  Invalid request code processing
79.00      C*
80.00      C                     SETON                     H2      Bad RQS code
81.00      C                     GOTO BADEND
82.00      C*
83.00      C*  End of program processing
84.00      C*
85.00      C           END       TAG
86.00      C  N98                SETON                     LR
87.00      C                     RETRN
88.00      C* BADEND tag is used then fall thru to RPG cycle error return
89.00      C           BADEND    TAG
                                                                  RSLW157-0
```

Figure 4-19 (Part 2 of 2). Standard Commit Processing Program

Deciding If It Is Necessary to Start Again: The initial program can call the standard commit processing program to determine if it is necessary to start again. The workstation user can than decide whether or not to start again.

The initial program passes a request code of S (search) to the standard program, which searches for any record for the user. If a record exists, the information for starting again is passed to the initial program and the information is displayed to the workstation user.

The commit identification in the notify object should contain information that the initial program can display identifying what program needs to be started again. For example, the last 50 characters of the commit identification can be reserved to contain this information. In the application program, this information could be in a compile-time array and moved to the data structure in an initialization step. Figure 4-18 on page 4-47 shows how to include this in the application program.

Figure 4-20 on page 4-52 is an example of an initial program that determines if a record exists in the notify object.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00            PGM
  2.00            DCLF        CMTINLD
  3.00            DCL         &RQSCOD *CHAR LEN(1) VALUE(S)  /* Search */
  4.00            DCL         &RTNCOD *CHAR LEN(1)
  5.00            DCL         &CMTID *CHAR LEN(220)
  6.00            DCL         &USER *CHAR LEN(10)
  7.00            DCL         &INFO *CHAR LEN(50)
  8.00            RTVJOBA     USER(&USER)
  9.00            CHGVAR      &CMTID (&USER *CAT XX)
 10.00                        /* The XX is reqrd to prevent a blank Pgm nam */
 11.00            CALL        STDCMT PARM(&RQSCOD &RTNCOD &CMTID)
 12.00            IF          (&RTNCOD *EQ '1') DO /* RESTART REQD */
 13.00            CHGVAR      &INFO %SST(&CMTID 171 50)
 14.00            SNDRCVF     RCDFMT(RESTART)
 15.00            ENDDO
 16.00                        /*                                          */
 17.00                        /* Enter normal initial program statements  */
 18.00                        /*    or -TFRCTL- to first menu program      */
 19.00                        /*                                          */
 20.00            ENDPGM
```

RV2W362-0

Figure 4-20. Initial Program Example

Commitment Control Practice Problem

This practice problem will assist you in understanding commitment control and its requirements. The following steps assume you are familiar with the OS/400 program and the data file utility (DFU), and have read this chapter. Before beginning this problem, do the following:

- Create a special library for this practice problem.
- Create source files and a job description.

Perform the following steps:

1. Create a physical file named ITMP (item master file). The data description specification (DDS) for this file is:

```
10    A    R ITMR
20    A      ITEM     2
30    A      ONHAND   5   0
40    A    K ITEM
```

2. Create a physical file named TRNP (transaction file). This file is used as a transaction log file. The DDS for this file is:

```
10    A    R TRNR
20    A      QTY      5  0
30    A      ITEM     2
40    A      USER     10
```

3. Create a logical file named TRNL (transaction logical). This file is used to assist in starting the application again. The *USER* field is the type LIFO sequence. The DDS for this file is:

```
10                       LIFO
20    A     R TRNR       PFILE (TRNP)
30    A     K USER
```

4. Enter the STRDFU command, and create a DFU application named ITMU for the ITMP file. Accept the defaults offered by DFU during the application definition.

5. Type the command CHGDTA ITMU and enter the following records for the ITMP file:

Item	On Hand
AA	450
BB	375
CC	4000

6. End the program using F3. This entry provides some data against which the program will operate.

7. Create the CL program Item Process (ITMPCSC) as follows:

```
PGM
DCL &USER *CHAR LEN(10)
RTVJOBA USER(&USER)
CALL ITMPCS PARM(&USER)
ENDPGM
```

   This is the control program that calls the ITMPCS program. It retrieves the user name and passes it to the processing program. This application assumes that unique user names are used.

8. Create a display file named ITMPCSD from the DDS shown in Figure 4-22 on page 4-58.

   There are two formats, the first for the basic prompt display and the second to allow the operator to review the last transaction entered. This display file is used by the ITMPCS program.

9. Study the logic flow provided in Figure 4-23 on page 4-59.

10. Enter the STRSEU command and type the source shown in Figure 4-21 on page 4-54.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FITMP    UF  E           K        DISK
  2.00      F*                                        KCOMIT
  3.00      FTRNP    O   E                    DISK
  4.00      F*                                        KCOMIT
  5.00      FTRNL    IF  E           K        DISK
  6.00      F            TRNR                         KRENAMETRNR1
  7.00      FITMPCSD  CF E                    WORKSTN
  8.00      C* Enter parameter with User name for -TRNP- file
  9.00      C           *ENTRY    PLIST
 10.00      C                     PARM           USER   10
 11.00      C           LOOP      TAG
 12.00      C                     EXFMTPROMPT
 13.00      C* Check for CF3 for end of program
 14.00      C   93                DO                                End of Pgm
 15.00      C                     SETON                        LR
 16.00      C                     RETRN
 17.00      C                     END
 18.00      C* Check for CF4 for review last transaction
 19.00      C   94                DO                                Review last
 20.00      C* Check for existence of a record for this user in -TRNL- file
 21.00      C           USER      CHAINTRNR1                64      Not found
 22.00      C   64                GOTO LOOP
 23.00      C                     EXFMTREVW
 24.00      C                     GOTO LOOP
 25.00      C                     END
 26.00      C* Access Item record
 27.00      C           ITEM      CHAINITMR                 62      Not found
 28.00      C* Handle -not found- Condition
 29.00      C   62                GOTO LOOP
 30.00      C* Does sufficient quantity exist
 31.00      C           ONHAND    SUB  QTY       TEST    50   61  Minus
 32.00      C* Handle insufficlent quantity
 33.00      C    61               DO
 34.00      C* Release Item record which was locked by the CHAIN for update
 35.00      C                     EXCPTRLSITM
 36.00      C                     GOTO LOOP
 37.00      C                     END
 38.00      C* Change ONHAND and update the Item record
 39.00      C                     Z-ADDTEST      ONHAND
 40.00      C                     UPDATITMR
 41.00      C* Test for Special Simulation Conditions
 42.00      C           ITEM      IFEQ 'CC'
 43.00      C*      Simulate program need for rollback
 44.00      C           QTY       IFEQ 100
 45.00      C                     SETON                     63      Simult Rlbck
 46.00      C*                    ROLBK
 47.00      C                     GOTO LOOP
 48.00      C                     END
 49.00      C*      Simulate an abnormal program cancellation by Div by zero
 50.00      C*        Operator Should respond -C- to inquiry message
 51.00      C           QTY       IFEQ 101
 52.00      C                     Z-ADDO         ZERO    30
 53.00      C           TESTZ     DIV  ZERO      TESTZ   30       Msg occurs
 54.00      C                     END
 55.00      C*      Simulate an abnormal job cancellation by DSPLY.
 56.00      C*        Operator Should System Request to another job
 57.00      C*         and cancel this one with OPTION(*IMMED)
 58.00      C           QTY       IFEQ 102
 59.00      C           'CC=102'  DSPLY                             Msg occurs
 60.00      C                     END
 61 00      C                     END                               ITEM=CC
 62.00      C* Write the -TRNP- file
 63 00      C                     WRITETRNR
 64.00      C* Commit the update to -ITMP- and write to -TRNP-
 65.00      C*                    COMIT
 66.00      C                     GOTO LOOP
 67.00      OITMR    E                RLSITM
```

RSLW158-0

Figure 4-21. RPG Source Program for ITMPTCS

11. Enter the CRTRPGPGM command to create program ITMPCS from the source entered in the previous step.

12. Type the command CALL ITMPCSC, press Enter, and press F4. A message should appear stating that there are no entries for this operator.

13. Enter the following data to see if the program operates correctly:

Quantity	Item
3	AA
4	BB

14. Press F4. The review display should appear with the BB item last entered. Enter the following data:

Quantity	Item
5	FF (Invalid item number message should occur.)
9000	BB (Insufficient quantity error message should occur.)
100	CC (Rollback message should occur.)
102	CC (RPG DSPLY operation should occur. Press the Enter key.)
101	CC (The program should display an inquiry message stating that a divide-by-zero condition has occurred or end, depending on the setting of job attribute INQMSGRPY. If the inquiry message appears, enter C to cancel the RPG program and then C to cancel the CL program on the subsequent inquiry. This simulates an unexpected error condition.)

15. Type the Display Data command DSPDTA ITMP.

    See if the records AA and BB have been updated correctly. The values should be AA = 447, BB = 371, and CC = 3697. Note that the quantities subtracted from CC occurred, but the transaction records were not written.

16. Type the Create Journal Receiver (CRTJRNRCV) command and the parameter JRNRCV(library name/RCVR1) to create a journal receiver used for commitment control.

    Files operating under the same commitment control boundary must be journaled to the same journal.

17. Type the Create Journal (CRTJRN) command using the parameters JRN(library name/JRNTEST) JRNRCV(library name/RCVR1) to create a journal to be used for commitment control.

18. Type the Start Journal Physical File (STRJRNPF) command using the parameters FILE(ITMP TRNP) JRN(JRNTEST) to journal the files to be used for commitment control.

    The IMAGES parameter uses a default of *AFTER, meaning only after-image changes of the records appear in the journal. The files ITMP and TRNP have now started journaling.

    This problem does not require the normal practice of saving files that are journaled.

19. Type the command CALL ITMPCSC and enter the following transactions:

Quantity	Item
5	AA
6	BB

    End the program by pressing F3.

20. Type the Display Journal command DSPJRN JRNTEST

    Note the entries appearing in the journal. The same sequence of entries (UP = update of ITMP followed by PT = put of TRNP) occurs in the journal as was performed by the program. This is because a logical file is defined over the physical file TRNP and the system overrides the RPG default. If no logical file existed, the RPG assumption of SEQONLY(*YES) would be used, and a block of PT entries would appear because the records would be kept in the RPG buffer until the block is full.

21. Change the CL program ITMPCSC as follows (the new statements are shown with an asterisk):

```
     PGM
     DCL &USER *CHAR LEN(10)
     RTVJOBA USER(&USER)
*    STRCMTCTL LCKLVL(*CHG)
     CALL ITMPCS PARM(&USER)
*    MONMSG MSGID(RPG9001) EXEC(ROLLBACK)
*    ENDCMTCTL
     ENDPGM
```

The STRCMTCTL command sets up the commitment control environment. The LCKLVL word specifies that records read for update but not updated can be released during the transaction. The MONMSG command handles any RPG escape messages and performs a ROLLBACK in case the RPG program abnormally ends. The ENDCMTCTL command ends the commit control environment.

22. Delete the existing ITMPCSC program and create it again.
23. Change the RPG program to remove the comment symbols at statements 2.00, 4.00, 46.00, and 65.00. The source is now ready for use with commitment control.
24. Delete the existing ITMPCS program and create it again. The program is now ready to operate under commitment control.
25. Type the command CALL ITMPCSC and the following transactions:

Quantity	Item
7	AA
8	BB

26. Use System Request and request the option to display the current job. When the Display Job display appears, select option 16 to request the display of the commitment control status.

    Note the values on the display. There should be two commits because two commit statements were run in the program.
27. Press F9 to see a list of the files under commitment control and the amount of activity for each file.
28. Return to the program and end it by pressing F3.
29. Enter DSPJRN JRNTEST and note the entries for the files and the special journal entries for commitment control:
    - BC: STRCMTCTL command occurred.
    - SC: Start commit cycle. This occurs whenever the first database operation in the transaction causes a record to be locked as part of commitment control.
    - CM: Commit operation has occurred.
    - EC: ENDCMTCTL command occurred.

    Note how the commitment control before- and after-images (UB and UP types) automatically occur even though you had originally requested IMAGES(*AFTER) for the journal.
30. Type the command CALL ITMPCSC and the following transactions:

Quantity	Item
12	AA
100	CC (This is the condition to simulate the need for an application use of rollback. The CC record in the ITMP file, which was updated by RPG statement 40.00 is rolled back.)

31. Press F4 to determine the last transaction entered.

    The last committed transaction is the entry for item AA.
32. Use System Request and request the Display Current Job option. When the Display Job display appears, request the display of the commitment control status.

    Note the values on the display and how they have been changed by the rollback.
33. Return to the program.
34. Return to the basic prompt display and end the program by pressing F3.
35. Type the command DSPJRN JRNTEST.

    Note the additional entries that appear in the journal for the use of the rollback entry (RB entry). When the ITMP record is rolled back, three entries are placed in the journal. This is because any change to the database file under commitment control produces a before (BR) and after (UR) entry.
36. Display the UB, UP, BR, and UR records and use option 5 to display the full entries. Since the *Quantity* field is in packed decimal, use F11 to request a hex display. Note the following:
    - The on-hand value of the ITMP record in the UB record
    - How the on-hand value is reduced by the UP record
    - How the BR record is the same as the UP record

- How the UR record returns the value as originally displayed for the UB record

The last entry is the RB entry for the end of the rollback.

37. Type the command CALL ITMPCSC, press Enter, and press F4. Note the last transaction entered.
38. Type the following transactions:

Quantity	Item
13	AA
101	CC (This is the condition to simulate an unexpected error condition, which causes the program to end. The simulation occurs by dividing a field by 0. The program will display an inquiry message or end, depending on the setting of the job attribute INQMSGRPY. If the inquiry message appears, enter C to end the program. Because the CL program was changed to monitor for RPG program errors, the second inquiry which occurred does not occur.)

39. Type the command DSPJRN JRNTEST.

    The same type of rollback handling has occurred, but this time the rollback was caused by the EXEC parameter of the MONMSG command in the CL program instead of the RPG program. Display the two RB entries to see which program caused them.

40. Type the command WRKJOB and write down the fully qualified job name to be used later.
41. Type the command CALL ITMPCSC and enter the following transaction:

Quantity	Item
14	AA
102	CC (The RPG DSPLY operation should occur to the external message queue. Use the System Request key and select option 1 on the system request menu to transfer to a secondary job.)

42. Sign on to the second job and reestablish your environment.
43. Type the command ENDJOB and specify the fully qualified job name identified earlier and OPTION(*IMMED). This simulates an abnormal job or system end.
44. Wait about 30 seconds, type the command CALL ITMPCSC and press F4.

    Note the last committed transaction. It should be the AA item entered earlier.

45. Return to the basic prompt display and end the program by pressing F3.
46. Type the command DSPJRN JRNTEST.

    The same type of rollback handling has occurred, but this time the rollback was caused by the system instead of one of the programs. The RB entry was written by the program QWTPITPP, which is the work management abnormal end program.

You have now used the basic functions of commitment control. You can proceed with commitment control on your applications or try some of the other functions such as:

- Using a notify object
- Locking records that are only read with LCKLVL(*ALL)
- Locking multiple records in the same file with LCKLVL(*ALL)

Figure 4-22 on page 4-58 shows the DDS for the display file.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A          R PROMPT
  2.00      A                                       CA03(93 'End of program')
  3 00      A                                       CA04(94 'Review last')
  4 00      A                                       SETOFF(64 'No rcd to rvw')
  5 00      A                                     1  2'INVENTORY TRANSACTIONS'
  6.00      A                                     3  2'Quantity'
  7.00      A            QTY           5  0I     +1
  8.00      A  61                                   ERRMSG('Invalid +
  9.00      A                                       quantity' 61)
 10.00      A                                    +5'ITEM'
 11.00      A            ITEM          2    I   +1
 12.00      A  62                                   ERRMSG('Invalid +
 13.00      A                                       Item number' 62)
 14.00      A  63                                   ERRMSG('Rollback +
 15.00      A                                       occurred' 63)
 16.00      A  64                                24  2'CF4 was pressed and +
 17.00      A                                       there are no +
 18.00      A                                       transactions for +
 19.00      A                                       this user'
 20.00      A                                       DSPATR(HI)
 21.00      A                                   23  2'CF4 Review last +
 22.00      A                                       transaction'
 23.00      A          R REVW
 24.00      A                                     1  2'INVENTORY TRANSACTIONS'
 25.00      A                                    +5'REVIEW LAST TRANSACTION'
 26.00      A                                     3  2'Quantity'
 27.00      A            QTY           5  0     +1EDTCDE(Z)
 28.00      A                                    +5'Item'
 29.00      A            ITEM          2       +1
                                                                    RSLS826-1
```

Figure 4-22. DDS for the Display File

Figure 4-23 on page 4-59 illustrates the logic flow for the practice problem.

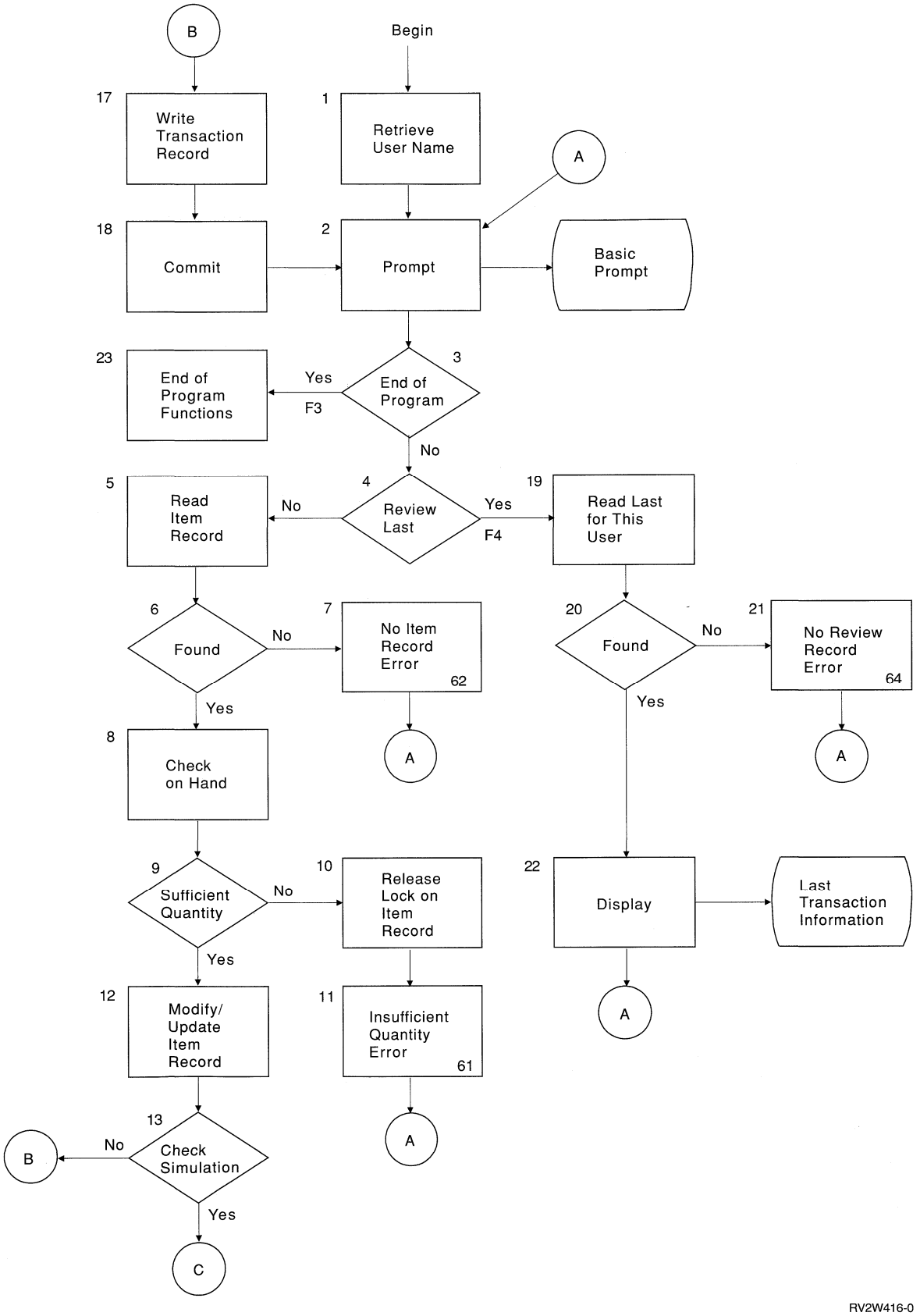


Figure 4-23 (Part 1 of 2). Logic Flow of the Practice Problem

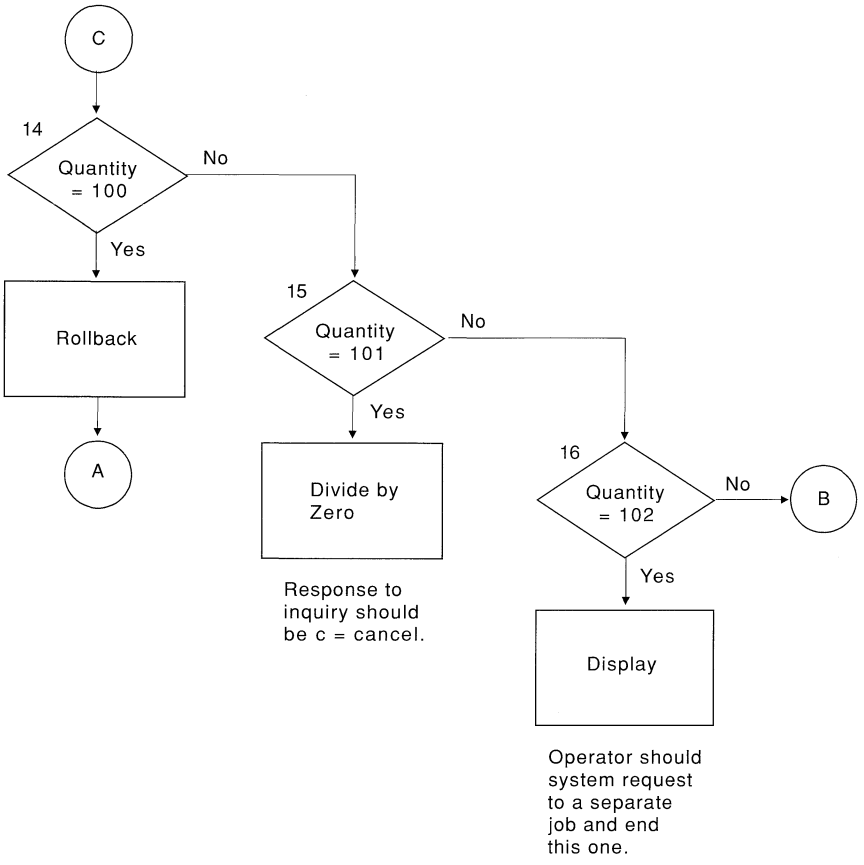


Figure 4-23 (Part 2 of 2). Logic Flow of the Practice Problem

Steps Associated with Logic Flow

The following steps are associated with the logic flow of the practice problem.

1. Retrieve the user name that is passed in as a parameter. This is used to write to the TRNP file and also used to retrieve the last transaction entered by each operator. This application assumes unique user names for operators.
2. Prompt for the basic display using the format name PROMPT.
3. If F3 is pressed, start an end of program function.
4. If F4 is pressed, start a routine to access the last transaction entered by the operator.
5. Read the item record using the field *ITEM*. Since the file is an update file, this request locks the record.
6. Check for a not found condition in the file ITMP.
7. If no ITMP record exists, set on indicator 62 to cause the error message and return to step 2.
8. Subtract the quantity requested (QTY) from the on hand balance (ONHAND) into a work area.
9. Check to see if sufficient quantity exists to meet the request.
10. If insufficient quantity exists, release the lock on the record in the ITMP file. This step is needed because of insufficient quantity.
11. Set on indicator 61 to signal an insufficient quantity display error message and return to step 2.
12. Change the ONHAND field for the new balance and update the ITMR record.
13. Check for special entry in the ITEM field that can be used to simulate conditions where ROLLBACK is required.

14. Check for QTY=100. Issue a ROLLBACK operation. This simulates a condition where the program senses a need for rollback.
15. Check for QTY=101. Cause an exception in the program that will produce an inquiry message. Use divide by zero for this function. The operator should enter C to cancel the program unless the job description INQMSGRPH option provides an automatic reply. This simulates a condition where an unexpected error has occurred and the operator cancels the program.
16. Check for QTY=102. Issue a display with inquiry operation. This stops the program at this step and allows the use of the System Request key to get to a different job. Cancel the updating job. This simulates a condition where an abnormal job or system end has occurred in the middle of a commit boundary.
17. Write the transaction record to TRNP.
18. Commit the records for the transaction and return to step 2 on page 4-60.
19. Read the first record on the access path for file TRNL, using USER as the key. Since this file is in LIFO sequence, this will be the last transaction record entered by this user.
20. Check for a record not found condition in the TRNL file that would be caused if the file does not contain entries for this user.
21. If there is no record for this user, set on indicator 64 to cause an error message and return to step 2 on page 4-60.
22. Display the last transaction entered for this user. This information can be used if the operator forgets what was previously entered or when the transaction is restarted. When the operator responds, return to step 2 on page 4-60.
23. Perform any end of program functions.

Part 2. Save-While-Active Function

Chapter 5. Save-While-Active Function

The save-while-active function provided on the AS/400 system allows you to modify objects while they are being saved. In contrast, other save functions provided on the system allow no access to the objects as they are being saved or only allow the objects to be read as they are being saved.

The save-while-active function can be used along with your other backup and recovery procedures to reduce or eliminate your outage for particular save operations. This chapter provides information about the save-while-active function and how it can be added to your current backup and recovery strategy.

Attention

The following information applies to most objects types. However, it only applies to documents and folders where specifically noted.

Overview of the Save-While-Active Function

You can request the save-while-active function by specifying the save-while-active (SAVACT) parameter on the save commands. The default is *NO. The SAVACT parameter can be specified on the following commands.

SAVLIB	Save Library
SAVOBJ	Save Object
SAVCHGOBJ	Save Changed Object
SAVDLO	Save Document Library Objects

The system performs the save-while-active function by maintaining an image of the object being saved as it existed at a single point in time. As the object is being changed by an application during the save, the system maintains an original copy of the pages of the objects that are being changed. In virtual storage, a **page** is a fixed-length block that has a virtual address and is transferred between main storage and auxiliary storage. It can be viewed as if the system is maintaining two images of an object as it is being saved: one image that contains the updates to the object that normal system activity works with, and a second image that is an image of the object from a single point in time that only the save-while-active job is using to save the object to the media. The system does not maintain two complete images of the object being saved. It only maintains two images for the pages of the objects that are being changed as the save is being performed.

The **checkpoint image** of an object is the image of the object at a single point in time. The **checkpoint** for an object is the instant in time that the checkpoint image of the object is taken. Checkpoint processing is part of the work that is performed by the system during the ***save preprocessing*** phase of a save-while-active operation. Other work that is performed by the system during save preprocessing includes determining which objects are to be saved and locking those objects to be saved. The ***save post-processing*** phase of a save operation includes unlocking the objects that were saved.

The time that the save request is started is referred to as the ***save time*** for the object. The time that the object reaches a checkpoint during the save operation is referred to as the ***save active time*** for the object.

Figure 5-1 on page 5-2 gives an overview of the terms and the processing that is performed for an object that is being saved with the save-while-active function.

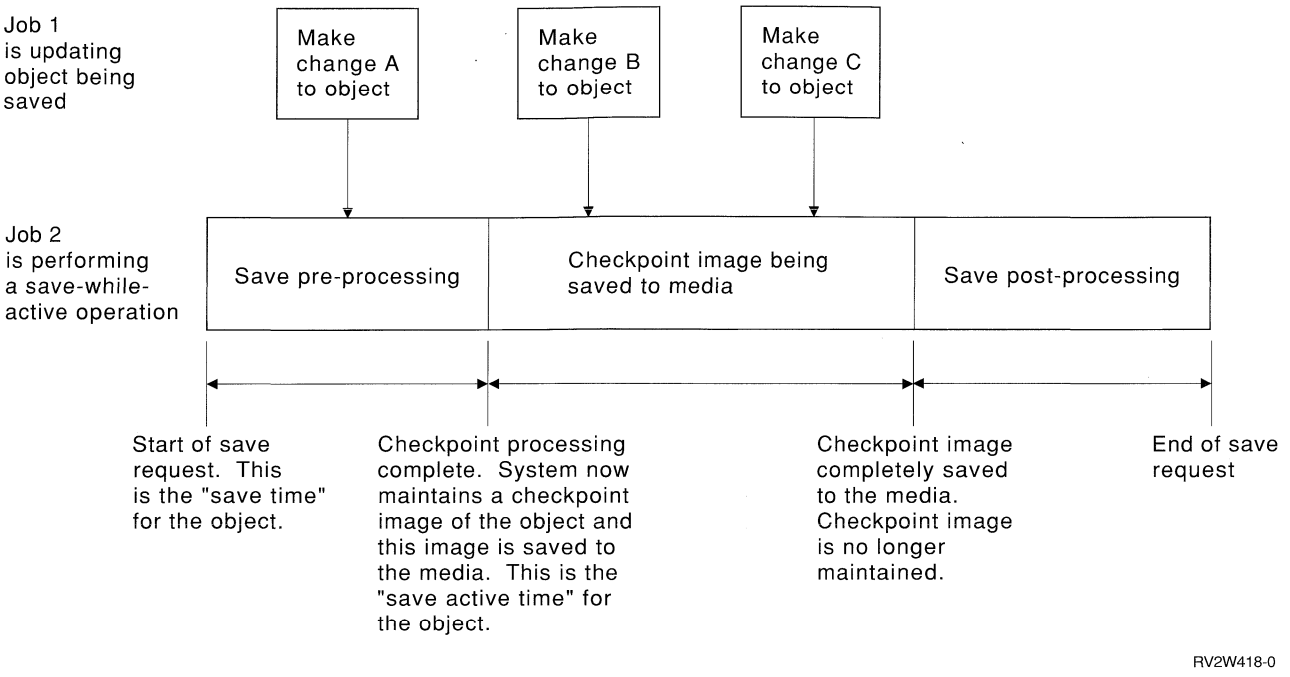


Figure 5-1. Save-while-active processing for an object

In Figure 5-1, Job 1 is making changes to an object that Job 2 is saving with the save-while-active function. Change A is saved to the media because that change is made before checkpoint processing is complete. Changes B and C are not saved to the media because those changes were made after the object reached the checkpoint, but they exist for the object on the system.

The amount of time it takes to perform backup procedures is referred to as the ***save window***. The save window for a save-while-active operation can be substantially reduced if applications are ended until the checkpoint processing is complete for the save-while-active operation. The amount of time it takes the system to establish the checkpoints varies depending upon several factors. In general, this time should be much less than the amount of time it takes to perform a dedicated save of the objects.

Using the Save-While-Active Function

The following topics provide information about how the save-while-active function can be used in your save strategy to reduce the time the system is not available during save operations.

Reduce Save Outage: If the requirement is to just reduce your outage for particular save operations, you can end the applications that make changes to the objects being saved until the system has established a checkpoint for each of those objects. Then, *no additional recovery procedures* are necessary when restoring the objects from the save-while-active media. *Therefore, the recommended method for performing a save-while-active request is to end the applications and then to start the applications again after the checkpoint processing for all application-dependent objects has completed.* Examples of application-dependent objects are data areas and physical files used by applications.

A message is sent after checkpoint processing is completed for all objects within a particular library or for all libraries in the save request. The applications can be started again when all application-dependent objects have reached a checkpoint. The checkpoint images of the objects that are saved to the media then appear as if a dedicated save was performed at the time the applications were ended.

Assume in Figure 5-1, that you end Job 1 when it is time to start the save-while-active procedure. After checkpoint processing is complete, Job 1 can be started again. Change A has not yet happened because Job 1 was not running. After Job

1 is started again, changes A, B, and C can be made while the checkpoint image of the object is being saved to the media. Changes A, B, and C, however, do not appear on the save media.

This method does not eliminate your save window, but can substantially reduce it. If you are saving objects from multiple libraries and a common application-dependency that spans the libraries exists, you should wait until checkpoint processing has completed for the all the libraries in the save request before starting the applications.

Eliminate Save Outage: The save-while-active function can eliminate your outage for particular save operations. However, you may have more complex and potentially longer recovery procedures after restoring objects from the save-while-active media.

To understand the need for the potentially more complex recovery procedures, you need to understand application boundaries. For save-while-active purposes, an **application boundary** is defined to be a point in time when all of the objects that a particular application is dependent upon are at a consistent state in relationship to each other and the objects are also in a state where the application can be started or started again. Even though particular objects reach a checkpoint together within a single library, the system cannot ensure that application boundaries are being maintained. The responsibility of maintaining application boundaries is left up to the user of the save-while-active function.

If applications are allowed to update the objects being saved as the objects are reaching a checkpoint, the system cannot determine if the images of the objects that are saved to the media are at application boundaries. Because these application boundaries must be maintained by the user, additional recovery procedures may need to be defined and performed after restoring objects from the save-while-active media. These additional recovery procedures bring the objects to a consistent state in relationship to each other after the restore operation is complete. The exact steps required for these recovery procedures must be determined by the user, and should be determined at the time the objects are being saved. For examples of recovery procedures, see "Restore Recovery Procedures" on page 5-10 and "Recommended Restore Recovery Procedures" on page 5-20.

Object Locking Rules

The system uses object locking rules that are less restrictive for save-while-active requests than those used for other save operations. These object locking rules allow application users to perform update operations and use most object-level commands after the checkpoint processing has been performed for the objects being saved. Generally, a share no update (*SHRNUP) lock is kept on the objects through the checkpoint processing. After the checkpoints have been established, the locks are then lowered to a shared for read (*SHRRD) lock for the remainder of the save operation.

These locking rules can conflict with object-level lock types of exclusive allow read (*EXCLRD), exclusive (*EXCL), and share update (*SHRUPD) that can be acquired by user applications and some object-level system commands. User applications that acquire an *EXCLRD object-level lock or use system commands that require an *EXCLRD object-level lock generally conflict with a save-while-active operation until the checkpoint processing is complete for the objects. Use the recommended method of ending the applications that make changes to the objects being saved with the save-while-active function until the checkpoint processing is completed. This eliminates any locking conflicts during the checkpoint processing.

User applications that acquire an *EXCL object-level lock or use system commands that require an *EXCL object-level lock cannot run in conjunction with a save-while-active operation, unless the application is changed to not require the *EXCL object-level lock.

These locking rules pertain to object-level locks and not database record-level locks. The opening and closing of database file members and any record-level I/O operations to database file members are allowed during any phase of the save-while-active operation.

In general, the system prevents the following operations for objects being saved during the entire save-while-active request:

- The object cannot be deleted.
- The object cannot be renamed.
- The object cannot be moved to a different library or folder.
- The ownership of the object cannot be changed.

See "Save-While-Active Object Locking" on page 5-25 for a complete list of object-level operations available during save-while-active operations.

Checkpoint Processing

Checkpoint processing occurs after the system determines exactly which objects are to be saved for a particular library. If the save-while-active request is for multiple libraries, then the checkpoint processing is performed for all libraries in the save request.

Checkpoint processing does not require that two complete copies be maintained for the objects being saved. The system only maintains two copies for the pages of the object that is being changed as the save is being performed. The more pages that are changed for an object during the save-while-active request, the greater the storage requirement for the object. Performance decreases slightly for the first update to a page after checkpoint processing to create the checkpoint image of the page is completed. The performance impact varies depending on the disk type, available disk storage, and processor model. Further updates to the same changed page do not require any additional processing, with respect to the checkpoint version of the page.

Figure 5-2 on page 5-5 shows how a checkpoint image of an object is maintained during a save-while-active operation. The shaded parts of the diagram represent the checkpoint version of the object.

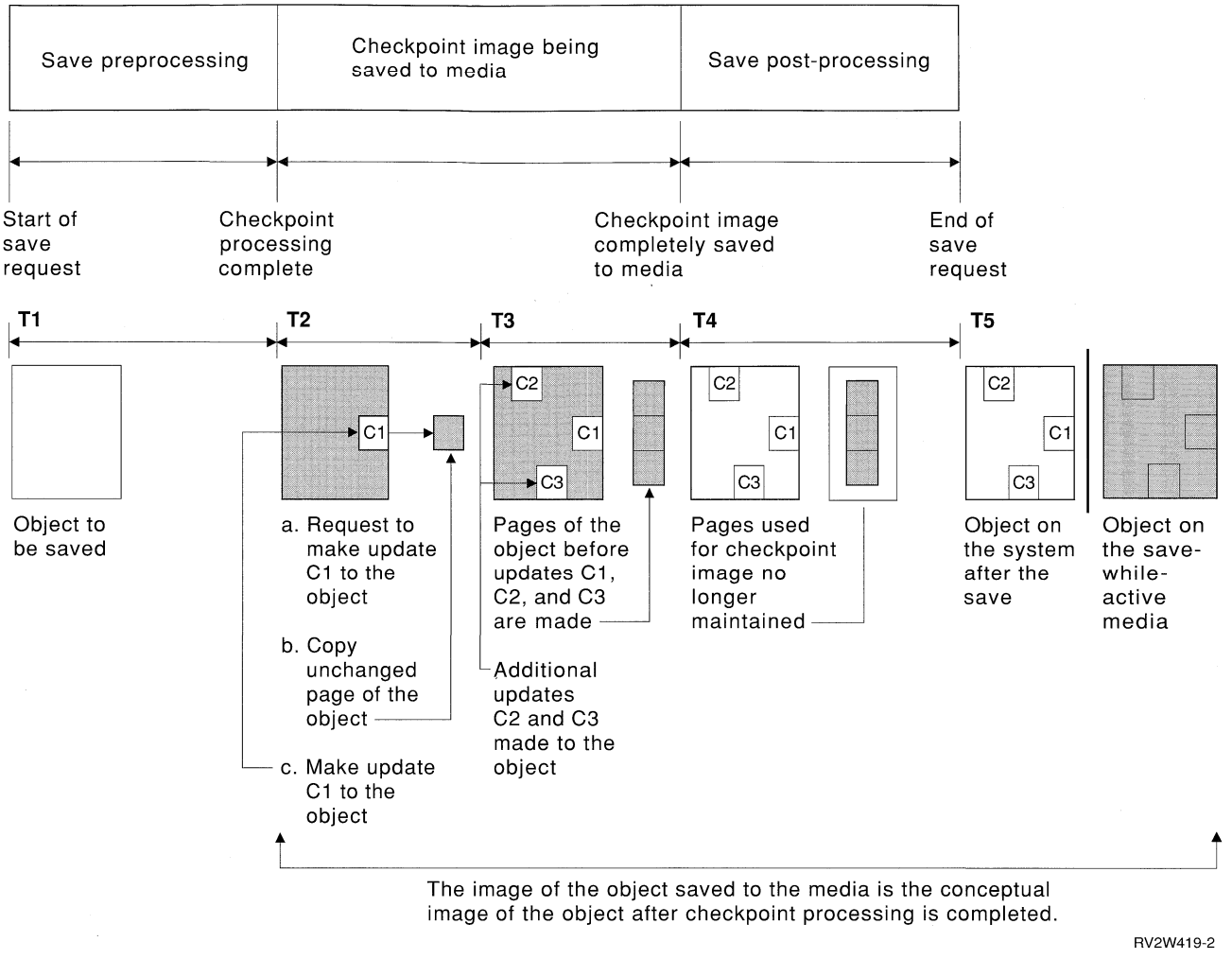


Figure 5-2. System Management of Updates to Objects after Checkpoint Processing is Complete

In Figure 5-2:

1. Time T1 is the save preprocessing phase of the save-while-active operation. At the end of time T1, the object has reached a checkpoint.
2. Time T2 shows an update to the object, referred to as C1, while the object is being saved to the media.
   a. A request is made to make update C1.
   b. A copy of the original page is made first.
   c. The change is made to the object.

   The original page copied is then part of the checkpoint image for the object.
3. Time T3 shows two additional changes, C2 and C3, have been made to the object. Note that any additional change requests that are made to the pages of the object already changed for C1, C2, or C3 do not require any additional processing with respect to the checkpoint image of the object. At the end of time T3, the object has been completely saved to the media.
4. Time T4 shows that the copied pages for the checkpoint image of the object are no longer maintained because they are no longer needed.
5. Time T5 shows the object on the system has the C1, C2, and C3 changes, but the copy, or image, of the object saved to the media does not contain those changes.

| Three options are provided that allow the user to specify the type of save-while-active checkpoint processing the system performs:

| 1. synchronize libraries
2. library
3. system-defined

Synchronize Libraries Checkpoint Processing: This processing is performed when SAVACT(*SYNCLIB) is specified on the save command. This option allows the user to request all libraries in the save request to reach a checkpoint together.

The time it takes to establish a checkpoint for a library is much shorter than the time it takes to save the library to tape. Therefore, establishing a checkpoint for multiple libraries at the beginning of the save operation significantly reduces the time the system is not available during checkpoint processing and save-while-active operations.

If you have an application that spans multiple libraries, the recovery is less complex. The value *SYNCLIB is recommended if you have applications that span multiple libraries. However, locks on the objects in all the libraries are held for a longer period of time because all of the objects are preprocessed before the first object is saved to the media. The amount of time depends of the size of the save and how active the system is at the time of the save.

Note: SAVACT(*SYNCLIB) can be specified even if only one library is being saved. Specifying SAVACT(*SYNCLIB) normally reaches the checkpoint quicker than SAVACT(*LIB).

Library checkpoint processing: This processing is performed when SAVACT(*LIB) is specified on the save command. This option allows the user to request that all objects in a single library reach a checkpoint together. This ensures that all objects in the library are saved in a consistent state in relationship to each other.

SAVACT(*LIB) allows users, who have database and non-database application-dependent objects in the same library, to restore using the save-while-active media and have the database and non-database objects restored back to a consistent state in relationship to each other.

System-defined checkpoint processing: This processing is performed when SAVACT(*SYSDFN) is specified on the save command. By specifying this option, the user requests that the system determines how the various objects being saved from a particular library are to be grouped for checkpoint processing.

Note: For the SAVDLO command, SAVACT(*YES) specifies system-define checkpoint processing.

Specifying SAVACT(*SYSDFN) may result in the system grouping objects within a single library into multiple checkpoint steps. This option may allow the system to perform better than SAVACT(*LIB) does, but not all objects in the library reach a checkpoint together. Therefore, using SAVACT(*SYSDFN) may not save all of the objects within the library at a consistent state in relationship to each other and may require more complex restore recovery procedures.

For database objects, SAVACT(*SYSDFN) does ensure that certain files with logical dependencies within the same library reach a checkpoint together. To better understand this point, you need to understand a database network. A **database network** consists of a set of related objects. For example, all logical files built over a single physical file make up a simple network. These simple networks can then be grouped together by a common logical file that is built over the physical files from two or more simple networks. Simple networks are continually grouped until no logical file exists that can group two smaller networks together. The final result is a database network.

Note: Library QUSRSYS can be considered as a part of a database network because it contains many objects used by applications and OfficeVision/400 that are placed under commitment control.

Database files within a database network in a single library always reach a checkpoint together. In addition, database files in the same library that are journaled to the same journal always reach a checkpoint together. Therefore, database networks in a single library that have files journaled to different journals also reach a checkpoint together.

Figure 5-3 on page 5-7 shows how the system ensures that certain database files in the save library reach a checkpoint together when SAVACT(*SYSDFN) is specified. All the objects shown in Figure 5-3 on page 5-7 reside in the same library. The objects with names starting with PF are physical files; LF are logical files.

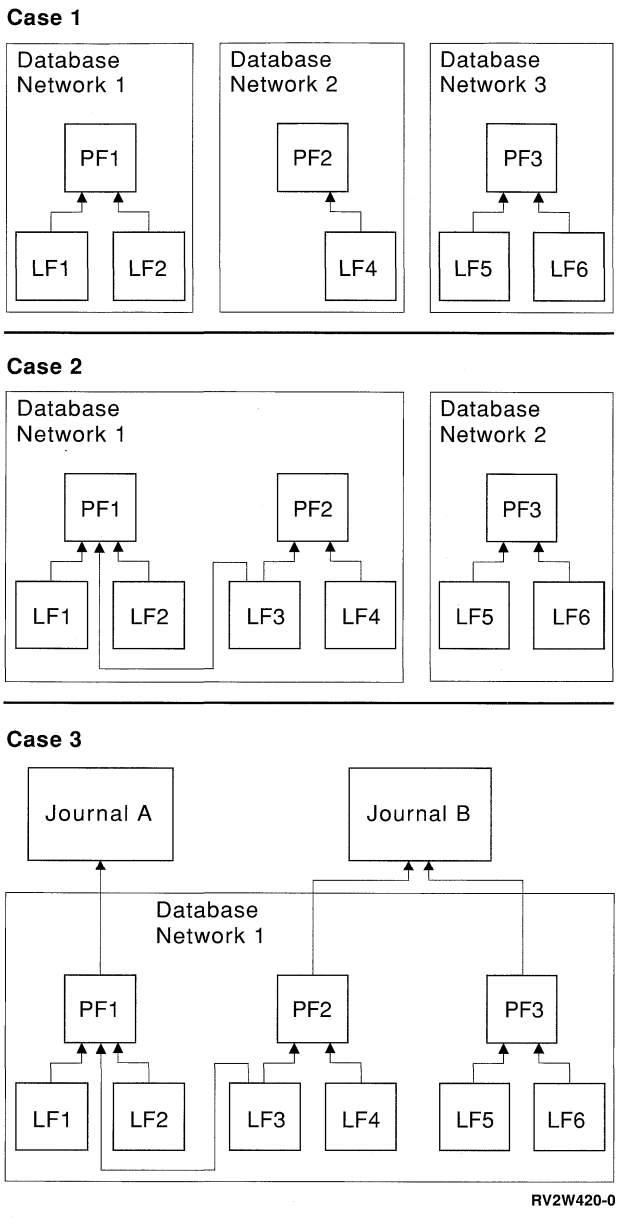


*Figure 5-3. Database Network Examples for SAVACT(*SYSDFN)*

In Figure 5-3:

- Case 1 shows the files being grouped into three separate database networks, with each database network reaching a checkpoint at a different point in time.
- Case 2 shows the system grouping the files into two separate database networks. For Case 2, logical file LF3 requires that physical files PF1 and PF2 and all of the logical files built over them reach a checkpoint together.
- Case 3 shows the system grouping all of the files into the same database network, and therefore, all of the files reach a checkpoint at the same point in time. For Case 3, journal B requires that physical files PF2 and PF3 reach a checkpoint together, and logical file LF3 requires that physical files PF1 and PF2 reach a checkpoint together.

For Case 3, notice that neither the journal, nor the attached journal receivers (they are not shown), are included in the database network of objects to reach a checkpoint together. Journals and journal receivers reach the same checkpoint as the files that are being journaled. The Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG) commands can still be used after restoring the files from the save-while-active media. Even though the journal and attached journal receiver do not have to reach the same checkpoint as the files being journaled, the attached journal receiver for each journal should be saved as part of the save request for the files or in a separate save request after the files have been saved.

When specifying SAVACT(*SYSDFN), other object types such as data areas do not reach the same checkpoint as any of the database files. Therefore, if your application has dependencies on database files and other objects such as data areas, those objects will reach a checkpoint at different points in time. If changes are allowed to these application-dependent objects during checkpoint processing, complex recovery procedures may be required after restoring the objects from the save-while-active media.

Note: SAVACT(*SYSDFN) should only be used if either of the following are true:

- All applications making updates to the objects being saved are ended until checkpoint processing is complete.
- All application-dependent objects reside within a single library and all application-dependent objects are database files that are journaled.

  In this case, the Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG) commands can be used to bring the saved objects to a consistent state in relationship to each other.

  Otherwise, using this option for other save situations may lead to very complicated recovery procedures after restoring from the save-while-active media.

Waiting for Objects during Checkpoint Processing: The amount of time that the save-while-active function waits for objects during checkpoint processing can be controlled by specifying a value for the SAVACTWAIT parameter. This parameter allows you to specify how long to wait for each object during checkpoint processing. Depending on the object type and what processing is currently active for an object, the system may actually wait multiple times for an object during the checkpoint processing. Therefore, the value specified on the SAVACTWAIT parameter is the time the save-while-active operations waits for an object each time.

Save-While-Active Timestamp Processing

Note: The following makes reference to the update history (UPDHST) parameter on the save commands. UPDHST(*NO) can only be specified if the device being used for the save operation is a save file (DEV(*SAVF)).

The date and time that a save operation is performed for a particular object is recorded by the system if UPDHST(*YES) is specified on the save command. This timestamp is taken early during the save preprocessing phase and identifies when the save operation was started for the object. This timestamp is referred to as the ***save time*** for the object. If multiple objects are being saved with the same save request and all of the objects reside in the same library, the save time will be the same for all of those saved objects. This timestamp is the *save date/time* that is displayed by the Display Object Description (DSPOBJD) command.

The save-while-active function introduces an additional timestamp that relates to save processing. This is referred to as the **save active time** for an object that is saved with the save-while-active function and identifies the time that the object reached the checkpoint. For objects that reach a checkpoint together, the save active time will be the same for all of those objects. This timestamp is the *save active date/time* that is displayed by the Display Object Description (DSPOBJD) command. As with the save time for an object, the save active time for an object is only updated if UPDHST(*YES) is specified on the save command that is requesting the save-while-active operation. Some objects do not require special save-while-active checkpoint processing. Therefore the save-while-active timestamp is the same as the save time for other objects. Examples of this are object types *JOBQ and *OUTQ that have only their descriptions saved, not their contents.

For physical file members, note that the *last save date/time* information that is displayed by the Display File Description (DSPFD) command could identify the last save time or the last save active time, depending on which type of save operation was last performed for each of the members.

Figure 5-1 on page 5-2 shows when the timestamp is recorded for objects during the various stages of a save-while-active operation.

Note: If all applications that are making changes to the objects being saved with the save-while-active function are ended until checkpoint processing is completed, the following restore recovery procedure considerations do not apply.

The save active time for an object can be useful when trying to determine what restore recovery procedures may be necessary after restoring objects from the save-while-active media. All of the changes made to the object before the save active timestamp will be present for the object on the save-while-active media. All of the changes made to the object after the save active timestamp will not be present for the object on the save-while-active media.

Restore Recovery Procedures Considerations: For journaled members that are saved with the save-while-active function, the start of save (journal code F, type SS) and the member saved (journal code F, type MS) journal entries in the journal contain both the save time and save active time in the entry specific data portion of those entries. The SS journal entry type identifies when the journaled file member reached the checkpoint. All journal entries after the SS journal entry for a journaled file member will not be reflected in the data that is saved to the media for a save-while-active operation. This information may be useful when determining what recovery procedures are necessary after restoring journaled files from the save-while-active media.

See Chapter 2, "Journal Management" for more information about the journaling function.

Save-While-Active Commitment Control Processing

Note

If all applications that are making changes to the objects being saved with the save-while-active function are ended until checkpoint processing is completed, this entire section does not apply.

Commitment Control Considerations: If an object is being updated under commitment control during the checkpoint processing phase of a save-while-active request, the system ensures that the object will be saved to the media at a commitment boundary. All objects that have reached a checkpoint together will be saved to the media at the same common commitment boundary. See "Checkpoint Processing" on page 5-4 for more information on how objects for a particular library may be grouped together with respect to checkpoint processing.

The system ensures that objects are saved to the media at a commitment boundary by performing the following steps during the save preprocessing phase of a save-while-active request:

- If the job performing the save-while-active request is not currently at a commitment boundary, the save request ends without saving any objects. This processing is the same for any save request.
- For a group of objects that are reaching a checkpoint together, if updates are in progress for any of those objects, the system delays reaching a checkpoint for those objects until all such transactions reach a commitment boundary. The system waits the amount of time specified on the SAVACTWAIT parameter for these transactions to reach a commitment boundary. If, after waiting the specified time, uncommitted transactions still exist, the save request is ended for that particular library.

  The system identifies which jobs have commitment definitions that are not currently at a commitment boundary and are delaying the checkpoint processing for a save-while-active request. After the checkpoint processing for a group of objects has been delayed approximately 30 seconds because of uncommitted transactions, the system sends a CPI8365 message to the QSYSOPR message queue for each job that is delaying the save-while-active request. Upon receiving these messages, you can then take the appropriate actions to bring all commitment definitions for those jobs to a commitment boundary.

  When no more commitment definitions are delaying the save-while-active job, the save-while-active job completes the checkpoint processing for the objects. After checkpoint processing is completed, changes are allowed for those objects under commitment control.

  It is possible for a save-while-active request to be delayed if a commitment definition has uncommitted changes even though the uncommitted changes are not for any database file being saved by the save-while-active request. This situation can occur if any of the database files are being journaled to the same journal as the commitment definition is using for unrelated, uncommitted changes.
- If an application is performing a read-for-update operation but no changes have been made, the application is considered to have started a commit cycle. Before V2R3M0, this would prevent a checkpoint from being established because of uncommitted transactions. In V2R3M0, the system allows a checkpoint to be established in the middle of a commit cycle as long as no changes have been made. Checkpoint processing is not stopped if the application is performing only a read-for-update operation.
- The system temporarily delays a job that has all commitment definitions at a commitment boundary, when additional changes that might be made under commitment control have the potential to make a change to an object that is reaching a checkpoint. The job is held at that commitment boundary until either the objects reach a checkpoint or the checkpoint processing for the save-while-active request has exceeded the time specified on the SAVACTWAIT parameter. During the time that such a job is delayed at a commitment boundary, CMTW is displayed as the job status when using the Work Active Job (WRKACTJOB) command.

Object-Level Resource Considerations: In addition to the above considerations, object-level resource changes cannot be made under commitment control if checkpoint processing is being performed for objects in the object-level resource library for a save-while-active request. Object-level resource changes cannot be made if either of the following are true:

- The commitment definition is at a commitment boundary.
- Only record-level changes have been made in the uncommitted transaction.

For this situation, the change is delayed until checkpoint processing is complete for the library. After a delay of approximately 60 seconds, inquiry message CPA8351 is sent to the user. The inquiry message allows the user to continue to wait for the checkpoint processing to complete or to cancel the request for the object-level resource. If the job is a batch job, inquiry message CPA8351 is sent to the QSYSOPR message queue.

Application Programming Interface Resource Considerations: In addition to the commitment control considerations described earlier, resources cannot be placed under commitment control if checkpoint processing is being performed for any save-while-active request and either of the following are true:

- With the Add Commitment Resource API (QTNADDCR program), the commitment definition is at a commitment boundary.
- Only record-level changes have been made in the uncommitted transaction.

For this situation, the add is delayed until checkpoint processing is complete for the save-while-active request. After a delay of approximately 60 seconds, inquiry message CPA8351 is sent to the user. The inquiry message allows the user to continue to wait for the checkpoint processing to complete or to cancel the request for the API resource. If the job is a batch job, inquiry message CPA8351 is sent to the QSYSOPR message queue.

If a commitment definition has an API commitment resource associated with it, and checkpoint processing is being performed for any save-while-active request, then the job performing a commit or rollback operation for the commitment definition is delayed immediately after the commit or rollback has been performed. The job is delayed until the checkpoint processing is completed for the save-while-active request. After the checkpoint processing is complete, control is returned back to the job issuing the commit or rollback. This delay is necessary because a commitment definition with an API commitment resource is only considered to be at a commitment boundary immediately after a commit or rollback operation but before control is returned to the user program. Once the commit or rollback operation returns control back to the user program, the commitment definition is no longer considered to be at a commitment boundary.

See Chapter 4, "Commitment Control" on page 4-1 for more information about the commitment control function.

Restore Recovery Procedures

Note

If all applications that are making changes to the objects being saved with the save-while-active function are ended until checkpoint processing is completed, the following restore recovery procedure ***does not*** apply.

In general, the system cannot preserve application boundaries because they are defined by the application. It is left up to the user of the save-while-active function to provide for any of the appropriate restore recovery procedures.

However, the system does ensure that a partial update to an individual object will not be saved by the save-while-active function. For example, if a record is being updated during the checkpoint processing phase of the save-while-active operation, the system ensures that the object is not saved to the media with part of the record updated. Either the entire update is, or is not, present in the file member saved to the media.

The following section discusses some of the considerations for save-while-active restore recovery procedures. These additional recovery procedures are needed to bring the objects to a consistent state in relationship to each other after the restore operation is completed. The exact steps

required for these recovery procedures must be determined by the user at the time the objects are being **saved**. The restore recovery procedures must be performed after the objects from the save-while-active media are restored, but before the objects are used by any application.

Considerations for Restore Recovery Procedures

The following should be considered when determining what recovery procedures may be necessary after restoring objects from save-while-active media. Restore recovery procedures need to be considered if applications that update the objects being saved with the save-while-active function are running while checkpoint processing is being performed for those objects. Depending on your situation, consider the following:

- **When some application-dependent objects are not database files**.

  If applications are dependent upon objects such as data areas, then user-written recovery procedures may be necessary after restoring objects from the save-while-active media. The necessary recovery may be similar to the recovery necessary if these objects are being updated when the system abnormally ends.

  If all application-dependent objects reside in one library and all of the objects are saved with one save request, SAVACT(*LIB) can be specified to ensure all of the objects reach a checkpoint together and all of the objects are saved in a consistent state in relationship to each other. However, the checkpoint versions of the objects may not be at an application boundary. User-written recovery procedures may still be necessary to bring the objects to an application boundary.

  Note: If an application has dependent objects that reside in multiple libraries, you should use SAVACT(*SYNCLIB) to ensure that all application-dependent objects reach a checkpoint together.

  If any application-dependent objects are not journaled database files, then SAVACT(*SYSDFN) should not be used.

- **When some of the application-dependent objects reside in multiple libraries**.

  If application-dependent objects reside in multiple libraries, all of the objects cannot reach a checkpoint together unless SAVACT(*SYNCLIB) is used. If SAVACT(*SYNCLIB) is not used, the necessary recovery may be similar to the recovery necessary if these objects are being updated when the system ends abnormally.

- **When all of the application-dependent objects are database files and some of the files are journaled**.

  If all application-dependent objects are database files and all of the files are journaled, then the Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG) commands can be used as part of the recovery procedures to bring all of the files to an application boundary after restoring from the save-while-active media. An additional journal entry (journal code F, type SS) is sent in conjunction with the member saved journal entry (journal code F, type MS) for a journaled file member being saved with the save-while-active function to note when the journaled file member reached a checkpoint. If only some of the application-dependent files are journaled, then the APYJRNCHG and RMVJRNCHG commands can be used to recover those files, but user-written recovery procedures may still be necessary for the objects that are not journaled.

  If **all** application-dependent files are journaled, SAVACT(*SYSDFN) may perform better than SAVACT(*LIB). SAVACT(*SYSDFN) allows fewer objects to need to reach a checkpoint together. In either case, the APYJRNCHG and RMVJRNCHG commands can be used to bring the journaled files to a common application boundary after restoring from the save-while-active media.

  If all application-dependent files are journaled but reside in multiple libraries and SAVACT(*SYNCLIB) is not specified, then the recovery most likely includes applying or removing journaled changes so that all of the application-dependent files are brought to a consistent state in relationship to each other. Because the journaled objects reside in multiple libraries, all of the objects cannot reach a checkpoint together. The files are brought to a common application boundary. The APYJRNCHG or RMVJRNCHG operation.

Note: It is critical that the currently attached journal receiver be saved along with the files being journaled. If more than one journal is being used to journal the files, then all attached receivers must be saved. Include the request to save the receiver in the same save request as that for the journaled files or in a separate save request after the save of the journaled files. This save is necessary because the attached journal receiver will contain the entries that may be required by any apply or remove journaled changes operation that is part of the restore recovery when using the save-while-active media.

- **When all of the application-dependent objects are database files and all of the changes made to these files are made under commitment control.**

  Recovery procedures may not be necessary after restoring from the save-while-active media if all of the following are true:

  - All application-dependent objects are database files.
  - All of the changes made to these files are made under commitment control.
  - All of the files reside in the same library.

  The save-while-active function ensures that no partial transaction is saved to the media. Therefore, after restoring from the save-while-active media, the files will exist as they existed at the commitment boundary when checkpoint processing completed. However, the files being at a commitment boundary may not mean that they are at an application boundary.

  Likewise, if all changes are being made under commitment control but the files under commitment control reside in multiple libraries, then the system saves the files at commitment boundaries on a library-by-library basis. Database files that are in different libraries and that are being changed under commitment control may be at different commitment boundaries with respect to the application.

  Note: If SAVACT(*SYNCLIB) is used, all changes are being made under commitment for files that reside in multiple libraries. In this case, the system saves the files at one commitment boundary for all the libraries in the save request.

  For either of these cases, the APYJRNCHG or RMVJRNCHG command can be used to bring the files to a common application boundary after restoring from the save-while-active media.

- **When some application-dependent objects are not database files, but all changes made to those application-dependent objects are made under commitment control and all objects reside in the same library.**

  Recovery procedures may not be necessary after restoring from the save-while-active media if all of the following are true:

  - Not all application-dependent objects are database files.
  - All of the changes made to these objects are made under commitment control.
  - All of the objects reside in the same library.

  Additional recovery procedures are not necessary if a commitment boundary is also an application boundary.

  Though object-level changes can be made under commitment control and changes can be made using the Add Commitment Resource API (QTNADDCR program) notice that these types of resource changes cannot be applied or removed from the database with the APYJRNCHG or RMVJRNCHG command.

Save-While-Active in Your Backup and Recovery Strategy

Save-while-active operations can eliminate or greatly reduce scheduled down-time for save operations. The save-while-active function can be ideal for your daily save operations.

Notes:

1. If the system is in restricted state, the SAVACT parameter is ignored.
2. Using the save-while-active function for full save operations is not recommended because of the additional performance impact. Examples of full save operations would include SAVLIB LIB(*IBM) or SAVLIB LIB(*ALLUSR). SAVLIB LIB(*IBM) and SAVLIB LIB(*ALLUSR)

are not allowed when using SAVACT(*SYNCLIB).

For save requests that require the system to be in a restricted state, such as SAVLIB LIB(*NONSYS), the SAVACT parameter is ignored if specified on the save request and the save-while-active processing is not performed.

Note: Loading, applying, or removing programming temporary fixes should not be done when running a save-while-active operation.

Other procedures used in your backup and recovery strategy still apply and should be considered when reviewing your backup and recovery procedures. Using the save-while-active function in your daily save operations may place additional requirements on your restore recovery procedures and your disaster recovery plan.

Performance Considerations

Save-while-active operations can be run any time, although performance results may vary. Performance depends on available disk storage, system I/O activity, available main storage, and processor model.

For better performance during save-while-active operations, low system activity is recommended. A few interactive jobs or batch jobs that are primarily read-only are excellent examples of this type of activity that allows for better system performance during the save-while-active operation.

In general, save-while-active checkpoint processing is performed faster for a small number of larger objects than for a large number of smaller objects.

Major factors in the performance of the save-while-active function:

- Amount of processing unit cycles available for the workload and the save

  Be prepared for the save-while-active operation to take longer than a save operation on a restricted system. The change in the time required for the save operation to complete may be as little as 10 percent longer to four to five times longer or more, depending on the system resources available for the save operation.

- The auxiliary storage subsystem capabilities

  When evaluating the time period for a save-while-active operation, the activity in auxiliary storage should be less than 30 percent before adding the activity for the save operation. This is due to the heavy auxiliary storage activity that is added with the save-while-active operation.

- The pageable size of the machine pool

  Additional pages are required in the machine pool for the system to use during the save-while-active operation. Additionally, saving many small objects or file members places additional requirements on the pageable portion of the machine pool. The addition of 600KB to the machine pool should be considered a minimum. Additional memory may improve the response time and the save time.

  Additional megabytes of storage for the machine pool may help performance if saving thousands of small objects or file members (less than 50KB object sizes). Monitoring the machine pool for paging activity is recommended.

- Job priority and pool usage

  You must decide which jobs have priority: the save operation or the other activity on the system. To maintain the best response time for interactive jobs, it is recommended that the save operation be given a lower priority than the interactive jobs. Additionally, it is recommended that you separate the save operation from other work on your system by using a separate memory pool with a 4MB or more minimum pool size if thousands of objects or file members are being saved.

  Using *LIB or *SYNCLIB generally requires more memory pool pages to achieve better performance.

- Number and size of objects

  If many small objects or file members are being saved, the paging in the machine pool may increase. Paging in the machine pool should be monitored. Steps should be taken to minimize paging to maintain better overall system performance. This is true for normal saves and operations (including document save and restore operations).

- The characteristics of the jobs the save-while-active operation is competing with

  The active jobs during a save-while-active operation can effect both the response time and the duration of the save operation. Choosing a time period of low CPU utilization and low update activity helps minimize the impact to both response time and save time duration.

If the save-while-active operation is run at a time when users are updating document library objects (DLO), the save-while-active process may affect these users. When users are changing document library objects, they may notice a delay if the save-while-active operation is checkpoint processing the document library objects.

For example, an OfficeVision/400 user may be editing a document while a save-while-active operation is running. If the save-while active operation is performing checkpoint processing on the document and the OfficeVision/400 editor attempts to update the document with text the user just typed, the editor may wait until checkpoint processing is complete before it can make the update. If the save-while-active job is running at low priority, or on a busy system, the user's edit session may wait for an extended period of time.

OfficeVision/400 user functions are designed to wait up to 30 minutes for checkpoint processing to complete. This limit should be more than adequate to allow checkpoint processing to complete. Most functions involving document library objects can be interrupted using the System Request process during this time, if the user feels the wait has become too long.

If for any reason, the save-while-active operation has not completed checkpoint processing for the document library objects within 30 minutes, the user function (such as the edit session, not the save-while-active process) is ended abnormally to indicate there is a problem. The AS/400 system administrator should determine why the save-while-active process is taking such an excessive amount of time for the document library objects to reach a checkpoint (system too busy or priority too low). Take the appropriate action to correct the problem. This may require contacting your IBM representative.

Storage Considerations

Save-while-active operations require that at least 1MB of free main storage exist. Additional main storage can further minimize performance impacts.

Changing Your Backup and Recovery Strategy

After reviewing your applications and the considerations discussed previously, you should be able to make an informed decision about how to use save-while-active operations in your backup and recovery strategy. You may conclude one of the following:

- Your current save operations are adequate for your scheduled save windows.
- Critical application libraries are candidates for save-while-active processing.
- Critical application libraries are candidates but may require modification to minimize restore recovery procedures.
- Critical documents or folders are candidates.
- All application libraries are candidates because of a compressed save window.

If you decide to use save-while-active operations as part of your backup and recovery strategy, and applications that update the objects to be saved can be ended until checkpoint processing is complete, then additional recovery procedures will not be necessary after restoring objects using the save-while-active media. After checkpoint processing is complete (indicated by a message being sent for each library or one message if *SYNCLIB is used), the applications can be started again and the media produced by the save-while-active operation appears as if the save was performed while the system was in a restricted state.

However, if you use the save-while-active function and the applications that update the objects to be saved cannot be ended, then additional recovery procedures are needed after restoring the objects from the save-while-active media. These additional recovery procedures must ensure that the objects that the applications are dependent upon are at a consistent state in relationship to each other. Each of the following should be considered when determining these recovery procedures:

- Whether the objects that the applications are dependent upon consist entirely of database files, or they depend on other objects types such as data areas.
- Whether the objects that the applications are dependent upon are contained in a single library, or span multiple libraries.
- Whether the database files that the applications are dependent upon are journaled.
- Whether the changes made to objects by the applications are being made under commitment control.

See "Restore Recovery Procedures" on page 5-10 for more information on recovery procedures after restoring objects from save-while-active media.

Save-While-Active Commands

The following provides information about the save-while-active parameters on the SAVLIB, SAVOBJ, SAVCHGOBJ, and SAVDLO commands.

Parameters on the SAVLIB, SAVOBJ, and SAVCHGOBJ Commands

The following describes the parameters that can be specified on the SAVLIB, SAVOBJ, and SAVCHGOBJ commands:

SAVACT

The SAVACT parameter is used to specify whether an object can be updated while the object is being saved. The possible values are:

***NO:** No updates are allowed to the objects in the save request. Normal save processing is used. Data integrity is preserved with the maximum performance.

***LIB:** Objects in a library can be saved while they are in use by another job. All of the objects in a library reach a checkpoint together and are saved in a consistent state relative to each other.

Notes:

1. Only physical files with members have the same save active date and time timestamp.
2. Libraries with thousands of objects may be too large for this option.

If multiple libraries are specified on the save request, the checkpoint processing is performed individually for the objects within each specified library.

***SYNCLIB:** Objects in a library can be saved while they are in use by another job. All of the objects in all of the libraries in the save operation reach a checkpoint together and are saved in a consistent state in relationship to each other.

***SYSDFN:** Objects in a library can be saved while they are in use by another job. Objects in a library may reach checkpoints at different times and may not be in a consistent state in relationship to each other.

Note: Specifying this value eliminates some size restrictions and may enable a library to be saved that could not be saved with SAVACT(*LIB).

See "Restore Recovery Procedures" on page 5-10 for more information on recovery procedures after restoring objects from save-while-active media.

SAVACTWAIT

The SAVACTWAIT parameter is used to specify the amount of time the system waits for an object during the save preprocessing phase of a save-while-active request. The save active wait time is used if an object is not immediately available, before continuing checkpoint processing for other objects in the save request. If a lock is not obtained on an object in the specified time, the object is not saved.

If a commitment boundary is not reached for all commitment definitions in the specified time for objects that are being updated under commitment control and reaching a checkpoint together, the save operation ends.

Depending on the object type and what processing is currently active for an object, the

system can actually wait for an object multiple times during the checkpoint processing. Therefore, the value specified on the SAVACTWAIT parameter may apply multiple times for a single object.

This parameter is valid only when SAVACT(*LIB), SAVACT(*SYNCLIB), or SAVACT(*SYSDFN) is specified.

The possible values are:

number-of-seconds: A value from 0 through 99999 that indicates the number of seconds to wait for a lock on an object in the save request during checkpoint processing or to wait for an object to reach a commitment boundary.

The default value is 120 seconds.

***NOMAX:** There is no maximum wait time to wait for a lock on an object in the save request or to wait for an object to reach a commitment boundary.

SAVACTMSGQ

Message CPI3710 or CPI3712 is sent to the message queue specified on the SAVACTMSGQ parameter notifying the user that all checkpoint processing is complete for the last checkpoint object group in the library or for all libraries in the save request.

If checkpoint processing does not complete for objects being saved from the library or for all libraries in the save request, message CPI3711 is sent to the message queue specified on the SAVACTMSGQ parameter and the save operation ends.

The possible values are:

***NONE:** No notification message is sent. This is the default value.

***WRKSTN:** The notification message is sent to the workstation message queue. This option is only valid for interactive jobs. The job receives escape message CPF378A and the save request ends if this option is specified for a batch job.

library-name: The library name where the message queue is found.

The possible library values are:

***LIBL:** The library list is used to locate the message queue.

***CURLIB:** The current library is used to locate the message queue. If no library is specified as the current library for the job, the QGPL library is used.

library-name: Specify the name of the library where the message queue is located.

message-queue-name: The notification message is sent to the named message queue.

This parameter is only valid when SAVACT(*SYNCLIB), SAVACT(*LIB), or SAVACT(*SYSDFN) is specified.

Special Object Processing: Consider the following special object processing when specifying the SAVACT parameter on the SAVLIB, SAVOBJ, and SAVCHGOBJ commands.

- The SAVACT parameter cannot be specified if STG(*FREE) is specified on the save command.
- The SAVACT parameter cannot be specified if a release other than the current release is specified on the save command.
- Objects currently being decompressed on first reference are not saved if the SAVACT parameter is specified on the save command.
- Objects currently allocated by a lock type of exclusive allow read (*EXCLRD) or exclusive (*EXCL) are not eligible for save-while-active processing.
- Save files in use by another job cannot be saved if the SAVACT parameter is specified on the save command.
- Objects created after the save time may not be saved if the SAVACT parameter is specified.
- System Service Tools (SST) functions should not be used for objects currently being saved by a save-while-active operation.

Parameters on the SAVDLO Command

The following parameters can be specified on the SAVDLO command.

SAVACT
The SAVACT parameter has the following possible values.

***NO:** Objects cannot be saved while they are in use by another job. This is the default value.

***YES:** Objects in the library can be saved while they are in use by another job. Data integrity is preserved with maximum performance.

SAVACTWAIT
The SAVACTWAIT parameter is used to specify the amount of time the system should wait for an object to become available during the save-while-active checkpoint processing before continuing the save operation. If a lock is not obtained on an object in the specified time, the object is not saved.

The possible values are:

number-of-seconds: A value from 0 through 99999 that indicates the number of seconds to wait for a lock on an object in the save request during checkpoint processing before continuing checkpoint processing on other objects in the save request.

The default value is 120 seconds.

***NOMAX:** There is no maximum wait time that checkpoint processing waits for an object to become available before continuing the save operation (only valid with SAVACT(*YES).

Special Office Processing: Consider the following special office object processing before specifying the SAVACT parameter on the SAVDLO command.

- The SAVACT parameter is not allowed if STG(*DELETE) is specified on the save command.
- Target release (TGTRLS parameter) must be the current release.
- CHKFORMRK(*NO) must also be specified.
- Mail is forced to wait until checkpoint processing is complete for the library being saved.
- Documents may not be saved during save-while-active processing if a reclaim operation (RCLDLO command) is running.
- Folders may not be saved during save-while-active processing if a reorganize operation (RGZDLO command) or a reclaim operation (RCLDLO command) is running.
- An application that works with a document as a personal computer file through system application programming interfaces (APIs) or through shared folders can be updating a document when a save-while-active operation is running. When updating document data, some applications save the updates to a temporary file. The changes are not permanently written to the document until the application session ends. Most text editors, including OfficeVision/400 editor and the DisplayWrite editors work this way. If a document is being updated by this type of application while the save-while-active operation is running, the document is saved as it was before the edit session began.

  Other applications update documents directly as the data is supplied to the application. For example, some spreadsheet applications and image applications work this way. If a document is being updated by this type of application while a save-while-active operation is running, the document is not saved. Diagnostic messages CPF8A80: *Document in use* and CPF90AC: *Document not saved* are sent to the joblog to indicate the object was not saved because it was in use. You must save the document at a later date.
- For performance considerations during save-while-active processing, see the topic "Performance Considerations" on page 5-13.

Save-While-Active Procedures

This section gives the general procedures required to use the save-while-active function.

Reduce Save Outage

The following general procedures can be used to reduce your outage for particular save operations. These procedures are the recommended way to use the save-while-active function on a daily basis and apply to any of the save commands that support the SAVACT parameter. These save-while-active procedures result in objects being saved as if they were saved in a dedicated fashion and do not require any special restore recovery procedures.

Objects in a Single Library: The following procedure is recommended when all application-dependent objects are in a single library:

1. End all application jobs that are making updates to the application-dependent objects.
2. With a single save command, start the save-while-active operation for the objects that reside in the application library. Because the application jobs that were updating the objects to be saved have been ended, SAVACT(*SYSDFN) can be specified to improve checkpoint processing performance.
3. Wait for message CPI3710 or CPI3712 (if *SYNCLIB is specified) to be sent to the message queue specified on the SAVACTMSGQ parameter indicating that checkpoint processing is complete for the objects.

   If checkpoint processing does not complete for the objects, message CPI3711 is sent to the message queue specified on the SAVACTMSGQ parameter and the save operation ends.
4. Start the application jobs again.
5. After the save-while-active operation is complete, if some of the objects saved were journaled database files and the attached receiver for each of the journals being used for those files was not saved as part of the save-while-active request, save each attached receiver.

Objects in Multiple Libraries: The following procedure is recommended when application-dependent objects span multiple libraries:

1. End all application jobs that are making updates to the application-dependent objects.
2. Start the save-while-active operation for the objects that reside in the application libraries. Specify SAVACT(*SYNCLIB) on the save command.
3. Message CPI3712 message is sent when checkpoint processing is complete for all the libraries in the save request.

   If checkpoint processing does not complete for the libraries in the save request, message CPI3711 is sent to the message queue specified on the SAVACTMSGQ parameter and the save operation ends.
4. Start the application jobs again.
5. After the save-while-active operation is complete, if some of the objects saved were journaled database files and the attached receiver for each of the journals being used for those files was not saved as part of the save-while-active request, save each attached receiver.

Eliminate Save Outage

The following general procedures can be used to eliminate your save outage for particular save operations. These save-while-active procedures do not require any applications to be ended to perform the save operation. However, these procedures ***may require additional restore recovery procedures.*** See "Restore Recovery Procedures" on page 5-10 for the necessary restore recovery considerations when using the following save procedures and "Recommended Restore Recovery Procedures" on page 5-20 for some possible restore recovery procedures after restoring from save-while-active media.

General Procedures: The following general procedures apply to any save-while-active procedure that does not end the jobs that are making changes to the objects being saved:

1. During checkpoint processing, you may want to monitor the save-while-active job looking for possible lock conflicts. This is identified with a status of LCKW from the Work Active Jobs (WRKACTJOB) display. The amount of time spent waiting for locks during the checkpoint processing can be controlled by specifying a value for the SAVACTWAIT parameter. If the save-while-active job cannot acquire a lock for a particular object during checkpoint processing, that object is not saved. If

SAVACTWAIT(*NOMAX) is specified, note that the save-while-active job waits indefinitely for a lock on a particular object.

If a lock conflict exists for a particular object, you may want to identify the job that holds the conflicting lock with the Work with Object Locks (WRKOBJLCK) command and take the appropriate steps to have the job release the lock so that the save-while-active job can continue and perform the save for that particular object.

Note: If particular objects are not saved by a save-while-active request due to lock conflicts, you should consider issuing the entire save-while-active request again after any and all such lock conflicts have been resolved. To just save the individual objects that had a lock conflict again with a separate save request means that those objects are not saved in a consistent state in relationship to the other objects that were saved by the original save-while-active request. This situation can lead to a complex restore recovery procedure after restoring the objects saved by these individual save-while-active requests.

2. During checkpoint processing, if changes are being made under commitment control to the objects being saved, you may want to monitor the QSYSOPR message queue for CPI8365 messages that indicate the jobs that have commitment definitions that are preventing the save-while-active job from proceeding.

   You can control the amount of time spent waiting for commitment definitions to reach a commitment boundary by specifying a value for the SAVACTWAIT parameter. CPI8365 informational messages are only sent to the QSYSOPR message queue if the SAVACTWAIT time is specified to be at least 30 seconds. Take the appropriate steps, as outlined in the recovery portion of the CPI8365 message, to bring all commitment definitions for a job to a commitment boundary. If SAVACTWAIT(*NOMAX) is specified, note that the save-while-active job waits indefinitely for all commitment definitions to reach a commitment boundary.

   If a commitment boundary cannot be reached for a particular commitment definition, the save-while-active request is ended for the library that contained the uncommitted transaction. Depending upon the type of uncommitted changes, either CPF836C messages are sent to the job log or CPI8367 messages are sent to the QSYSOPR message queue giving the job names that had commitment definitions that prevented the save-while-active request for the library.

Objects in a Single Library: The following procedure describes how the save-while-active function can be used if all application-dependent objects are in a single library and the application jobs are not to be ended:

1. With a single save command, start the save-while-active operation for the objects that reside in the application library. If all application-dependent objects are journaled database files, then SAVACT(*SYSDFN) can be specified to improve checkpoint processing performance; otherwise, specify SAVACT(*LIB).

   Note: If all application-dependent objects are not journaled database files, and therefore you are specifying SAVACT(*LIB), it is important that you save all of the application-dependent objects with a single save request. This ensures that all of the objects reach a checkpoint together and are saved in a consistent state in relationship to each other.

2. Wait for message CPI3710 to be sent to the message queue specified on the SAVACTMSGQ parameter indicating that checkpoint processing is complete for the objects.

   When the CPI3710 message has been received, no additional lock conflicts for objects or jobs with uncommitted transactions occur.

   If checkpoint processing does not complete for the objects, message CPI3711 is sent to the message queue specified on the SAVACTMSGQ parameter and the save operation ends. However, the lock conflicts do not cause the checkpoint processing to fail. The objects with the lock conflicts are just not saved to the media.

3. After the save-while-active operation is complete, if some of the objects saved were journaled database files and the attached receiver for each of the journals being used for those

files was not saved as part of the save-while-active request, save each attached receiver.

Objects in Multiple Libraries: The following procedure outlines how the save-while-active function can be used if application-dependent objects span multiple libraries and the application jobs are not to be ended.

1. Start the save-while-active operation for the objects that reside in the application libraries. This can be done with a single save command (SAVACT(*SYNCLIB)).

   When message CPI3710 (for SAVACT(*SYNCLIB)) has been received, no additional lock conflicts for objects or jobs with uncommitted transactions occur.

   If checkpoint processing does not complete for the objects being saved from a library, message CPI3711 is sent to the message queue specified for the SAVACTMSGQ parameter and the save operation ends for the library. Objects with a lock conflict still allow checkpoint processing for the library to complete. The save operation continues. However, the objects with a lock conflict are just not saved.

2. After the save-while-active operation is complete, if some of the objects saved were journaled database files and the attached receiver for each of the journals being used for those files was not saved as part of the save-while-active request, save each attached receiver.

Recommended Restore Recovery Procedures

Note

The following section only applies if all applications that are making changes to the objects being saved with the save-while-active function are not ended until checkpoint processing is completed.

The following provides some recommended recovery procedures after restoring from the save-while-active media. The following is provided as a recommendation only, and your restore recovery procedures may need to be somewhat different depending upon your applications and your particular application dependencies.

The restore recovery for journaled files may include Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG) operations. The following recommendation uses the APYJRNCHG command exclusively, as this is the most common recovery to be performed for the files to bring them to application boundaries. However, the RMVJRNCHG command can be used instead of the APYJRNCHG to bring the files to an application boundary if removing changes from the database is desired rather than applying changes to the database. The RMVJRNCHG command can be used if the before images are being journaled for the files.

Regardless of whether all objects reached a checkpoint together, if the APYJRNCHG command is required for the restore recovery, then the TOSEQ parameter must specify a known application boundary when the APYJRNCHG command is run. Multiple APYJRNCHG commands are required if the files are journaled to different journals, and the TOSEQ value specified on each of the APYJRNCHG commands must correspond to the same known application boundary.

The following gives a general recommendation to follow for restore recovery procedures:

1. If some of the objects to be restored are journaled database files, make sure that the necessary journals are on the system. If all necessary journals are not on the system, restore the journals first. If the journals are in the same library as the files to be restored and the journals were saved with the same save request as the files, the system automatically restores the journals first.
2. Restore the objects from the save-while-active media.
3. If some of the objects restored are journaled database files, restore any required journal receivers that do not already exist on the system. Generally, the required receivers include the receivers that contain the start of save (journal code F, type SS) journal entries for the journaled file members up through the receiver that contains the journal entry that is the desired application boundary. These receivers need to be on-line for each of the journals used to journal the restored files.

4. Perform one of the following restore recovery procedures if all of the following are true:
   - Some application-dependent objects are not journaled database files.
   - All of the objects are in the same library and SAVACT(*LIB) is specified on the same save command.
   - All objects in all of the libraries are saved using SAVACT(*SYNCLIB).

   . If all of the above are true, then all of the objects reached a checkpoint together and the restored objects are in a consistent state in relationship to each other. However, if the objects need to be brought forward to some defined application boundary, the APYJRNCHG command can only be used for the journaled database files. For files that are not journaled and other object types, user-defined recovery procedures must be performed.

   If any of the above conditions are not true, then the objects are not saved in a consistent state in relationship to each other. The APYJRNCHG command can be used to bring the journaled database files forward to some common application boundary. For database files that are not journaled and other object types, user-defined recovery procedures must be performed.

   a. If all application-dependent objects are journaled database files but all of the changes made to the files are not made under commitment control, then the APYJRNCHG command must be used to bring all of the files to an application boundary.

   b. If all application-dependent objects are database files that are only updated under commitment control, but the files exist in different libraries, then the files restored are at commitment boundaries, but not all of the files will be at the same common commitment boundary. Use the APYJRNCHG command specifying the CMTBDY(*YES) parameter to bring the files forward to some common application boundary.

      Specifying CMTBDY(*YES) ensures that the apply operation starts on a commitment boundary and that the system applies complete transactions up through the specified sequence number that corresponds to your application boundary.

   c. If all application-dependent objects are database files that exist in the same library and the files are only updated under commitment control, the files will be restored as they existed at some common commitment boundary at save time. If the commitment boundary is an application boundary, then no additional restore recovery procedures are necessary. However, if the common commitment transaction boundary is not an application boundary or additional transactions exist in the journal that you want in the database, then use the APYJRNCHG command specifying the CMTBDY(*YES) parameter to bring the files forward to some defined application boundary.

      Specifying CMTBDY(*YES) ensures that the apply operation starts on a commitment boundary and that the system applies complete transactions up through the specified sequence number that corresponds to your application boundary.

Save-While-Active Examples

The following examples are provided to show some typical uses of the save-while-active function. Your exact usage of the function may differ, based on your specific application requirements.

Save-While-Active Operation to Reduce Save Outage

Example 1: This example makes use of two libraries, LIB1 and LIB2. Both libraries contain objects that are to be saved on a daily basis. Your current save strategy ends all jobs that make changes to the objects in the two libraries for the entire time that the libraries are being saved.

For this example, objects of any type can exist in the two libraries. The database files that exist in the two libraries may or may not be journaled.

Save operation: The several hour save outage can be greatly reduced by the following steps:

1. End all application jobs that are making

updates to the objects in libraries LIB1 and LIB2.

2. The following command is submitted as an individual batch job:

```
SAVLIB LIB(LIB1 LIB2) DEV(TAP01) SAVACT(*SYNCLIB) +
       SAVACTMSGQ(QSYSOPR) +
       ACCPTH(*YES)
```

   Note: The SAVOBJ or SAVCHGOBJ commands could also be used, depending upon your specific needs.

   The objects in library LIB1 and LIB2 reach a checkpoint together, as specified by SAVACT(*SYNCLIB), and the libraries are saved to TAP01. The message indicating that checkpoint processing is complete is sent to QSYSOPR.

   Access paths are also being saved for the logical files by both commands, as specified by ACCPTH(*YES). If this is specified, the access paths, in most cases, will not need to be built after restoring the files from this save media.

   Each library is saved with a separate save command to complete the checkpoint processing as quickly as possible. The libraries could be saved together with one save command, but the checkpoint processing would take longer because the system would first perform the checkpoint processing for library LIB1, followed by LIB2.

3. After checkpoint processing is complete, message CPI3712 is sent to message queue QSYSOPR.

   If checkpoint processing does not complete for the objects, message CPI3711 is sent to the message queue specified for the SAVACTMSGQ parameter and the save operation ends.

4. After receiving CPI3712 message, start the application jobs that make updates to the objects in the two libraries.

The objects exist on the tapes as they were at the time the application jobs were ended, prior to the save command being run. However, by using the save-while-active function, the amount of time that the applications are not available is greatly reduced.

Restore operation: The objects can be restored from the tapes just as if the save-while-active function was not used. No additional restore recovery procedures are required.

Save-While-Active Operation to Eliminate Save Outage

Example 2: As with the first example, this example makes use of two libraries, LIB1 and LIB2. However, for this example, both libraries contain only journaled database files and the journals for those files. The changes made to the journaled database files may or may not be made under commitment control.

The objective of this example is to not end the applications that are making changes to the objects in these libraries while the objects are being saved. Not ending the applications introduces additional restore considerations for the recovery to be performed after the objects are restored from the save-while-active media.

Save operation: The save outage is eliminated by the following steps:

1. Each of the following commands are submitted as an individual batch job:

```
SAVLIB LIB(LIB1) DEV(TAP01) SAVACT(*SYSDFN) +
       SAVACTWAIT(600) +
       SAVACTMSGQ(QSYSOPR) +
       ACCPTH(*YES)

SAVLIB LIB(LIB2) DEV(TAP02) SAVACT(*SYSDFN) +
       SAVACTWAIT(600) +
       SAVACTMSGQ(QSYSOPR) +
       ACCPTH(*YES)
```

   Note: The SAVOBJ or SAVCHGOBJ commands could also be used, depending upon your specific needs.

   Each of the database networks in library LIB1 reaches a different checkpoint, as specified by SAVACT(*SYSDFN), and the library is saved to TAP01. Likewise, each of the database networks in library LIB2 reaches a checkpoint at different points in time and the library is saved to TAP02. The system waits 10 minutes, as specified by the SAVACTWAIT parameter, to resolve each lock conflict and for any active commitment definitions to reach a commitment boundary during checkpoint processing.

   Access paths are also saved for the logical files by both commands, as specified by

ACCPTH(*YES). This is specified so that access paths, in most cases, will not be built after restoring the files from this save media.

The restore recovery procedures needed when restoring objects from this media are dependent upon each of the database members in LIB1 and LIB2 being updated with the timestamp of this save operation.

Specifying SAVACT(*SYSDFN) may allow the checkpoint processing to be performed more quickly than specifying SAVACT(*LIB).

In addition, each library is saved with a separate save command to complete the checkpoint processing as quickly as possible. The libraries could be saved together with one save command, but the checkpoint processing would take longer because the system would first perform the checkpoint processing for library LIB1, followed by LIB2. Using two save commands to two separate tape drives allows the system to perform the checkpoint processing for the libraries concurrently.

Using this technique only works if the objects being saved reside in multiple libraries because the system does not allow concurrent multiple save requests for the same library.

2. When checkpoint processing is complete for each of the libraries, a separate CPI3710 message is sent to QSYSOPR, as specified by the SAVACTMSGQ parameter. Until each CPI3710 message is sent to message queue QSYSOPR, you may want to monitor for any lock conflicts that either of the save-while-active jobs may encounter. This can be done by using the Work with Active Jobs (WRKACTJOB) command and seeing that the status of a save-while-active job is not LCKW. Use the Work with Object Locks (WRKOBJLCK) command to help resolve any lock conflicts so that the save-while-active job can continue.

   If changes are being made under commitment control to the files in LIB1 and LIB2, then you may also want to monitor for CPI8365 messages being sent to QSYSOPR. A separate CPI8365 message is sent for each job that has a commitment definition that is not at a commitment boundary and is holding out one of the save-while-active jobs. If any CPI8365 messages are received, follow the steps as outlined in the recovery portion of the message to bring all of the commitment definitions for the job to a commitment boundary. The CPI8365 messages can be received until checkpoint processing is completed for both libraries.

3. Wait for both save-while-active jobs to complete.

4. After each of the batch jobs has completed, verify that all of the required objects were saved. If lock conflicts prevented some of the objects from being saved, you may consider issuing the original save commands again after resolving any and all lock conflicts.

5. Because all of the objects from both of the libraries are not being saved to tape in a consistent state with relationship to each other, it is absolutely necessary to save the attached receiver of each journal being used to journal the files in libraries LIB1 and LIB2. If the attached journal receivers do not reside in library LIB1 or LIB2, then separate save requests must be issued to save each of the attached receivers.

   Save all of the attached receivers with the following command. Multiple save commands may be necessary for this step. Note that it is not necessary to use the save-while-active function when saving journal receivers. The following command defaults to SAVACT(*NO).

```
SAVOBJ OBJ(attached-receiver) +
       SAVLIB(attached-receiver-library) +
       OBJTYPE(*JRNRCV) +
       DEV(TAP01)
```

Restore operation: The following are the steps performed when restoring libraries LIB1 and LIB2:

1. The two libraries are restored with the following commands:

```
RSTLIB SAVLIB(LIB1) DEV(TAP01)

RSTLIB SAVLIB(LIB2) DEV(TAP02)
```

   If the journals still exist on the system, they are not restored. That is not a problem. If they did not exist, the system will restore the journal objects before the database files.

   At the completion of these restore commands, the database files exist on the system, but they will not be in a consistent state in relationship to each other.

2. Restore the necessary journal receivers that were attached at the time the libraries were saved. If the journal receivers are in libraries

other than LIB1 or LIB2 at the time of the save and they do not currently exist on the system, the following restore command can be used to restore the receivers:

```
RSTOBJ OBJ(attached-receiver-at-save-time) +
       SAVLIB(receiver-library) +
       DEV(TAP01)
```

Otherwise, if the attached receivers were in LIB1 or LIB2 at save time and they did not exist prior to the RSTLIB operation, they were restored as part of that RSTLIB operation.

3. Determine a point in time, or application boundary, that the database files in LIB1 and LIB2 will need to be brought forward to, so that all of the files are in a consistent state in relationship to each other. After determining the desired application boundary, additional journal receivers may need to be restored. If additional journal receivers are required, but are not online, they can be restored using the following restore command. Multiple restore commands may be necessary for this step.

   ```
   RSTOBJ OBJ(other-needed-receivers) +
          SAVLIB(receiver-library) +
          DEV(TAP01)
   ```

   The Work with Journal Attributes (WRKJRNA) and Display Journal (DSPJRN) commands can be helpful in finding the application boundary.

   The WRKJRNA command can be used to determine the appropriate range of receivers needed for the ensuing Apply Journaled Changes (APYJRNCHG) operations. The DSPJRN command can be used to locate the exact sequence number that identifies the desired application boundary. If multiple journals are involved, the same application boundary (most likely identified by the timestamp) must be located in each journal, and the appropriate journal sequence number must be noted.

4. Bring the database files forward to a specific application boundary using one of the following Apply Journaled Changes (APYJRNCHG) commands. Different variations of the APYJRNCHG command may be appropriate based on the given criteria.

   If changes were being made under commitment control to the files as they were being saved, then CMTBDY(*YES) may be specified on any of the following APYJRNCHG commands to ensure that commitment boundaries are preserved.

   a. If the journal was not restored and the save-while-active tapes used represent the most recent save of the database files specifying UPDHST(*YES), then the following commands can be used to apply the journaled changes to the files:

      ```
      APYJRNCHG JRN(jrnlib/jrnname) +
                FILE((LIB1/*ALL)) +
                TOENT(seq#-for-application-boundary)

      APYJRNCHG JRN(jrnlib/jrnname) +
                FILE((LIB2/*ALL)) +
                TOENT(seq#-for-application-boundary)
      ```

      If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENT parameter) that identifies the desired application boundary. Note that the TOENT sequence number is very likely different for each journal in LIB1 and LIB2, but they all identify a common application boundary.

   b. If the journal was restored and the save-while-active tapes used represent the most recent save of the database files specifying UPDHST(*YES), then the following commands can be used to apply the journaled changes to the files:

      ```
      APYJRNCHG JRN(jrnlib/jrnname) +
                FILE((LIB1/*ALL)) +
                RCVRNG(rcv-attached-at-save-time +
                       ending-rcv) +
                TOENT(seq#-for-application-boundry)

      APYJRNCHG JRN(jrnlib/jrnname) +
                FILE((LIB2/*ALL)) +
                RCVRNG(rcv-attached-at-save-time +
                       ending-rcv) +
                TOENT(seq#-for-application-boundary)
      ```

      Because the journal was restored, the system cannot determine the correct receiver range. Therefore, the correct range of receivers must be specified on the RCVRNG parameter. Note that the attached receiver at the time that the libraries were saved is the specified starting journal receiver.

      If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENT parameter) that identifies the desired application boundary. Note that the TOENT sequence number is very likely different for each journal in LIB1 and

LIB2, but they all identify a common application boundary.

c. If the save-while-active tapes used do not represent the most recent save of the database files specifying UPDHST(*YES), then you must first use the DSPJRN command to determine the sequence number of the start of save (journal code F, entry type SS) journal entry for ***each*** database member. An individual APYJRNCHG command must then be issued for each of the database members. The following command would be a example of such an APYJRNCHG command:

```
APYJRNCHG JRN(jrnlib/jrnname) +
          FILE((filelib/filename filembr)) +
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          FROMENT(seq#-for-SS-entry) +
          TOENT(seq#-for-application-boundary)
```

Because the most recent save of the database file members is not being used, FROMENT(*LASTSAVE) cannot be specified on the APYJRNCHG commands. An individual sequence number must be specified for each of the database file members in libraries LIB1 and LIB2.

Note: Some of the APYJRNCHG commands could specify multiple file members if there is a continuous series of SS entries in the journal. The members identified by the continuous series of SS journal entries could be applied to with a single APYJRNCHG command by specifying the earliest sequence number of all the SS entries in the continuous series for the FROMENT parameter.

Save-While-Active Object Locking

Most AS/400 objects can be updated by jobs while another job is saving those objects with the save-while-active function. However, there are some locking conflicts that will still occur for some objects during a save-while-active operation. The topic "Potential Locking Conflicts," shows the objects that can have a locking conflict if an attempt is made to update or use the object after checkpoint processing is complete while the object is being saved with the save-while-active function.

During the save preprocessing for an object, which includes the checkpoint processing, lock conflicts caused by updates to these objects can prevent the object from being saved to the save-while-active media. This fact, and the fact that there are no additional restore recovery procedures, are the major reasons why the recommended method for performing a save-while-active operation is to end all applications making updates to the objects until all the objects reach a checkpoint.

Objects that have no potential for a lock conflict as they are being updated during either phase of a save-while-active operation or have no special processing considerations are not listed in "Potential Locking Conflicts." See Figure 5-1 on page 5-2 for a description of save-while-active processing.

In the topic "Potential Locking Conflicts," it is indicated if there is a potential for a lock conflict during the checkpoint processing of the save-while-active operation. Some operations to update an object may not always receive an exception indicating that the object could not be locked.

Potential Locking Conflicts

CHTFMT (Chart format): Potential lock conflict exists. No changes allowed.

CSPTBL (CSP table): Potential lock conflict exists. No changes are allowed.

DOC (Document): No potential lock conflict exists. However, the Original view of the document before the edit session is saved.

Note: Some applications update document library objects directly as the data is supplied to the application rather than saving the updates to a temporary file. Document library objects that are being updated this way (typically documents being updated by PC-based applications) are not saved.

DTAARA (Data area): No potential lock conflict exists. However, the object may be saved in an inconsistent state if user-supplied update was done without using the CHGDTAARA interface.

DUO (Mail documents): No potential lock conflict exists. However, see note for *DOC.

| ***FILE-LF (Logical file):*** Changes are not allowed for the following:

- Change Logical File attributes (CHGLF, SQL 'Label ON', SQL 'Comment ON')
- Change Object Description (CHGOBJD)
- Add Logical File Member until checkpoint processing complete. (ADDLFM or CRTLF with member create)
- Delete File (DLTF)
- Remove File Member (RMVM)
- Grant Object Authority to a database file (GRTOBJAUT)
- Revoke Object Authority to a database file (RVKOBJAUT)
- Change Object Ownership for a database file (CHGOBJOWN)
- Move Object for a database file (MOVOBJ)
- Rename Object for a database file (RNMOBJ)
- Rename Member (RNMM)
- Start or End Journal Access Path (STRJRNAP or ENDJRNAP)

| ***FILE-PF (Physical file):*** No potential lock conflicts exist. However, Changes are not allowed for the following:

- FORTRAN End-of-Data (EOD) function.
- Change Physical File attributes (CHGPF, CHGSRCPF, SQL 'Label ON', SQL 'Comment ON')
- Change Object Description (CHGOBJD)
- Clear Physical File Member (CLRPFM, OPEN with option to clear file (such as edit a member under SEU)
- Add Physical File Member until checkpoint processing complete. (ADDPFM, CRTPF or CRTSRCPF with member create)
- Delete File (DLTF)
- Remove File Member (RMVM)
- Reorganize Physical File Member (RGZPFM)
- Grant Object Authority to a database file (GRTOBJAUT)
- Revoke Object Authority to a database file (RVKOBJAUT)
- Change Object Ownership for a database file (CHGOBJOWN)
- Move Object for a database file (MOVOBJ)
- Rename Object for a database file (RNMOBJ)
- Rename Member (RNMM)
- Apply or Remove Journaled Changes (APYJRNCHG or RMVJRNCHG)
- Start or End Journal Access Path (STRJRNAP or ENDJRNAP)
- Start or End Journaling Physical File (STRJRNPF or ENDJRNPF)

| ***FILE-SAVF (Save file):*** A potential lock conflict exists if SAVFDTA(*YES) is specified. No lock conflict exists if SAVFDTA(*NO) is specified.

FLR (Folder): No potential lock conflict exists. However, see object type *DOC (Document).

IGCDCT (IGC dictionary): No potential lock conflict exists. However, you may not have the current edit changes if you exit during checkpoint processing.

IGCTBL (IGC table): No potential lock conflict exists. However, you cannot list IGC table from CGU when the table is being saved.

JRN (Journal): No potential lock conflict exists. However, the following is not allowed:

- Change Journal (CHGJRN)
- Delete Journal (DLTJRN)
- Move Object (MOVOBJ) for a journal
- Work with Journal (WRKJRN) to recover a journal

JRNRCV (Journal Receiver): No potential lock conflict exists. However, the following are not allowed:

- Change Journal (CHGJRN) to change the journal receiver
- Create Journal (CRTJRN) and the receiver is to be attached to a journal
- Delete Journal (DLTJRN) and the journal receiver is associated with the journal
- Delete Journal Receiver (DLTJRNRCV) for a journal receiver
- Move Object (MOVOBJ) for a journal receiver

- Work with Journal (WRKJRN) to recover a damaged journal receiver

MSGF (Message file): No potential lock conflict exists. However, you cannot add or remove a message description during save processing.

PGM (Program): No potential lock conflict exists. However, you cannot update the associated space of program object during save processing.

QMFORM (Query manager form): A potential lock conflict exists. No update is allowed during save processing.

QMQRY (Query manager query): A potential lock conflict exists. No update is allowed during save processing.

QRYDFN (Query definition): A potential lock conflict exists. No update is allowed during save processing.

SBSD (Subsystem description): A potential lock conflict exist. The subsystem cannot be started during save processing.

USRIDX (User queue): No potential object lock exists. However, you can have inconsistent views if changing the object during save processing. The user must have *SHRUPD lock before making the change.

USRQ (User queue): No potential object lock exists. However, you can have inconsistent views if changing the object during save processing. The user must have *SHRUPD lock before making the change.

USRSPC (User space): No potential object lock exists. However, you can have inconsistent views if changing the object during save processing. The user must have *SHRUPD lock before making the change.

Part 3. Auxiliary Storage Pools

Chapter 6. Auxiliary Storage Pools

The following topics provide general information about single-level storage, allocation of disk space, disk failure, and auxiliary storage pools (ASPs).

Understanding Single-Level Storage

The concept of **single-level storage** is that, at a low level within the machine, a single virtual address space (virtual storage) exists. This storage is large enough to contain all data to be stored on the system. Functions operating above this low level see data as being stored in contiguously addressable locations in this space, no matter how the data may in fact be stored on auxiliary storage and in main storage.

The AS/400 machine interface (MI) instruction set provides an object-related interface. Space for an object is allocated as connected virtual storage. In reality, the data resides on auxiliary storage in disk extents that are not connected. Although programs see this data as being addressed directly in virtual storage, the data is brought into main storage, when needed, for use by programs run by the processing unit.

The internal machine functions that support the virtual address space involve three primary parts:

- **Auxiliary storage management** allocates and deallocates disk space for data placed in virtual storage.
- **Main storage management** copies data into main storage when it is needed, and then back to its permanent home on auxiliary storage after it has been updated.
- **Processing unit addressing** automatically addresses the appropriate location in main storage when a virtual address is used.

Refer to Figure 6-1 on page 6-2 for an illustration of single-level storage.

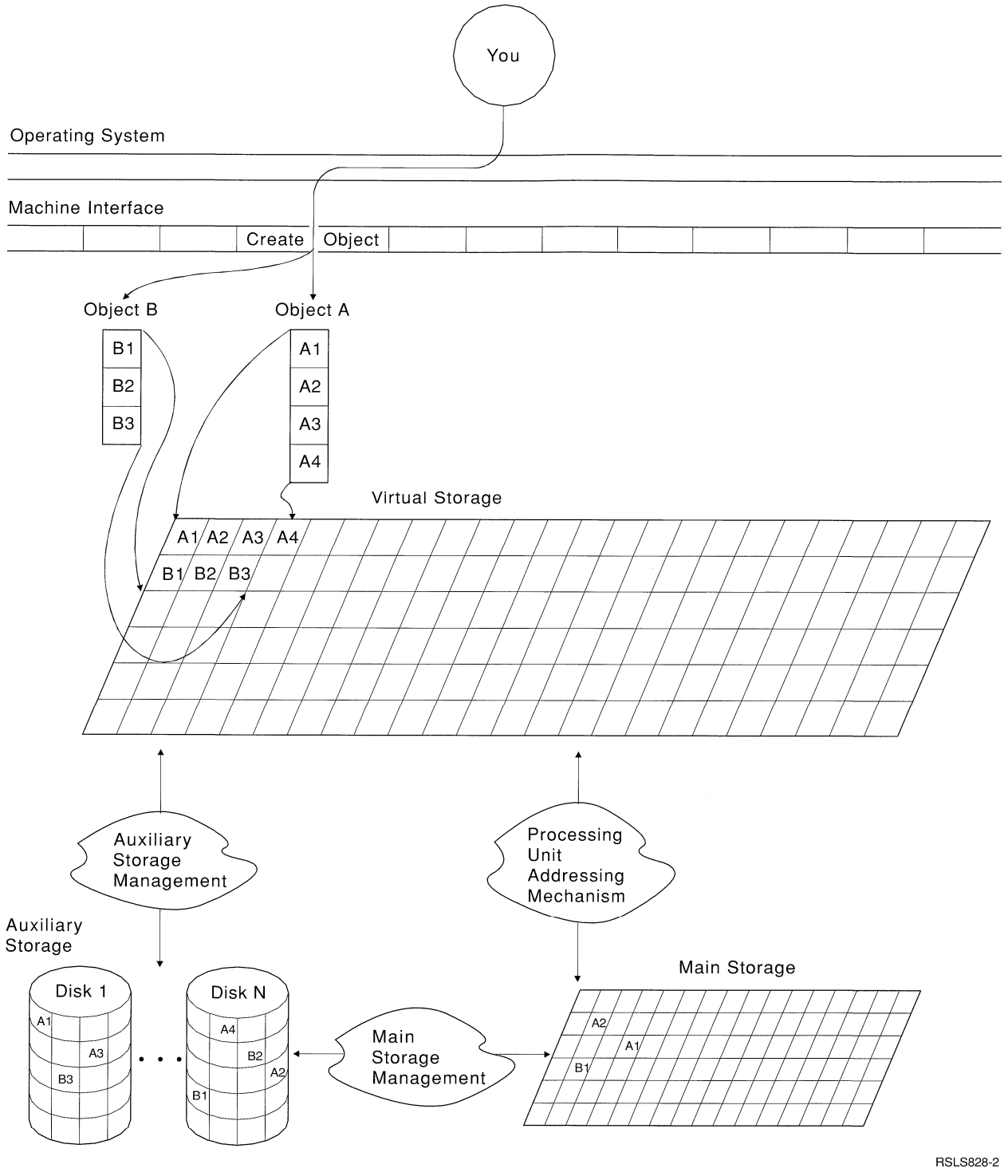


Figure 6-1. Single-Level Storage

Allocation of Space to Store Objects on Disk

When an object is created, you do not have to allocate space for the object. Allocation of space on auxiliary storage is done automatically and is determined by the space required, space available, and by space utilization of each disk attached to the system. To balance space utilization, auxiliary storage management typically allocates space for an object across two or more disks.

As an example of allocation, Figure 6-1 on page 6-2 shows that object A, when created, was allocated into virtual storage units at connected locations A1, A2, A3, and A4. This unit of storage, referred to as a page, is determined by internal machine algorithms. These pages of virtual storage are then physically allocated to auxiliary storage to provide balanced use. The locations in auxiliary storage normally will not be connected. Similarly, locations selected in main storage for these pages of virtual storage (units A1 and A2 in the figure) are not connected.

Disk Failure with Data Loss

Having the system manage disk storage means you do not have to perform this time-consuming function, but it also means you do not know where data for objects resides on disk. Because you do not know this, there is no way for you to tell what was stored on a disk when it fails. Furthermore, because the system spreads the data for an object across many disks, it is likely that pieces of many objects were lost when a disk failed. No functions exist to tell you what piece of an object was lost or how to restore a piece of an object. Because you cannot always determine what was lost, your only recovery option is to reload the entire system from backup media.

If you have not saved all objects on off-line media immediately prior to a failure, you will not be able to recover recently entered data. Therefore, when your previously saved objects are restored, the system is operational but the database is not current.

Even if journaling of database files is used, the receiver that contains the recently entered transactions will not be available if it too was lost when the disk failed. You will have to use alternatives to recover recently entered data, and in some cases that may not be possible. The recovery of recently entered data can complicate and lengthen the overall recovery process.

Considering these complications to recovering lost data, it is important to prepare for the possible need for recovery. You can simplify the recovery process by using disk recovery tools.

How the System Addresses Disk Units

The system processor requires information located in main storage. Main storage transfers information to disk (auxiliary storage pools (ASPs)).

Because ASPs are made of many disk units, other hardware is required to manage the transfer of data. Figure 6-2 on page 6-4 illustrates the hardware.

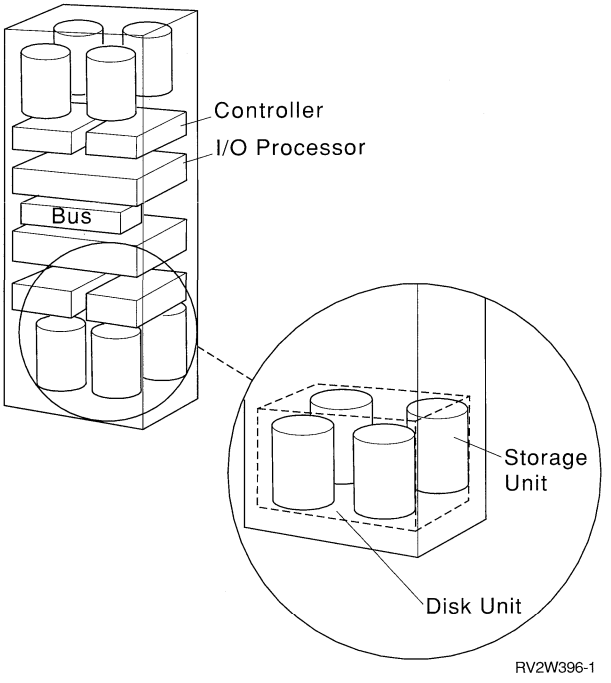


Figure 6-2. Hardware Used for Data Transfer

This hardware includes:

Bus: The bus is the main communications channel for input and output data transfer.

- All system models have at least 1 bus.
- Models B50 and above have 2 or 3 buses.
- 9406 Models D, E, and F have up to 7 buses.
- The bus is addressed by the system as 0, 1, or 2 buses.

I/O processor: The I/O processor is attached to the bus and controls information between the bus and specific groups of I/O controllers.

- There are different types of I/O processors, depending on the input and output types to control.
- Disk storage uses the 6110, 6111, 6112, or 2615 I/O processor for the 9406 system unit.
- One 6110 or 6112 I/O processor can control up to:
  - Eight 9332 disk units
  - Eight 9335 B01 disk units
- One 6111 or 6112 I/O processor can control up to two 9336 disk units (four to eight storage units).
- The 9332, 9335, and 9336 disk units cannot be mixed on the same I/O processor.
- I/O processor performance is directly related to the amount of input and output activity it controls.
- The I/O processor is addressed by the system as 0 through F.

Controller: The controller attaches to the I/O processor and handles the information transfer between the I/O processor and the disk units.

- Disk units have different types of controllers based on disk type.
- Disk unit 9332 has its own controller.
- The 9335, model A01, controller is used to control up to four 9335, model B01, disk units. Because of controller contention, performance requirements may dictate that the A01 controllers attach to no more than two B01 disk units.
- The controllers are addressed by the system as 00 through 07 on each bus.

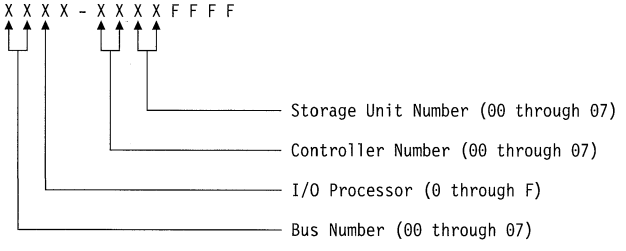
Disk units: Disk units are the actual devices that contain the storage units.

- Disk units are further divided into independently addressed sections known as storage units.
- The number of storage units per disk unit vary by disk unit type and model. For example, the 9332, model 400, has two storage units.
- Storage units are addressable by the system. For example, the 9335 disk unit has 0 through 7, and the 9332 disk unit has 0 through 1.

How the System Addresses Individual Storage Units

For data transfer, the system needs a way to identify a single storage unit. Every hardware component (bus, I/O processor, controller, and storage unit) has a unique address.

The address has eight digits and is identified as follows:



How Disk Units Are Attached to the System

In addition to understanding single-level storage, it is important that you understand how disk units are attached to the system before proceeding to the discussions of the disk recovery tools.

Different models of disk units are attached to the AS/400 system. The storage areas within the disk units are referred to as **storage units**. The number of storage units and the storage capacity per storage unit varies by disk unit type and model. Table 6-1 and Table 6-2 summarizes the characteristics of each disk device type and model.

Table 6-1. Storage Capacity by Disk Unit Type

Disk Unit Type	Model	Storage Units per Disk Unit	MB per Storage Unit
2800	001	2	320.18
2801	001	2	320.18
6100	015	1	315.59
6102	010/30	1	320.18
6103	010/030	1	400.69
6104	030	1	988.80
6105	010/020/030/040	1 or 2	320.18
6107	010/030/020/040	1 or 2	400.69
6109	030/040	1 or 2	988.80
6603	030/040	1 or 2	1967
9332	2xx/4xx	1 or 2	200.28
9332	6xx	2	300.42
9335	B01	2	427.93
9336	010	2 to 4	471.2
9336	020	2 to 4	857.2
9337	110 115	4 to 7 4 to 7	406.6 Base 542 Feature
9337	120	4 to 7	727 Base 970 Feature
9337	140	4 to 7	1475 Base 1967 Feature
9337	010 020	2 to 7	542 Base 970 Feature
9337	040	4 to 7	1967
9337	210 215	up to 8 up to 8	542 406
9337	220 225	up to 8 up to 8	970 727
9337	240	up to 8	1967
9337	240	up to 8	1475
9337	222	up to 8	1967
9337	342	up to 8	1967

Table 6-2. Models and Capacity for the 9337 Disk Unit Subsystems

9337 Model	MB/Unit	MB/Number of Units						
		2	3	4	5	6	7	8
01x[1]	542	1084	1626	2168	2710	3252	3794	N/A
02x[1]	970	1940	2910	3880	4850	5820	6790	N/A
040	1967	N/A	N/A	7868	9835	11802	13769	N/A
21x	542	1084	1626	2168	2710	3252	3794	4336
22x	970	1940	2910	3880	4850	5820	6794	7760
240	1967	3934	5895	7868	9835	11802	13769	15736
11x HA[3,5]	406/542	N/A	N/A	1626	2168	2710	3252	[2]
12x HA[3,5]	728/970	N/A	N/A	2910	3880	4850	5820	[2]
140 HA[3,5]	1475/1967	N/A	N/A	5900	7867	9834	11801	[2]
21x HA[3,4,5]	406/542	N/A	N/A	1626	2168	2710	3252	3794
22x[3,4,5]	728/970	N/A	N/A	2910	3880	4850	5820	6790
240 HA[3,4,5]	1475/1967	N/A	N/A	5900	7867	9834	11801	13769

Notes:

1 This base model comes with two disk units as the base configuration without device parity protection. This model can be upgraded to use device parity protection.

2 This unit is called the write-assist disk unit (WAD). The WAD is an integral part of some of the 9337 high-availability models. The AS/400 system cannot address or manage this disk unit.

3 This model comes with four disk units as the base configuration with device parity protection built in. Disk units 1, 3, 5, and 7 are device parity units. One fourth of four disk units for this model is reserved for parity information and is unavailable for user data. Additional units of the same type can be added and all storage on those units will be available for user data.

5 This model has the write cache feature.

6 This model must have two disk units in the base configuration and a minimum of two feature disk units to support device parity protection.

Disks are assigned to an auxiliary storage pool (ASP) on a storage unit basis. The system treats each storage unit within a disk unit as a separate unit of auxiliary storage. When a new disk unit is attached to the system, the system initially treats each storage unit within it as nonconfigured storage units. Through dedicated service tool (DST) options you can allocate these storage units to either the system ASP or a user ASP of your choosing. When allocating nonconfigured storage units, use the serial number information assigned by the manufacturer to ensure you are selecting the correct physical device. Additionally, the individual storage units within the disk unit can be identified through the *Address* field on the DST Display Disk Configuration display.

When you allocate a nonconfigured storage unit to an ASP, the system assigns a number to the storage unit. The storage unit number can be used instead of the serial number and address.

When a storage unit has mirrored protection, two storage units (mirrored pair) are assigned the same unit number. The serial number and the address distinguish between the two units in a mirrored pair.

You may want to know which physical disk is being identified with each unit number. Make note of the unit number assignment to ensure that correct identification is made. If you need to verify the unit number assignment, use the DST Display Configuration Status display to show the serial numbers and addresses of each unit.

The storage unit addressed by the system as unit 1 is always used by the system to store the Licensed Internal Code. The amount of storage used on unit 1 is quite large and varies depending on the configuration of your system. Because unit 1 contains the initial programs and data used during an IPL of the system, it is also known as the **load source unit**.

The system reserves 1.08 megabytes of storage per storage unit (other than unit 1), reducing the amount of space available by that amount.

General Information about Auxiliary Storage Pools

An **auxiliary storage pool (ASP)** is a group of units defined from all the disk units that make up auxiliary storage. ASPs provide the means of isolating objects on a specific disk unit, or disk units, to prevent the loss of data due to a disk media failure on other disk units not included in the ASP.

If the system experiences a disk unit failure with data loss, recovery is required only for the libraries or objects in the ASP that contained the failed disk. System and user objects in other ASPs are protected from the disk failure.

In addition to the recovery advantage, placing libraries or objects in an ASP can improve performance because the system dedicates the disk units associated with that ASP to the objects in that ASP. In a heavy journaling environment, placing libraries and objects in a user ASP can reduce contention between the journal receivers and the files if they are in different ASPs, and can improve journaling performance.

Auxiliary Storage Limits

During an IPL, the system determines how much auxiliary storage is configured on the system. The total amount is the sum of the capacity of the configured units and their mirrored pairs, if any. Disk units that are not configured are not included. The amount of disk storage is compared to the maximum supported for a particular model (see Table 6-3 on page 6-9). If more than the recommended amount is configured, a message (CPI1158) is sent to the system operator's message queue (QSYSOPR) and the QSYSMSG message queue (if it exists on the system). This message indicates too much auxiliary storage exists on the system. This message is sent once during each IPL as long as the amount of auxiliary storage on the system is more than the maximum amount supported.

The following information is used to determine the maximum amount of auxiliary storage recommended for each system model. The numbers have been rounded to the nearest tenth.

Table 6-3. Auxiliary Storage Limits by System Model	
System Model	**Maximum Auxiliary Storage in GB**
B30, B35, B40, B45	13.7
B50	27.5
F10, F20, F25	19.7
D35, E35, D45, E45, F35, F45	43.0
B60, B70	55.1
D50, E50, F50	78.0
D60, E60, D70, E70, F60	114.0
D80, E80, E90, E95, F70, F80, F90, F95	192.0
Note: For models not shown in the table, no message is sent. Restrictions already exist that limit the amount of auxiliary storage on the models not shown.	

System ASP

The system ASP (ASP 1) is automatically created by the system and includes unit 1, as well as all other configured disks that are not assigned to a user ASP. Unlike a disk that is configured, a disk in nonconfigured status is attached to the system, but is not part of an ASP and is not being used. The system ASP contains all system objects for the OS/400 licensed program and all user objects not assigned to a user ASP.

If the system ASP fills to capacity, the system ends abnormally. You must IPL the system again and take the correct steps, such as deleting objects, to reduce the storage utilization within the system ASP.

You can specify a threshold that, when reached, warns the system operator of potential shortage of space.

For example, if you set the threshold value at 80 for the system ASP, the system operator (QSYSOPR) and system message queue (QSYSMSG) are notified when the system ASP is 80% full. This message is sent every hour until the value is changed, or until objects are deleted or transferred out of the system ASP. If this message is ignored and the system ASP fills to capacity, the system stops and you must perform an IPL of the system. The system may have trouble performing another IPL because objects were left in an unusable condition.

Do not wait until your system fills to capacity before you save it. Saving your system may cause it to fill to capacity.

For a description of the details associated with the system ASP filling to capacity while checksum protection is in effect, see the topic "Handling Protected Storage That Has Reached Maximum Storage."

Notes:

1. For Version 1 hardware, main storage dump space must be contained on unit 1 in the system ASP.
2. For Version 2 hardware on 9406 Model D, the main store dump space must be contained on 2800 Model 001 storage units in the system ASP. Enough 2800 Model 001 storage units must be configured to contain the main storage dump space in the system ASP.
3. If the system contains (or is expected to contain) 224MB or more of main storage, then at least two 2800-001 storage units must be configured to the system ASP (ASP 1). If the system ASP has mirrored protection, then all four 2800-001 storage units must be configured to the system ASP. Failure to provide enough disk units to contain the main storage dump space may result in significantly longer IPLs and will greatly limit problem support.
4. If there is not enough free space to allocate to main store dump space, a message (CPI0987) will be sent to the QSYSOPR message queue and the QHST log.

User ASPs

A user auxiliary storage pool (ASP) is created by grouping together a physical set of disk units and assigning them to an ASP. This protects against the loss of both the files and the receivers if a disk media failure occurs.

You can configure user ASPs 2 through 16. They can contain libraries and associated objects. Associated objects in the library can include journals, journal receivers, and save files. User ASPs (old type) created before Version 1 Release 3 could only contain journals, journal receivers, and save files whose libraries were in the system ASP.

A user ASP must be exclusively one type or the other:

- If the user ASP contains libraries, then all the objects in the user ASP must be contained within these libraries. Their libraries cannot be in the system ASP. This is the recommended way to use user ASPs.
- If the library for the objects in the user ASP exists in the system ASP, then the only objects allowed in the user ASP are journals, journal receiver and save files. This type of user ASP is not recommended.

When using user ASPs in the recommend way, consider the following:

- Journals and the files being journaled **must** be in the same ASP. The journal receivers should be placed in a different ASP. This protects against the loss of both the files and the receivers if a disk media failure occurs.
- Physical file and logical files should be in the same library in the user ASP. If they are in different libraries in the ASP, you must ensure that the physical files are restored before the logical files.
- Journaling cannot be started on an object (STRJRNPF or STRJRNAP command) if the journal (object type *JRN) and the object to be journaled are in different ASPs.
- Journaling cannot be started again for a file that is saved and then restored to a different ASP that does not contain the journal. The journal and the file must be in the same ASP for journaling to be automatically started again for the file.

- If the system ASP is lost, data is preserved in the user ASPs. However, when RCLSTG is run during the recovery procedures, all objects in the user ASP have their ownership transferred to default owner (QDFTOWN) user profile. You must transfer ownership for tyhe objects back to the original owners.

Isolating libraries and associated objects in a user ASP protects them from disk failures in other ASPs and reduces recovery time. See "Object Types Not Allowed in a User ASP" on page 6-14 for restrictions.

The advantages are:

- Additional data protection. By separating libraries or objects in a user ASP, you protect them from data loss when a disk unit in the system ASP or other user ASPs fails. For example, if you have a disk unit failure, and data contained on the system ASP is lost, objects contained in user ASPs are not affected and can be used to recover objects in the system ASP. (For information on recovering objects, see *Basic Backup and Recovery Guide*.) Conversely, if a failure causes data contained in a user ASP to be lost, data in the system ASP is not affected.
- Improved system performance. You can place libraries or objects in a user ASP, allowing you to dedicate the disk units in the ASP exclusively for the use of those objects. If you do extensive journaling, a dedicated disk unit for the journal receiver can also improve journaling performance.

  However, placing many active journal receivers in the same user ASP is not productive because the resulting contention between writing to more than one receiver in the ASP can slow system performance. For maximum performance, place each active journal receiver in a separate user ASP.
- All disk types can be allocated to a user ASP, but unit 1 is always allocated to the system ASP (ASP 1).
- User ASPs are configured using the DST display, Work with ASP Configuration. For more information about configuring user ASPs, see Chapter 7, "Working with Auxiliary Storage Pools" on page 7-1.
- As with the system ASP, you can specify individual threshold values for each user ASP. For information on how to do this using dedicated services tool or system service tools, see the topics "Accessing DST Options" on page 7-3 and "Accessing SST Options" on page 7-2. If you do not specify a value, the system uses the default value of 90%.
- You can use mirrored or checksum protection on one or more ASPs.

  Notes:

  1. All ASPs, including the system ASP, must have mirrored protection or consist entirely of disk units with device parity protection. to ensure that the system continues to run after a disk failure in an ASP.
  2. If a disk failure occurs in an ASP that does not have mirrored protection, the system may not continue to run, depending on the type of disk unit and the error.
  3. If a disk failure occurs in an ASP that has mirrored protection, the system continues to run (unless the failed unit has a mirrored unit that has failed).
  4. If a storage unit fails in an ASP that has device parity protection, the system continues running as long as no other storage unit in the disk unit

fails. For more information about device parity protection, see Chapter 10, "Device Parity Protection and Mixed ASP Support"

Considerations for Using User ASPs

You can create two types of user ASPs.

- Create your libraries in the user ASP by using the ASP parameter on the CRTLIB command. All objects that are created in those libraries are allocated to that user ASP. This type is recommended.
- Create your libraries in the system ASP (ASP 1), and then create the journals, journal receivers, and save files in user ASPs by using the ASP parameter on the corresponding create command (CRTJRN, CRTJRNRCV, or CRTSAVF). This is the old type of ASP. This type of ASP (supported before Version 1 Release 3) is **not recommended** because the recovery steps are more complex.

A user ASP must be exclusively one type or the other. If the library for the objects in the user ASP exists in the system ASP, then the only objects allowed in the user ASP are journals, journal receiver and save files. If the user ASP contains libraries, then all the objects in the user ASP must be contained within these libraries. Their libraries cannot be in the system ASP.

Consider the following when creating the recommended type of user ASPs:

- An IPL cannot be performed with a failed unit in any ASP (unless the failing unit has a mirrored unit that has not failed or the disk unit has device parity protection). (Chapter 10, "Device Parity Protection and Mixed ASP Support" on page 10-1 has more information about device parity protection.) The failing unit must be repaired or replaced, or the ASP definition for the failing unit must be removed by changing the disk configuration.
- The system ASP should have checksum protection, device parity protection, or mirrored protection. Using checksum protection, device parity protection, or mirrored protection reduces the chance of the system ASP losing all data. When the system ASP is lost, addressability to objects in every user ASP is lost. The addressability can only be recovered by restoring the entire system or by running the Reclaim Storage (RCLSTG) command.
- There are performance considerations when using checksum protection or mirrored protection.
- When a user ASP becomes full, objects can overflow into the system ASP. If the user ASP overflows, the overflow status for the ASP should be reset as soon as possible for the following reasons:
  - The contents of the user ASP are cleared if a user ASP overflows and a data-loss failure occurs that causes either the user ASP or the system ASP to be cleared.
  - If a user ASP is cleared while in overflow status, the user must run the Reclaim Storage (RCLSTG) command to recover. A reclaim storage operation finds and cleans up any objects or parts of objects that overflowed into the system ASP.
- No database network can cross ASP boundaries. A **database network** consists of a set of related physical and logical files. All logical files built over a

single physical file make up a simple network. These simple networks can then be grouped together by a common logical file that is built over the physical files from two or more simple networks. Simple networks are continually grouped until there exists no logical file that can group two smaller networks together. The final result is a database network.

- Users cannot create a file in one ASP that depends on a file in a different ASP. All based-on physical files for a logical file must be in the same ASP as the logical file. The system builds access paths only for database files in the same ASP as the based-on physical file (temporary queries are not limited). Access paths are never shared by files in different ASPs. Record formats are not shared between different ASPs; a format request is ignored and a new record format is created.
- SQL collections are not allowed in user ASPs. SQL collections require logical files with based-on physical files in library QSYS. Because the based-on physical files are in the system ASP and the logical files in a user ASP, an SQL collection is not allowed.

  Warning!

  System or product libraries (libraries that begin with a Q or #) must not be created in or restored to a user ASP. Doing so can cause unpredictable results.

- When using user ASPs in the recommended way, journaling cannot be started on an object (STRJRNPF or STRJRNAP command) if the journal (object type *JRN) and the object to be journaled are in different ASPs.
- When using user ASPs in the recommended way, journaling cannot be started again for a file that is saved and then restored to a different ASP that does not contain the journal. The journal and the file must be in the same ASP for journaling to be automatically started again for the file.
- The Reorganize Physical File Member (RGZPFM) command requires sufficient work space to contain a copy of the member being reorganized. The work space must exist in the same ASP.

- Normally, you cannot move objects from a library in one ASP to a library in another ASP using the Move Object (MOVOBJ) command. The one exception to this rule is files in library QRCL can be moved to their original libraries.

  Journals and journal receivers cannot be moved out of the libraries that they were created in. The only way to move an object from one ASP to another is to create a new object. An object can be placed in a different ASP by doing the following:

  1. Save the object.
  2. Delete the existing object.
  3. Create the library using the Create Library (CRTLIB) command.
  4. Restore the object to the library.

Object Types Not Allowed in a User ASP

The Create Library (CRTLIB) command allows you to place a library in a specific user ASP.

Commands that restore or create objects allowed in user ASPs, automatically place the object in the same ASP as the library. The system does not allow a user to create an object in a user ASP if the object type is not allowed in a user ASP.

The following list shows the object types that are not allowed in a user ASP.

Object Type	Description
***AUTL**	Authorization list
***CFGL**	Configuration list
***CNNL**	Connection list
***COSD**	Class-of-service description
***CTLD**	Controller description
***DEVD**	Device description
***DOC**	Documents
***EDTD**	Edit description
***FLR**	Folders
***IGCSRT**	DBCS sort
***IGCTBL**	DBCS table
***JOBSCD**	Job scheduler
***LIND**	Line description
***MODD**	Mode description
***NWID**	Network interface description
***RCT**	Reference code translation table
***S36**	System/36 machine description
***USRPRF**	User profiles

Limiting the Types of Objects in a User ASP

You can limit the types of objects in a user ASP (ASPs 2 through 16) if a library does not exist in the user ASP. However this type of ASP (available before V1R3) is **not recommended** because it requires a complex set of recovery steps.

You can limit the object types to:

- Journals (*JRN)
- Journal receivers (*JRNRCV)
- Save files (*SAVF)

You create any of the object types listed above in a user ASP and have the library in the system ASP. You can isolate the object in a user ASP by specifying a user ASP number in the ASP parameter on the create command for the specific object type (CRTJRN, CRTJRNRCV, or CRTSAVF) will create the object in the specified

ASP. If the user ASP contains any of these isolated objects, no other object types (other than journals, journal receivers, and save files) can be created in that user ASP.

If you want to create a library in a user ASP, but the user ASP contains journals, journal receivers, and save files whose library is in the system ASP, you must first delete all the journals, journal receivers, and save files in the user ASP.

Checksum protection is not recommended for user ASPs containing journals, journal receivers, and save files whose library is in the system ASP for the following reasons:

- Inefficient because the use of journaling and save files is a different form of backup. Checksum protection would be backing up a backup copy.
- Poor performance because of redundancy data.

Use the Display Object description (DSPOBJD) command and specify DETAIL(*FULL) to identify the ASP that the object is in.

Planning the Configuration of User ASPs

Figure 6-3 on page 6-16 illustrates a configuration of system and user auxiliary storage pools.

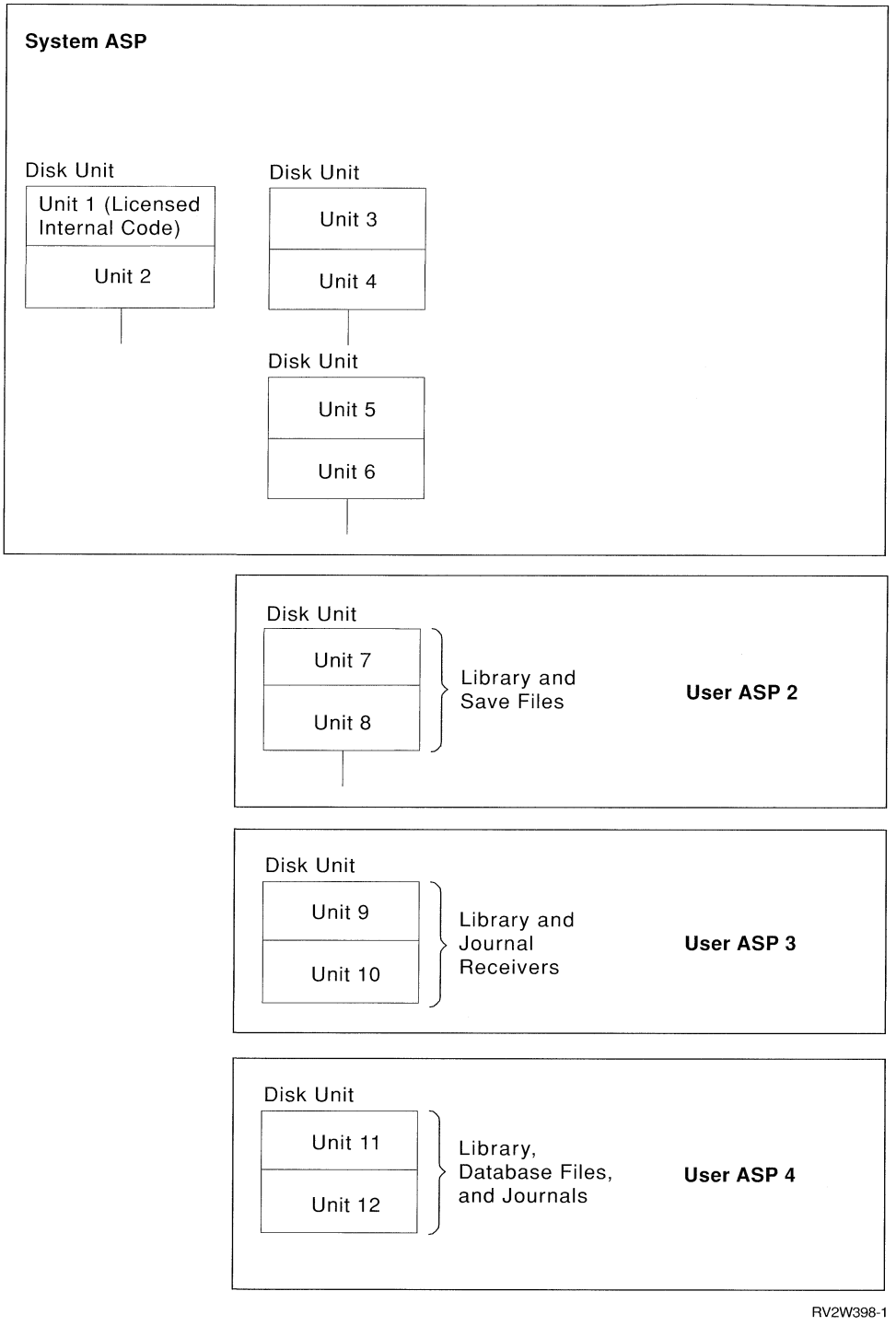


Figure 6-3. Example of System and User ASP Configuration

When planning your user ASP configuration, you must first consider what objects to place in each user ASP. The following list can help you determine the appropriate number of user ASPs for your system:

Recovery: If you are using user ASPs for recovery, configure at least one complete storage device for a single user ASP. Some 9332 or 9335 disk unit failures can damage both storage units in the disk unit. If this type of failure occurs, and the two units are in different unprotected ASPs, you would lose the data on both

ASPs. If you configure one unit in the system ASP containing data files, and the second unit as a user ASP containing the journal receiver, a single device failure could destroy both the data files and journal entries for the file. The exceptions to this rule are the 2800, 9336, and 6107 disk units. The storage units within the 2800, 9336, 9337 models 110 and 120, or 6107 models 20 and 40 disk units are independent of each other.

Improved system performance only: If you are using user ASPs for better system performance, consider dedicating the ASP to one object that is very active. In this case, you can configure the ASP with only one disk unit. It is not recommended that you assign one storage unit to an ASP unless the disk unit type is a 9336 or a 9337 models 110 and 120.

For example, journaling performance can be improved by allocating one user ASP exclusively for journal receivers attached to the same journal. By having the journal and database files in a separate ASP from the attached receivers, there is no contention for journal receiver write operations because the units associated with the ASP do not have to be repositioned before each read or write operation.

Another way to improve performance is to make sure there are enough storage units in the user ASP to support the number of physical input and output operations that are done against the objects in the user ASP. You may have to experiment by moving objects to a different user ASP and then monitoring performance in the user ASP to see if the storage units are used excessively. For more information on working with disk status (WRKDSKSTS command) to determine if the storage units have excessive use, see the *Work Management Guide*. If the units have excessive use, you should consider adding more disk units to the user ASP.

Extensive journaling: If journals and files being journaled are in the same ASP as the receivers and the ASP overflows, you must end journaling of all files and delete all objects in the ASP before clearing the overflowed ASP. However, other recovery procedures, such as deleting unused objects or moving objects to a different ASP, should be tried before clearing the overflowed ASP. If the journal receiver is in a different ASP than the journal, and the user ASP that the receiver is in overflows, you can create a new receiver in a different user ASP, change the journal (CHGJRN command), save the detached receiver, delete it, and then clear the overflowed ASP without ending journaling.

Access path journaling: If you plan to use access path journaling, it is recommended that you first change the journal to a journal receiver in the system ASP (ASP 1) for a few days. Start access path journaling to see storage requirements for the receiver before you allocate the specific size for a user ASP.

User ASPs can be configured with more than one storage unit. The maximum number of storage units allowed is determined by the maximum allowed for the entire system. User ASPs are configured and identified by a user-assigned number, from 2 through 16, and are assigned using the Work with ASP Configuration display. The *Advanced Backup and Recovery Guide* The topic "Creating a User ASP and Adding Disk Units to the New User ASP" on page 7-4 has more information on how to create a user ASP and add disk units to the new user ASP.

Meeting Storage Requirements

There are several ways to obtain storage. The following list describes the ways to obtain storage. The first is the most preferable only if the save and restore is avoided. Otherwise, the first and second are the same.

- You can reconfigure some of the storage units in the system ASP. If checksum protection is in effect, you may have units in the checksum protected system ASP that are not eligible to participate in checksum sets. For more information on checksum protection, see Chapter 8, "Introduction to Checksum Protection" on page 8-1.

  Because these units cannot contain data from protected storage, they are probably of limited value in the system ASP that has checksum protection. You can make better use of these units by moving them into user ASPs. The advantages of this reconfiguration include:

  - No additional disks need to be purchased to form the user ASP.
  - A save and restore operation of the ASP that has checksum protection **is not** required to accomplish the reconfiguration.
- You can move a unit from the system or user ASP. This method **does not** require that you save and restore the ASP from which the unit is taken. No additional equipment is required.
- You can purchase new disks and add storage units to your existing user ASPs, or configure them into new user ASPs.

If you choose to reassign a unit from one ASP to another, you must carefully consider the effect that it does have on the capacity of both ASPs. After reviewing the system disk configuration, you must determine which disk units are candidates for reassignment.

Determine the effect of removing disk units from the source ASP by considering the space requirements for the objects to be stored in the source ASP and target user ASPs:

- The current storage requirements for an ASP can be determined from the SST Display Disk Configuration Capacity display.
- If you are planning to start checksum protection for an ASP, consider checksum disk requirements. Checksum protection requires that like types of disk units be grouped in minimum sets of two and can decrease the amount of disk storage available in the ASP. For more information on checksum protection, see Chapter 8, "Introduction to Checksum Protection" on page 8-1.
- If you are planning to start mirrored protection for an ASP, consider the disk requirements for mirrored protection. Mirrored protection increases the number of disk units required to provide a given amount of disk storage. See "Task 1. Calculate Mirrored Capacity Using SST" on page 16-2 for more information on disk unit requirements for mirrored protection.
- Determine storage requirements for objects you intend to create in user ASPs by running test applications, programs, or commands and observing the disk space used. From those observations, you can determine overall storage requirements for the target user ASPs.

- If you are going to use checksum or mirrored protection on the system ASP, you must have a sufficient number of storage units to allow for good system performance.

Chapter 7. Working with Auxiliary Storage Pools

This chapter contains procedures for working with auxiliary storage pools. For general information and considerations for auxiliary storage, see the *Basic Backup and Recovery Guide*, SC41-0036.

Overview of SST and DST Options

Disk storage management procedures are available through system service tools (SST) and dedicated service tools (DST). The SST options are a subset of the options available through DST.

Full function for Work with Disk Unit options is available only through DST before the IPL step for Storage Management Recovery is started. Once Storage Management Recovery is started, the Work with Disk Units options available through DST are identical to the SST subset of options.

System service tools (SST) and dedicated service tools (DST) provide menus for you to use to display and to change your disk configuration. This topic describes the menus and the procedures to use when working with auxiliary storage pools, checksum protection, and mirrored protection. Some of the disk functions can be done by service personnel. Other functions can be performed by you:

- Display Disk Configuration display:
  - Display disk configuration status
  - Display disk configuration capacity
  - Display disk configuration protection
  - Display nonconfigured units
  - Display device parity status
- Work with ASP Configuration display:
  - Display disk configuration capacity
  - Create user ASP
  - Delete user ASP
  - Add disk units to ASP
  - Delete ASP data
  - Change ASP storage threshold
  - Move unit from one ASP to another
  - Remove unit from configuration
- Work with Checksum Protection display:
  - Display checksum configuration
  - Start checksum protection
  - Stop checksum protection
  - Change unprotected storage
  - Calculate checksum configuration
- Work with Mirrored Protection display:
  - Display disk configuration
  - Start mirrored protection
  - Stop mirrored protection
  - Calculate mirrored capacity

- Work with Device Parity Protection display:
  - Display device parity status
  - Prepare to start device parity protection
  - Prepare to stop device parity protection

The following functions are performed by or under the direction of the service representative to analyze and correct hardware problems.

- Analyze disk problem
- Work with Disk Unit Recovery display:
  - Save disk unit data
  - Restore disk unit data
  - Replace configured unit
  - Assign missing unit
  - Recover configuration
  - Disk unit recovery procedures
  - Suspend/resume mirrored protection
  - Copy disk unit data
  - Delete disk unit data
  - Upgrade load source utility
- Work with Disk Unit Information display:
  - Work with vital product data
  - Work with field replacement unit data

Accessing SST Options

SST options are started using the Start System Service Tools (STRSST) command or through the following displays:

- AS/400 Main Menu
- Problem Handling
- SST
- Work with Disk Units from SST menu

The Work with Disk Units display from SST is a different display than the DST full function display because it shows only those options allowed under SST. Figure 7-1 shows the Work with Disk Units display.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

Figure 7-1. Work with Disk Units Display for SST Options

Use SST when your system has already been installed and you want to display your disk configuration, display your checksum configuration (if you have checksum protection), calculate a checksum configuration, plan a system with mirrored pro-

tection, or display, change, or calculate storage thresholds. If your system has mirrored protection, you can also suspend or resume mirrored protection and perform other recovery procedures.

Accessing DST Options

The DST options are started manually during an IPL of the system. Because full function DST options are only available during IPL processing, you must first power down your system, if it is currently running, before starting the IPL procedure.

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Working with User ASPs

The following ASP operations are described in this topic:

- Creating a user ASP and adding units to the new ASP
- Deleting a user ASP
- Adding units to an existing ASP
- Calculating storage requirements using SST
- Moving a disk unit from the system ASP to another ASP
- Moving a disk unit from one user ASP to another
- Removing a disk unit from the system ASP
- Removing a disk unit from a user ASP
- Recovering an overflowed user ASP

Creating a User ASP and Adding Disk Units to the New User ASP

This example adds a new disk unit to the system and creates user ASP 2 to contain it. The threshold limit in this example is set to 88%.

Your service representative attaches the new disk unit. The new unit is in nonconfigured status.

Task Overview

1. Access DST
2. Display Disk Configuration
3. Create the user ASP
4. Add units to the ASP
5. Change Storage Threshold

Warning: A sufficient number of 2800-001 or 2801-001 storage units must be configured to the system ASP (ASP 1) to allow for enough main storage dump space. (See note 3 on page 6-10.)

Task 1. Access DST Options

Sign on to DST using the following steps:

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                        Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit              F12=Cancel
```

Task 2. Display the Disk Configuration

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                              Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display after the disk is attached.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 1 (Display disk configuration) on the Work with Disk Configuration display.

```
                        Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units
     5. Display device parity status
```

Note: Disk units on the system are either configured or nonconfigured. Options 1, 2, and 3 on the Display Disk Configuration display show information about the configured units. Option 4 shows the nonconfigured units attached to the system. When you physically attach a unit to the system, it becomes part of the nonconfigured set until you place it in an ASP. Option 5 shows the status of disk units that are using device parity protection.

4. Select option 1 (Display disk configuration status) and press the Enter key.

```
                    Display Disk Configuration Status

                Serial
ASP  Unit       Number      Type  Model  Address        Status
 1                                                     Unprotected
       1   10-00A7529  9332   400   0010-0000FFFF  Configured
       2   10-00A7529  9332   400   0010-0001FFFF  Configured
       3   10-00A4936  9332   400   0010-0100FFFF  Configured
       4   10-00A4936  9332   400   0010-0101FFFF  Configured
       5   10-00A7498  9332   400   0010-0300FFFF  Configured
       6   10-00A7498  9332   400   0010-0301FFFF  Configured
       7   10-00A7530  9332   400   0010-0400FFFF  Configured
       8   10-00A7530  9332   400   0010-0401FFFF  Configured




Press Enter to continue.

F3=Exit   F5=Refresh  F11=Display disk configuration capacity  F12=Cancel
```

The following fields appear on the Display Disk Configuration Status display:

- *ASP*. The auxiliary storage pool number.
- *Unit*. The number assigned by the system to identify a specific disk unit.
- *Serial Number*. The number assigned by the manufacturer to identify a specific disk unit.
- *Type*. The number assigned by the manufacturer to identify a type of disk unit.
- *Model*. The numbers or letters used to identify the feature level of a specific product type.
- *Address*. Identifies the following:
  - Location of the storage device controller card (columns 1 through 4)
  - Functional controller for the disk unit (columns 5 and 6)
  - Disk unit itself (columns 7 and 8)
  - FFFF (columns 9 through 12)
- *Status*. The valid values for this field are:
  - For ASPs:
    - *Unprotected*. No software protection exists for this ASP.
    - *Checksummed*. The units in the ASP are protected by checksum protection if the units are a part of a checksum set.
    - *Mirrored*. Some or all the units in the ASP are protected by mirrored protection. Any unit in a disk unit subsystem that has device parity protection does not participate in the mirrored protection provided by the software.

    Note: An ASP that is unprotected or has mirrored protection may contain units with device parity protection. The *Status* field for the unit shows if the ASP contains units with device parity protection. The *Status* field for the ASP shows the level of protection (unprotected or mirrored) for the ASP.

- For units in an unprotected ASP.
  - *Configured*.
  - *Device parity*.

    The valid status values for the storage units with device parity protection are:

    DPY/Active. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is fully operational.

    DPY/Failed. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has failed. If another unit in the disk unit subsystem fails, data could be lost.

    DPY/Rebuild. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The data on this unit is being rebuilt from other units in the disk unit subsystem. If another unit in the disk unit subsystem fails, data could be lost.

    DPY/Unprotected. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is operational. However, another unit in the disk unit subsystem has failed or is being rebuilt. If another unit in the disk unit subsystem fails, data could be lost.

    DPY/HDW Failure. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A hardware-related failure has occurred. The failure does not affect data or performance. However, an exposure to an outage exists if another failure of a redundant component, such as a power supply, occurs.

    DPY/Degraded. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A decrease in performance has occurred because a component that is not critical has failed. The failed component needs to be repaired or replaced.

    DPY/Power Loss. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has lost power.

    DPY/Not Ready. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The unit is not ready to perform I/O operations.

    DPY/Unknown. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The status of this unit is not known to the system.
- For units in an ASP that has checksum protection.
  - *Checksummed*. Indicates that the unit is part of a checksum set.
  - *Configured*. Indicates that the unit is not part of a checksum set.
- For units in a mirrored ASP.

- *Active*. This unit is capable of having data written to it, or read from it.
- *Suspended*. This unit is not capable of having data written to it, or read from it. The data on this unit is not current. For example, if the disk needs repair action or has been manually suspended, it would be in a *Suspended* state.
- *Resuming*. The current data is being copied (or will be copied) to this unit from the other active unit of the mirrored pair.
- *Unknown*. The system configuration mechanism cannot determine what the valid configuration should be.

Note: Units in an ASP with mirrored protection can have device parity protection. However, these units do not participate in the mirrored protection provided by the system software.

5. Press F11 (Display non-configured unit) three times to display non-configured units.

```
                    Display Non-Configured Units

Serial
Number       Type   Model   Address         Status
10-00A7503   9332     400   0010-0100FFFF   Non-configured
10-00A7503   9332     400   0010-0101FFFF   Non-configured
10-00A3651   9332     400   0010-0400FFFF   Non-configured
10-00A3651   9332     400   0010-0401FFFF   Non-configured
```

6. Write down the serial number and address of the units that you are going to use.
7. Return to the Work with Disk Configuration display by pressing F12 (Cancel) two times.

```
                      Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

Task 3. Create the User ASP

1. Select option 3 (Work with ASP configuration) on the Work with Disk configuration display.

```
                        Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

2. Select option 2 (Create user ASP) on the Work with ASP Configuration display.

```
                          Create User ASP

Type choice, press Enter.

   ASP number . . . . . .    __     2-16
```

3. Enter an ASP number in the *ASP number* prompt on the Create User ASP display and press the Enter key.

Task 4. Adding Disk Units to the New User ASP

1. After you enter an ASP number, the Select Units to Add to ASP display appears.

```
                       Specify ASPs to Add Units to

Specify the ASP to add each unit to.

Specify  Serial
  ASP    Number      Type  Model  Address
   _     10-00A7503  9332    400    0010-0100FFFF
   _     10-00A7503  9332    400    0010-0101FFFF
   _     10-00A3651  9332    400    0010-0400FFFF
   _     10-00A3651  9332    400    0010-0401FFFF
```

2. Use the serial number of the units you wrote down when you displayed the nonconfigured units to select the units to place within the specified ASP. Type a 1 in the *Option* column for each unit and press the Enter key.

   Note: Each storage unit within a replaceable disk unit will have the same serial number.

   The Confirm Add Units display is shown. The Confirm Add Units display shows what the entire system configuration will be when you add the units. Use the serial number of the disk unit you wrote down when you displayed the disk configuration to verify that you are selecting the correct disk units to add to the ASP.

```
                              Confirm Add Units

Add will take several minutes for each unit.  The system will
have the displayed protection after the unit(s) are added.

Press F9=Capacity Information to display the resulting capacity.
Press Enter to confirm your choice for 1=Add units.
Press F12=Cancel to return and change your choice.


            Serial
ASP  Unit   Number       Type  Model  Address          Protection       CSS
 2                                                    Unprotected
       9    10-00A3651   9332    400  0010-0400FFFF    Unprotected
      10    10-00A3651   9332    400  0010-0401FFFF    Unprotected
```

3. If you are satisfied with the configuration, press the Enter key to add the disk units to the ASP.

   Note: The system assigns a threshold value of 90% for newly created user ASPs.

   Adding units takes several minutes, during which the system may appear inactive. The time it takes depends on the size of each unit being added and the ability of the system to do multiple adds at the same time.

Task 5. Changing the Storage Threshold of the New User ASP

1. After the add operation is complete, return to the Work with ASP Configuration display.

```
                          Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

2. Select option 6 (Change ASP storage threshold) on the Work with ASP Configuration display. The Select ASP display is shown.

```
                                Select ASP

Type option, press Enter.
  1=Select

                                          --Protected--   --Unprotected--
Option  ASP  Threshold  Overflow     Size  %Used      Size  %Used
  _       1      90%       No          0  0.00%       627  69.15%
  _       2      90%       No          0  0.00%       855   0.25%
```

3. Select the ASP you want to change the threshold for on the Select ASP display and press the Enter key. The following display is shown.

```
                          Change Storage Threshold

                            --Protected--  --Unprotected--
ASP   Threshold  Overflow     Size   %Used     Size  %Used
 2          90%    No           0    0.00%     855   0.25%

This is an unprotected ASP.  The threshold represents the amount
of unprotected storage used before a warning message is sent to
the system operator.

Type choice, press Enter.

  New threshold . . . . . .     88% 1-100




F3=Exit    F11=Display disk configuration status   F12=Cancel
```

Figure 7-2. Change Storage Threshold Display

4. Change the threshold value from the default value of 88% to the value you want.

   The display used to change the storage threshold has the following fields:

   - *ASP*. The auxiliary storage pool number.
   - *Threshold*. The current threshold. The system notifies you when this percentage of storage is full.
   - *Overflow*. Indicates if this ASP has overflowed.
   - *Protected Size*. Amount of storage in the ASP that is protected by checksum or mirrored protection.
   - *Protected % Used*. Percent of protected size that is used.
   - *Unprotected Size*. Amount of storage in the ASP that is unprotected.
   - *Unprotected % Used*. Percent of unprotected size that is used.

- *New threshold.* The field used to enter a new threshold value. The new value must be higher than the current amount used or a warning message is shown or sent to QSYSOPR.

5. Return to the Work with ASP Configuration display.
6. Select option 1 (Display disk configuration capacity).
7. Verify the configuration on the Display Disk Configuration Capacity display.

```
                    Display Disk Configuration Capacity

                                                    --Protected--  --Unprotected--
ASP  Unit  Type  Model  Threshold  Overflow   Size  %Used    Size  %Used
 1                          90%    No            0   0.00%   1600  52.70%
       1  9332   400                           0   0.00%    200  98.20%
       2  9332   400                           0   0.00%    200  45.40%
       3  9332   400                           0   0.00%    200  55.60%
       4  9332   400                           0   0.00%    200  56.00%
       5  9332   400                           0   0.00%    200  59.00%
       6  9332   400                           0   0.00%    200  57.00%
       7  9332   400                           0   0.00%    200  21.70%
       8  9332   400                           0   0.00%    200  28.90%
```

Adding Units to an Existing ASP

This example adds a new disk unit to the system ASP or a user ASP.

Your service representative attaches the new disk unit. The new unit is in nonconfigured status.

Task Overview

You will perform the following steps during this task

1. Access DST
2. Display Disk Configuration
3. Add Unit to existing ASP

Warning: A sufficient number of 2800-001 storage units must be configured to the system ASP (ASP 1) to allow for enough main storage dump space. (See note 3 on page 6-10.)

Attention!

You cannot not add units that have device parity protection to an ASP that has checksum protection.

Task 1. Access DST Options

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

```
ENDSBS SBS(*ALL) OPTION(*IMMED)
```

Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.

4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Task 2. Display the Disk Configuration

You will use the information in step 6 when adding the units to the existing ASP.

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display after the disk is attached.

```
                      Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 1 (Display disk configuration) on the Work with Disk Configuration display.

```
                        Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units
     5. Display device parity status
```

Note: Disk units on the system are either configured or nonconfigured. Options 1, 2, and 3 on the Display Disk Configuration display show information about the configured units. Option 4 shows the nonconfigured units attached to the system. When you physically attach a unit to the system, it becomes part of the nonconfigured set until you place it in an ASP. Option 5 shows the status of disk units that are using device parity protection.

4. Select option 1 (Display disk configuration status) and press the Enter key.

```
                      Display Disk Configuration Status

                Serial
ASP  Unit       Number      Type  Model  Address         Status
 1                                                      Unprotected
        1       10-00A7529  9332    400  0010-0000FFFF   Configured
        2       10-00A7529  9332    400  0010-0001FFFF   Configured
        3       10-00A4936  9332    400  0010-0100FFFF   Configured
        4       10-00A4936  9332    400  0010-0101FFFF   Configured
        5       10-00A7498  9332    400  0010-0300FFFF   Configured
        6       10-00A7498  9332    400  0010-0301FFFF   Configured
        7       10-00A7530  9332    400  0010-0400FFFF   Configured
        8       10-00A7530  9332    400  0010-0401FFFF   Configured



Press Enter to continue.

F3=Exit   F5=Refresh  F11=Display disk configuration capacity  F12=Cancel
```

The following fields appear on the Display Disk Configuration Status display:

- *ASP*. The auxiliary storage pool number.
- *Unit*. The number assigned by the system to identify a specific disk unit.
- *Serial Number*. The number assigned by the manufacturer to identify a specific disk unit.
- *Type*. The number assigned by the manufacturer to identify a type of disk unit.
- *Model*. The numbers or letters used to identify the feature level of a specific product type.
- *Address*. Identifies the following:

- Location of the storage device controller card (columns 1 through 4)
- Functional controller for the disk unit (columns 5 and 6)
- Disk unit itself (columns 7 and 8)
- FFFF (columns 9 through 12)

- *Status*. The valid values for this field are:
  - For ASPs:
    - *Unprotected*. No software protection exists for this ASP.
    - *Checksummed*. The units in the ASP are protected by checksum protection if the units are a part of a checksum set.
    - *Mirrored*. Some or all the units in the ASP are protected by mirrored protection. Any unit in a disk unit subsystem that has device parity protection does not participate in the mirrored protection provided by the software.

    Note: An ASP that is unprotected or has mirrored protection may contain units with device parity protection. The *Status* field for the unit shows if the ASP contains units with device parity protection. The *Status* field for the ASP shows the level of protection (unprotected or mirrored) for the ASP.
  - For units in an unprotected ASP.
    - *Configured*.
    - *Device parity*.

      The valid status values for the storage units with device parity protection are:

      DPY/Active. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is fully operational.

      DPY/Failed. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has failed. If another unit in the disk unit subsystem fails, data could be lost.

      DPY/Rebuild. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The data on this unit is being rebuilt from other units in the disk unit subsystem. If another unit in the disk unit subsystem fails, data could be lost.

      DPY/Unprotected. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is operational. However, another unit in the disk unit subsystem has failed or is being rebuilt. If another unit in the disk unit subsystem fails, data could be lost.

      DPY/HDW Failure. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A hardware-related failure has occurred. The failure does not affect data or performance. However, an exposure to an outage exists if another failure of a redundant component, such as a power supply, occurs.

DPY/Degraded. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A decrease in performance has occurred because a component that is not critical has failed. The failed component needs to be repaired or replaced.

DPY/Power Loss. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has lost power.

DPY/Not Ready. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The unit is not ready to perform I/O operations.

DPY/Unknown. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The status of this unit is not known to the system.

- For units in an ASP that has checksum protection.
  - *Checksummed*. Indicates that the unit is part of a checksum set.
  - *Configured*. Indicates that the unit is not part of a checksum set.
- For units in a mirrored ASP.
  - *Active*. This unit is capable of having data written to it, or read from it.
  - *Suspended*. This unit is not capable of having data written to it, or read from it. The data on this unit is not current. For example, if the disk needs repair action or has been manually suspended, it would be in a *Suspended* state.
  - *Resuming*. The current data is being copied (or will be copied) to this unit from the other active unit of the mirrored pair.
  - *Unknown*. The system configuration mechanism cannot determine what the valid configuration should be.

Note: Units in an ASP with mirrored protection can have device parity protection. However, these units do not participate in the mirrored protection provided by the system software.

5. Press F11 (Display non-configured unit) three times to display non-configured units.

```
                    Display Non-Configured Units

Serial
Number       Type   Model   Address         Status
10-00A7503   9332    400    0010-0100FFFF   Non-configured
10-00A7503   9332    400    0010-0101FFFF   Non-configured
10-00A3651   9332    400    0010-0400FFFF   Non-configured
10-00A3651   9332    400    0010-0401FFFF   Non-configured
```

6. Write down the serial number and address of the units that you are going to use.

7. Return to the Work with Disk Configuration display by pressing F12 (Cancel) two times.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

Task 3. Add Units to an Existing ASP

1. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

2. Select option 4 (Add units to existing ASP) on the Work with Disk Configuration display.

```
                      Specify ASPs to Add Units to

Specify the ASP to add each unit to.

Specify  Serial
  ASP    Number       Type  Model  Address
   __    10-00A7503   9332    400    0010-0100FFFF
   __    10-00A7503   9332    400    0010-0101FFFF
   __    10-00A3651   9332    400    0010-0400FFFF
   __    10-00A3651   9332    400    0010-0401FFFF
```

3. Enter the ASP number on the Specify ASPs to Add Units to display and press the Enter key.

   Use the serial number of the units you wrote down when you displayed the nonconfigured units to select the units to place within the specified ASP.

   Note: Storage units within the same replaceable disk unit have the same serial number.

```
                              Confirm Add Units

Add will take several minutes for each unit.  The system will
have the displayed protection after the unit(s) are added.

Press F9=Capacity Information to display the resulting capacity.
Press Enter to confirm your choice for 1=Add units.
Press F12=Cancel to return and change your choice.


             Serial
 ASP  Unit   Number       Type  Model  Address          Protection       CSS
  2                                                    Unprotected
        9    10-00A3651   9332    400  0010-0400FFFF    Unprotected
       10    10-00A3651   9332    400  0010-0401FFFF    Unprotected
```

The Confirm Add Units display shows what the entire system configuration will be when you add the units. Use the serial number of the unit you wrote down when you displayed the disk configuration to verify that you are selecting the correct units to add to the ASP.

4. If you are satisfied with the configuration, press the Enter key to add the disk units to the ASP.

   Adding units can take from several minutes to several hours. During that time, the system may appear inactive. The time it takes the system to add units depends on the type, model, and size of each unit being added and the ability of the system to do multiple adds at the same time.

5. If you have no other DST functions to perform, press F3 (Exit) until you return to the Use Dedicated Service Tools (DST) menu.
6. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools (DST) menu and press the Enter key.
7. Select option 1 (Perform an IPL) on the IPL or Install the System menu and press the Enter key.

Deleting a User ASP

Deleting an ASP causes all units within the ASP to be removed from the configuration and all data in the ASP to be destroyed. You cannot delete the system ASP. A warning message appears, providing the opportunity to cancel the deletion. You may save data before deleting a user ASP.

Before deleting a user ASP, be sure to delete all the objects in the user ASP. Deleting the objects allows the operating system to correctly update the control information and avoids references to destroyed or damaged objects. The Display Object Description (DSPOBJD) command can be used to determine the ASP storage for an object that has been allocated.

To delete a user ASP, do the following:

Task 1. Delete the Objects in the User ASP

Delete the objects in the user ASP. If the object is a physical file, you must delete the associated logical files first, and then delete the physical file. Do the following:

1. Display the library in the user ASP:

   ```
   WRKLIB LIB(library-name)
   ```

2. Type a 12 (Work with objects) in the *Opt* column and press the Enter key.
3. Find the objects to be deleted in the *Object* column.
4. Type a 4 (Delete) in the *Opt* column for each object you want to delete.
5. Press the Enter key.

Task 2. Access DST Options

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .   _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                         Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Task 3. Delete the User ASP

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                              Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 3 (Delete user ASP) on the Work with ASP Configuration display and press the Enter key.

```
                              Delete User ASP

Type option, press Enter
   4=Delete

                                        --Protected--  --Unprotected--
Option  ASP  Threshold  Overflow   Size  %Used     Size  %Used
          1       90%    No        600  77.84%       0  0.00%
          2       90%    No          0   0.00%     200  0.53%
          3       90%    No          0   0.00%     200  0.53%
```

5. Type a 4 in the *Option* field of the ASP you want to delete and press the Enter key. The Confirm Delete of User ASP display is shown.

```
                        Confirm Delete Of User ASP

Warning: Deleting a user ASP will remove all units of that
ASP from the configuration.  The units will become nonconfigured.

Press F10 to confirm your choice for 4=delete
Press F12=Cancel to return to change your choice

                                    --Protected--  --Unprotected--
Option  ASP  Threshold  Overflow    Size  %Used     Size  %Used
  4      2       90%    No             0   0.00%     200  0.53%
```

6. Press F10 (Confirm) to confirm that delete of the ASP. The delete operation may take several minutes.
7. If you have no other DST functions to perform, press F3 (Exit) until you return to the Use Dedicated Service Tools (DST) menu.
8. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools (DST) menu and press the Enter key.
9. Select option 1 (Perform an IPL) on the IPL or Install the System menu and press the Enter key.

Considerations for Moving or Removing a Disk Unit from an ASP

When planning to move or remove a disk unit from an ASP, consider the amount of storage used in the source ASP. If the source ASP has sufficient storage, the data on the disk unit being moved or removed is moved to other storage units in the ASP before the move or remove takes place. As a result, data loss is prevented and a restore of the data in the ASP is not required.

Note: As it is with any ASP management function, it is recommended that you have a current save of your entire system before performing a move or remove operation.

It is recommended that you determine if the ASP has sufficient storage before starting the move or remove operation. (See "Determining the Total Amount of Storage Used in the ASP" on page 7-26 to find out how much storage an ASP has.)

Consider the following before moving or removing a disk unit from an ASP:

- Unit 1 (load source) in the system ASP cannot be moved or removed.
- Units can be moved to or removed from the system ASP even if data in a user ASP has overflowed into the system ASP.
- Units can be moved or removed from an overflowed user ASP.
- More than one unit can be moved or removed at the same time. However, units cannot be moved into and out of the same ASP at the same time.
- A single storage unit cannot be moved to or from an ASP that has mirrored protection.
- Both units in a mirrored pair must be removed at the same time.
- Only a unit not in a checksum set can be moved or removed from an ASP that has checksum protection.

- If the ASP does not have sufficient storage to move data from the disk unit being moved or removed to other units in the ASP, an error display is shown indicating that the move or remove cannot take place because there is insufficient storage.

Determining the Total Amount of Storage Used in the ASP

In the following example, assume four 9332 Model 400 disk units are in the system ASP for a total of eight storage units. Storage units 1 and 2 are in a 9332 Model 400. Storage units 3 and 4 are in another 9332 Model 400 disk unit, and so on. Each storage unit contains 200MB for a total of 1600MB in the system ASP. In this example, the system currently uses 52.7% of the available disk space. The calculations in this example assume two storage units (400MB) are being moved or removed.

To determine the amount of storage used in the system ASP, do the following:

1. Type the following and press the Enter key to display the System Service Tool (SST) menu.

   ```
   STRSST
   ```

```
                    System Service Tools (SST)

Select one of the following:

     1. Start a service tool
     2. Work with active service tools
     3. Work with disk units
     4. Work with diskette data recovery
```

2. Select option 3 (Work with disk units) on the System Service Tool (SST) menu.

```
                          Work with Disk Units

Select one of the following:

     1. Display disk configuration
     2. Display checksum configuration
     3. Calculate checksum configuration
     4. Work with ASP threshold
     5. Work with disk unit recovery
     6. Work with disk unit information
     7. Calculate mirrored capacity
```

3. Select option 1 (Display disk configuration) on the Work with Disk Units display.

```
                        Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units
     5. Display device parity status
```

4. Select option 2 (Display disk configuration capacity) on the Display Disk Configuration display.

```
                    Display Disk Configuration Capacity

                                                    --Protected--  --Unprotected--
ASP  Unit  Type  Model  Threshold  Overflow  Size  %Used    Size  %Used
 1                          90%    No           0   0.00%   1600  52.70%
       1   9332   400                          0   0.00%    200  98.20%
       2   9332   400                          0   0.00%    200  45.40%
       3   9332   400                          0   0.00%    200  55.60%
       4   9332   400                          0   0.00%    200  56.00%
       5   9332   400                          0   0.00%    200  59.00%
       6   9332   400                          0   0.00%    200  57.00%
       7   9332   400                          0   0.00%    200  21.70%
       8   9332   400                          0   0.00%    200  28.90%
```

The Display Disk Configuration display provides the following information:

- *ASP*. The auxiliary storage pool number.
- *Unit*. The number assigned by the system to identify the storage unit.
- *Type*. The disk unit type assigned by the manufacturer.
- *Model*. The disk model that identifies the feature level for a specific type of disk unit.
- *Threshold*. The current threshold. The system notifies you when this percentage of storage is full.
- *Overflow*. Indicates if this ASP is overflowed.
- *Protected Size*. Amount of storage in the ASP that is protected by checksum or mirrored protection.
- *Protected % Used*. Percent of protected size that is used.
- *Unprotected Size*. Amount of storage in the ASP that is unprotected.
- *Unprotected % Used*. Percent of unprotected size that is used.

5. Multiply the size of the ASP in the *unprotected column* by the number in the *%Used* column for the ASP. For example:

   1600 X 52.70% = 843.20MB or 1600 X .5270 = 843.20MB

6. Divide the total in step 5 by 1200MB (1600MB minus the 400MB being moved or removed) to determine the resulting percentage used for the ASP after the disk unit (two storage units) are moved or removed.

```
843.2MB/1200MB  =  .7026 X 100 = 70.26%
```

7. The percentage calculated for the ASP in step 6 on page 7-27 should not exceed the storage threshold for the ASP. If the calculation is over the threshold specified for the ASP, consider adding disk units. The system will not allow you to remove a unit from an ASP that results in exceeding the threshold for the ASP.

   Note: If you did not change the threshold for the system ASP or created a user ASP without specifying a threshold, the default value of 90% is used. The default value can be changed.

Moving a Disk Unit from an Existing ASP that Has Sufficient Storage

Option 7 (Move unit from one ASP to another) on the Work with ASP Configuration display allows you to move storage units from one ASP to another. However, the source ASP must have sufficient storage to allow the data on the selected storage units to be moved to other storage units within the source ASP. To determine if an ASP has sufficient unused storage, see the topic, "Determining the Total Amount of Storage Used in the ASP" on page 7-26.

Use this procedure if any of the following conditions apply:

- A new ASP needs to be created from the existing disk configuration without adding new disk units.
- Disk units need to be moved because they were added accidentally to the wrong ASP.
- An ASP needs more storage (for example, the ASP has reached its storage threshold or has overflowed) and there is sufficient storage in another ASP to move units from that ASP
- A disk unit not in a checksum set needs to be moved to a different ASP.
- The disk configuration needs to be adjusted between the existing ASPs to allow the start of checksum or mirrored protection.
- There is not a sufficient number of 2800-001 or 2801-001 storage units configured to the system ASP (ASP 1) to allow for enough main storage dump space. (See note 3 on page 6-10.)

Restrictions:

- The unit to be moved must not be in a checksum set. (A unit not in a checksum set can be moved from an ASP that has checksum protection. To move a unit in a checksum set, checksum protection for the ASP must be stopped, the units moved, and then checksum protection started again.)
- There must be sufficient unused storage in the source ASP (user or system) to move the data from the selected disk units to the remaining disk units in the source ASP. To determine if the ASP has sufficient storage, see "Determining the Total Amount of Storage Used in the ASP" on page 7-26.
- Units cannot be moved to or from an ASP that has mirrored protection. Both units of a mirrored pair must first be removed from the ASP. The units can then be added to one or more different ASPs.

- A sufficient number of 2800-001 storage units must be configured to the system ASP (ASP 1) to allow for enough main storage dump space. (See note 3 on page 6-10.)

Task Overview

In this procedure you will do the following:

1. Access Dedicated Service Tools (DST)
2. Move units from one ASP to another

The following example moves a disk unit from the system ASP to another ASP. Because the target user ASP does not exist, the system will create it with a default storage threshold of 90%.

Task 1. Access DST Options

Do the following steps to access the DST options.

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                        Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Task 2. Move the Disk Unit

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools (DST) menu.

```
                              Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 7 (Move units from one ASP to another) on the Work with ASP Configuration display. The Specify ASP to Move Units From display is shown.

```
                      Specify ASP to Move Disk Units

To move units to different ASPs, specify the ASP that you want
to move each one to in the 'New ASP' field.

Specify the units to be moved, press Enter.

New  Current        Serial                      --Protected-- --Unprotected--
ASP    ASP    Unit  Number      Type  Model    Size  %Used    Size   %Used
         1                                                   1600 52.70%
                1   10-00A7529  9332   400       0   0.00%    200 98.20%
                2   10-00A7529  9332   400       0   0.00%    200 45.50%
                3   10-00A4936  9332   400       0   0.00%    200 55.60%
                4   10-00A4936  9332   400       0   0.00%    200 56.00%
 2              5   10-00A7498  9332   400       0   0.00%    200 59.00%
 2              6   10-00A7498  9332   400       0   0.00%    200 57.00%
                7   10-00A7530  9332   400       0   0.00%    200 21.00%
                8   10-00A7530  9332   400       0   0.00%    200 28.90%


 F3=Exit   F5=Refresh       F11=Display disk configuration capacity
 F12=Cancel
```

5. Type the number of the ASP you want to move the units to in the *New ASP* column and press the Enter key.

The Confirm Continuation display is shown before the Confirm Move of Unit display if the system ASP has checksum protection, the source ASP has mirrored protection, or the storage management directories are not usable.

```
                              Confirm Continuation

In order to proceed the system must perform internal processing
that may take several minutes during which the system may appear
inactive.  Once the processing starts only the Work with Disk
Configuration operation will be available until an IPL of
the system to DST is performed.

Press Enter to continue.
Press F12=Cancel to return and change your choice.
```

6. Press the Enter key.

```
                              Confirm Move of Unit

Moving units will take several minutes.

Press F9=Capacity information to display the capacity information,
Press Enter to confirm your choice to move the units.
Press F12=Cancel to return to change your choice.

New  Current        Serial                      --Protected-- --Unprotected--
ASP    ASP    Unit  Number      Type  Model    Size  %Used    Size   %Used
         1                                      0   0.00%    1600 52.70%
                 1   10-00A7529  9332   400      0   0.00%     200 98.20%
                 2   10-00A7529  9332   400      0   0.00%     200 45.50%
                 3   10-00A4936  9332   400      0   0.00%     200 55.60%
                 4   10-00A4936  9332   400      0   0.00%     200 56.00%
                 7   10-00A7530  9332   400      0   0.00%     200 21.00%
                 8   10-00A7530  9332   400      0   0.00%     200 28.90%
           2                                    0   0.00%     400  0.00
                 5   10-00A7498  9332   400      0   0.00%     200 00.00%
                 6   10-00A7498  9332   400      0   0.00%     200 00.00%

F9=Capacity information  F12=Cancel
```

7. Press F9 (Capacity information) to display the resulting capacity.

```
                              Resulting Capacity

The configuration change that you requested would result in the
following ASP capacities.

Press Enter to continue.

                ----------Current----------  ----------Modified----------
                --Protected--  -Unprotected-  --Protected--  -Unprotected-
ASP   Threshold   Size  %Used   Size  %Used   Size   %Used    Size   %Used
 1        90%        0   0.00%   1600 52.70%     0    0.00%   1200   70.26%
 2        90%        0   0.00%      0  0.00%     0    0.00%    400    0.52%
```

8. Press the Enter key to return to the Work with ASP Configuration display.
9. Press the Enter key on the Confirm Move of Units display to move the selected units. The system will move the data off the selected units to the remaining

units in the source ASP. The move can take several minutes during which the system appears inactive.

10. When the move operation is complete, you return to the Work with ASP Configuration display.
11. If you have no other DST functions to perform, press F3 (Exit) until you return to the Use Dedicated Service Tools (DST) menu.
12. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools (DST) menu and press the Enter key.
13. Select option 1 (Perform an IPL) on the IPL or Install the System menu and press the Enter key.

Removing a Disk Unit from an ASP that Has Sufficient Storage

Option 8 (Remove unit from configuration) on the Work with ASP Configuration display allows you to remove storage units from the configuration. However, the source ASP must have sufficient unused storage to allow the data on the storage units being removed to be moved to other storage units in the source ASPs. To determine if an ASP has sufficient storage, see the topic, "Determining the Total Amount of Storage Used in the ASP" on page 7-26.

Use this procedure if any of the following conditions apply:

- Disk units need to be removed from one system and then added to a different system.
- A disk unit is failing and there is no replacement.
- Disk units need to be upgraded to a different type. (Upgrading can be accomplished by first adding the new disk units and then removing the old ones.) Unit 1 (load source) cannot be removed on Version 2 hardware (2800-001 disk unit). Unit 1 can be removed on Version 1 hardware using the Upgrade Load Source Utility.

Restrictions:

- The unit to be removed is not in a checksum set. A unit not in a checksum set can be removed from an ASP that has checksum protection. To remove a unit in a checksum set, checksum protection for the ASP must be stopped before the unit can be removed.
- Both units in a mirrored pair must be removed at the same time. The system will not let you select only one unit in a mirrored pair. The system automatically selects the other unit.
- The ASP has sufficient storage to move the data from the selected disk unit to the remaining disk units in the source ASP. To determine if the ASP has sufficient storage to remove a disk unit, see the topic "Determining the Total Amount of Storage Used in the ASP" on page 7-26.
- A sufficient number of 2800-001 storage units must be configured to the system ASP (ASP 1) to allow for enough main storage dump space. (See note 3 on page 6-10.)

Task Overview

In this procedure, you will do the following:

1. Access Dedicated Service Tools (DST)
2. Remove the units from the configuration

The following example removes a disk unit from the system ASP that has sufficient storage.

Task 1. Access DST Options

Do the following steps to access the DST options.

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                         Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     _

F3=Exit              F12=Cancel
```

Task 2. Remove the Disk Unit

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools (DST) menu.

```
                              Work with Disk Units

 Select one of the following:

      1. Work with disk configuration
      2. Analyze disk device problem
      3. Work with disk unit recovery
      4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 8 (Remove units from configuration) on the Work with ASP Configuration display.

```
                      Remove Units from Configuration

Type options, press Enter.
  4=Remove unit from configuration


                Serial
OPT  Unit  ASP  Number      Type   Model   Address        Status
       2    1   10-00A7529  9332    400    0010-0001FFFF  Configured
       3    1   10-00A4936  9332    400    0010-0100FFFF  Configured
       4    1   10-00A4936  9332    400    0010-0101FFFF  Configured
 4     5    1   10-00A7498  9332    400    0010-0300FFFF  Configured
 4     6    1   10-00A7498  9332    400    0010-0301FFFF  Configured
       7    1   10-00A7530  9332    400    0010-0400FFFF  Configured
       8    1   10-00A7530  9332    400    0010-0401FFFF  Configured



 Press Enter to continue.

 F3=Exit     F5=Refresh    F11=Non-configured units   F12=Cancel
```

5. Type a 4 (Remove unit from configuration) in the *OPT* column for each unit you want to remove and press the Enter key.

The Confirm Continuation display may be shown before the Confirm Remove of Unit display if the ASP has checksum protection or the storage management directories are not usable.

```
                              Confirm Continuation

In order to proceed the system must perform internal processing
that may take several minutes during which the system may appear
inactive.  Once the processing starts only the Work with Disk
Configuration operation will be available until an IPL of
the system to DST is performed.

Press Enter to continue.
Press F12=Cancel to return and change your choice.
```

6. Press the Enter key.

```
                              Confirm Remove of Units

Removing disk units will take several minutes.

Press Enter to confirm remove of disk units.
Press F9=Capacity information to display the capacity information.
Press F12=Cancel to return to change your choice.


                  Serial
OPT  Unit  ASP    Number       Type   Model   Address        Status
 4     5    1    10-00A7498   9332    400    0010-0300FFFF  Configured
 4     6    1    10-00A7498   9332    400    0010-0301FFFF  Configured
```

7. Press F9 (Capacity information) to display the resulting capacity.

```
                              Resulting Capacity

The configuration change that you requested would result in the
following ASP capacities.

Press Enter to continue.

                ----------Current----------  ----------Modified----------
                --Protected--  -Unprotected-  --Protected--  -Unprotected-
ASP   Threshold   Size  %Used   Size  %Used   Size   %Used    Size   %Used
 1        90%         0   0.00%   1600 52.70%     0    0.00%   1200   70.26%
```

8. Press the Enter key to return to the Confirm Remove of Units display.
9. Press the Enter key on the Confirm Remove of Units display to remove the selected units. The system moves the data off the units selected to be removed to the remaining units in the source ASP. The remove can take several minutes or several hours during which the system appears inactive.

Notes:

a. The time it takes to remove a unit depends on the disk unit type and model.

b. If the data on the unit being removed is severely fragmented and the amount of storage used is low, the remove operation could take several hours.

10. When the remove operation is complete, you return to the Work with ASP Configuration display.
11. If you have no other DST functions to perform, press F3 (Exit) until you return to the Use Dedicated Service Tools (DST) menu.
12. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools (DST) menu and press the Enter key.
13. Select option 1 (Perform an IPL) on the IPL or Install the System menu and press the Enter key.

Removing a Failed Disk Unit from the System ASP

Use this procedure if an immediate replacement for a failed disk unit in the system ASP is not available. When the disk unit is removed, all the data in the system ASP is cleared. The operating system and most of the user data can be restored. When the replacement disk unit arrives, the data can be added back to the system ASP.

Restriction:

- A sufficient number of 2800-001 storage units are configured to the system ASP (ASP 1) to allow for enough main storage dump space. (See note 3 on page 6-10.)

Task Overview

In this procedure, you will do the following:

1. Access DST
2. Delete ASP data
3. Remove unit
4. Restore the operating system
5. Restore user data

The following example removes a failed disk unit from the system ASP. Storage is not sufficient to restore all the objects to the system.

Attention!

You must determine how much storage will not be available. After you determine the amount of storage not available, you must determine which objects or libraries are not critical to your daily operations. Use the DSPOBJD command to determine the amount of storage for the objects that are not critical in the ASP. The amount of storage used by the objects or libraries must be equal to or more than the amount of storage that will not be available.

You must determine if slower performance is better than the cost of the system not being available.

Task 1. Access DST Options

When a disk unit fails, the operating system functions are not available. Therefore, you cannot access DST the normal way. To access DST, do the following:

1. Turn off the power to the system by pushing the Power switch down. The switch returns to center after you push it.
2. Ensure the key is in the keylock switch on the control panel.
3. Ensure the keylock switch is in the Manual position.
4. Press the Select switch until function code **02** is shown in the Function display on the control panel.
5. Press the Enter button on the control panel.
6. Select IPL type A by pressing the Select switch until **A** is shown on the Data display.
7. Press the Enter button on the control panel.
8. When the system has powered down, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

9. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

10. Sign on DST with the DST security-level or full-level password. *Security Reference*, SC41-8083, has more information about DST passwords.

    The Use Dedicated Service Tools (DST) menu is shown.

```
                        Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit              F12=Cancel
```

Task 2. Delete the ASP Data

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                              Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk configuration display.

```
                    Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 5 (Delete ASP data) on the Work with ASP Configuration display.

   Note: Selecting this option deletes all data in the system ASP. Do not use this procedure unless you have a failed disk unit and there is no immediate replacement for the disk unit.

```
                    Select ASP to Delete Data From

Type options, press Enter
  4=Delete ASP data

Option  ASP  Threshold  Overflow    --Protected--  --Unprotected
                                      Size  %Used     Size  %Used
          1       90%        No       0.00  0.00%    1200  74.84%
          2       90%       Yes       0.00  0.00%     200  99.99%
          3       90%       Yes       0.00  0.00%     200  99.99%
```

5. Type a 4 in the *Option* column to select the ASP you want to delete the data from and press the Enter key. The following display is shown.

```
                          Confirm Delete ASP Data

Warning:  All data will be deleted from the selected ASPs.  You
have selected to delete data from ASP 1.  This will prevent you
from changing the disk configuration in some ways until the system
is IPLed again to DST.

Press F10 to confirm your choice for 4=Delete ASP data.
Press F12=Cancel to return to change your choice.

                                      --Protected--  --Unprotected--
Option  ASP  Threshold  Overflow     Size  %Used     Size  %Used
  4      1      90%       No           0   0.00     1200    *
```

6. Press F10 (Confirm) to confirm your choice to delete the ASP data.
7. When the delete of the ASP data is complete, you return to the Use Dedicated Service Tools (DST) menu.

```
                        Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit           F12=Cancel
```

Task 3. Remove the Disk Unit from the ASP

To remove the disk unit from the ASP, do the following:

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools (DST) menu.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                       Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                          Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 8 (Remove units from configuration) on the Work with ASP Configuration display.

```
                        Remove Units from Configuration

Type options, press Enter.
  4=Remove unit from configuration


                  Serial
OPT  Unit  ASP    Number      Type   Model   Address        Status
       2    1   10-00A7529  9332    400    0010-0001FFFF  Configured
       3    1   10-00A4936  9332    400    0010-0100FFFF  Configured
       4    1   10-00A4936  9332    400    0010-0101FFFF  Configured
 4     5    1   10-00A7498  9332    400    0010-0300FFFF  Configured
 4     6    1   10-00A7498  9332    400    0010-0301FFFF  Configured
       7    1   10-00A7530  9332    400    0010-0400FFFF  Configured
       8    1   10-00A7530  9332    400    0010-0401FFFF  Configured



 Press Enter to continue.

 F3=Exit     F5=Refresh    F11=Non-configured units   F12=Cancel
```

5. Type a 4 (Remove unit from configuration) in the *OPT* column for each unit you want to remove and press the Enter key.
6. The Confirm Remove of Unit display is shown.

```
                            Confirm Remove of Units

Removing disk units will take several minutes.

Press Enter to confirm remove of disk units.
Press F9=Capacity information to display the capacity information.
Press F12=Cancel to return to change your choice.


                  Serial
OPT  Unit  ASP    Number      Type   Model   Address        Status
 4     5    1   10-00A7498  9332    400    0010-0300FFFF  Configured
 4     6    1   10-00A7498  9332    400    0010-0301FFFF  Configured
```

7. Press F9 (Capacity information) to display the resulting capacity.

```
                              Resulting Capacity

The configuration change that you requested would result in the
following ASP capacities.

Press Enter to continue.

                ----------Current----------  ----------Modified----------
                --Protected--  -Unprotected-  --Protected--  -Unprotected-
ASP   Threshold   Size  %Used   Size  %Used   Size   %Used    Size   %Used
 1        90%        0  0.00%   1600 52.70%     0    0.00%   1200   70.26%
```

8. Press the Enter key to return to the Confirm Remove of Units display.
9. Press the Enter key on the Confirm Remove of Units display to remove the selected units. The system moves the data off the units selected to be

removed to the remaining units in the source ASP. The remove can take several minutes during which the system appears inactive.

10. When the remove operation is complete, you return to the Work with ASP Configuration display.

Task 4. Install the Licensed Internal Code

Before You Begin. . .

__ Clean the read and write head of the tape unit.

__ Find the most recent set of tapes used for your last complete save operation. The first volume contains the licensed internal code.

Note: Do not use the distribution tapes supplied by IBM unless you do not have a current SAVSYS tape. The SAVSYS tape contains the licensed internal code and the associated PTFs.

1. Ensure the key is in the keylock switch on the control panel.
2. Turn the key in the keylock switch until it points to the Manual position.
3. Press the Function Select switch to display `02` in the Function display on the control panel.
4. Press the Enter button on the control panel.
5. Select IPL type D (this specifies that the IPL source comes from tape) by pressing the Function Select switch on the control panel until **D** is shown on the Data display.
6. Press the Enter button on the control panel.
7. For the 9406 system unit, ensure that the power switches for the tape unit used for the IPL and all disk units are in the On position.
8. Find the Licensed Internal Code tape, which is the first volume of the most current set of SAVSYS tapes or the first volume of the distribution tapes.

   Warning!

   Use the distribution tapes only if no SAVSYS tape exists. If you use the distribution tapes, some system information will be lost. This includes, but is not limited to, PTFs and PTF packages. All cumulative PTF packages and individual PTFs applied after the initial installation of your system must be installed again.

9. Place the tape in the tape unit used for the IPL. For more information on loading the tape, see the setup manual for the device.

   Note: If your tape unit cannot be loaded when the power is off, continue with the next step. You will be prompted later by an SRC code for the tape.

10. Turn on the power to the system by pushing the Power switch on the control panel up. The switch returns to center after you push it. The 9402 system unit has a green button labeled Power On on the control panel.
11. If you could not load your tape in a previous step, load the first tape volume into the tape unit used for the IPL. Make the device ready and then continue with the next step.

Note: If your system was not powered down after ending the subsystems, do the following:

a. Press the Function Select switch to display `03` (continue the IPL) in the Function display on the control panel.

b. Press the Enter button on the control panel.

12. If the system attention light is on and one of the SRC codes shown in the following table is displayed in the Data display, complete the instructions for that SRC code. Otherwise, continue with the next step.

Table 7-1. SRC codes

Symptom	Action
A100 1933 **A12x 1933** ('x' is any character)	This SRC is shown if the tape device for the alternate IPL is not ready. Make sure the correct tape is loaded and make the tape device ready. Wait for the System Attention light to go off. Then, continue with the next step. If the System Attention light stays on for more than 5 minutes, check to see if you have the correct tape loaded in the tape device for the alternate IPL and make the tape device ready. Then continue with the next step.
B1xx 1803 **B1xx 1806** **B1xx 1938**	These SRCs are shown if the tape device for the alternate IPL was not found or was not ready. Make sure the tape device is powered on, the correct tape is loaded, and ready. Then continue with the next step.
B1xx 1934	This SRC is shown if the wrong tape is loaded. Load the correct tape and make the tape device ready. Then continue with the next step. This SRC is also shown if the high speed feature is enabled on the 2440 tape unit. The high speed feature must be disabled before installing or restoring the Licensed Internal Code.
2507 0001 **2642 0001** **2643 0001**	These SRCs are shown if a tape is not loaded in the tape device for the alternate IPL. Make sure the correct tape is loaded in the correct drive and then continue with the next step.

Note:

If any SRC listed in the table does not disappear from the control panel, do the following:

1. Press the Function Select switch to display `03` (continue the IPL) in the Function display on the control panel.
2. Press the Enter button on the control panel.

13. Ensure that the tape is online or ready. No action is required for tape units that perform this step automatically (such as the tape cartridge unit).
14. Ensure that the console display is turned on.
15. Wait for the yellow System Attention light on the control panel to light up.

    There is a delay while the system loads information from the tape. SRCs showing status are continuously updated on the control panel while processing occurs. This can take from 5 to 20 minutes; the time varies depending on the speed of the tape unit and the processor speed for the specific system model.

    When SRC A6xx 6001 is displayed, the system is prepared to start installing or restoring the Licensed Internal Code on the disk unit containing unit 1. Continue with the next step.

16. Select function code 24 by pressing the Function Select switch on the control panel until 24 is shown in the function display on the control panel.
17. Press the Enter button on the control panel.

> **Warning!**
>
> Function Code 24 (Install) is used only to install the licensed internal code from the SAVSTG media, recover from the loss of unit 1 in the system ASP, or to recovery from a disaster. Function code 24 (Install) deletes all information on the disk unit containing unit 1, including customer data. All system configuration information is also deleted. All disk units except unit 1 become nonconfigured units during the IPL. If you are restoring to the same system and have mirrored protection, checksum protection, or user ASPs configured, the service representative must use the Recover Configuration option in Dedicated Service Tools (DST) to recover the configuration.

The System Attention light may appear in one or two minutes and SRC A6xx 6002 is displayed. If you are sure you want to install the licensed internal code, select function code 24 again and press the Enter button. When SRC A6xx 6002 is displayed, option 23 can also be selected. If option 23 is selected, a restore of the licensed internal code is performed (not an install). Data on the disk units will not be lost.

Notes:

a. If another SRC is displayed after A6xx 6001 that is not in the A6xx xxxx format, then the system needs additional attention. Call your service representative.

b. If the following SRCs are displayed after SRC A6xx 6001 is displayed, see Appendix A, "Licensed Internal Code SRCs That Require User Input (A6xx xxxx)" for an explanation of these SRCs and the steps to follow.

A6xx 6002 Disk unit may contain a valid system
A6xx 6003 Disk unit not currently a load source
A6xx 6004 Disk unit not currently a load source
A6xx 6005 Disk unit not found

18. After pressing the Enter button on the control panel, the system starts displaying status SRCs again, which will be continuously updated to show the status of the install or restore operation. An example of a status SRC is D6xx 6201 (stand-alone install operation is running).
19. If the yellow system attention light is on again, and SRC A6xx 6048 (New tape volume needs to be loaded) is displayed, the system needs the next tape. The *xx* tells which volume needs to be loaded. Load the correct tape and make the device ready. The install or restore operation automatically continues.

    If SRC A6xx 6051 appears, the stand-alone function is requesting the Model-Unique Licensed Internal Code tape found inside the back cover of the system unit or on the side of the 9402 Model C04, D02, and E02 system units. Unload the current tape from the tape device and load the Model-Unique Licensed Internal Code tape.

    - A6xx 6051 Model-Unique Licensed Internal Code tape needs to be loaded.
    - A6xx 6052 Tape loaded was not the Model-Unique Licensed Internal Code.

If another SRC A6xx xxxx is displayed, look up the displayed SRC in Appendix A, "Licensed Internal Code SRCs That Require User Input (A6xx xxxx)" on page A-1 and follow the instructions. For all other SRCs call your service representative.

20. When the IPL is complete, the IPL or Install the System menu is shown. Continue with the next task.

Task 5. Start Restoring the Operating System

You use two displays to select the install options. The IPL or Install the System display allows you to restore the operating system or work with the service tools. The Install the Operating System display allows you to set the options to be used for restoring the system, and for the system date and time.

1. At the IPL or Install the System menu:

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

2. Type a 2 (Install the operating system).

   Note: **Do not** use option 4 (Perform automatic installation of the operating system) to restore the operating system.

3. Press the Enter key.

   The Confirm Install of the Operating System display is shown.

```
                  Confirm Install of Operating System

Press Enter to confirm your choice to install the operating system.
Press F12 to return and cancel your choice to install the
  operating system.
```

4. Press the Enter key.
5. The following display may be shown if your system is set up to prevent unauthorized installation of the operating system:

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

6. Enter the DST password and press the Enter key. For more information about preventing unauthorized installation of the operating system, see the *Basic Security Guide*.
7. The Select a Language Group display is shown. This display shows the primary language currently on the system or on the save tapes.

   The value specified on the display must be the same as the national language that is on the distribution media, or on your most recent SAVSYS tape.

```
                          Select a Language Group

Note:  The language feature shown is the language feature
installed on the system.

Type choice, press Enter.

  Language feature . . . . . . . . . . . . . . .  2924
```

8. Press the Enter key.

   After the language feature is entered, the Confirm Language Feature Selection display is shown. If you need to change your system's primary language, see the *Licensed Programs and New Release Installation Guide* for more information.

```
                    Confirm Language Feature Selection

 Language feature  . . . . . . . . . . . . . . . . . :   2924

 Press Enter to confirm your choice for language feature.
   Installing the system will continue.
 Press F12 to return to change your choice for
   language feature.
```

9. Press the Enter key to confirm the information.
10. Status messages are displayed.

The following is an example of a status display shown before the Install the Operating System display is shown. The status messages shown do not require any action by the user.

```
                          Running IPL Step

Current IPL step . . . . : Storage Management Recovery
```

After the IPL steps complete, the Install the Operating System menu appears.

```
                       Install the Operating System

Type options, press Enter.

     Install
       option . . . .  __              1=Take defaults (No other
                                         options are displayed)
                                       2=Change install options

     Date:
       Year . . . . .  __              00-99
       Month  . . . .  __              01-12
       Day  . . . . .  __              01-31

     Time:
       Hour . . . . .  __              00-23
       Minute . . . .  __              00-59
       Second . . . .  __              00-59
```

11. When the Install the Operating System display is shown, continue with the task to select install options.

Task 6. Select the Install Options

On the Install the Operating System display is shown, do one of the following.

1. Select option 1 (Take defaults) to restore objects such as the system values, system reply lists, and the edit descriptions. This option is selected when you are performing a total system restore operation. When you select 1 for *Default option*, the operating system is installed again and no more install options displays are shown. Do the following:

   a. Fill in the date and time.

      Date

      The system inserts the date based on the internal clock. If the date is incorrect, you can type over the date to change it.

Time

The system inserts the time based on the internal clock. If the time is incorrect, you can type over the time to change it.

b. Press the Enter key.

c. Messages are shown to indicate how many programs and language objects are restored. These messages are for your information only.

d. Continue loading tapes in sequence when messages are shown that ask you to load the next tape. The system searches through the tapes and loads the necessary programs and language information. After processing all the system save tapes, the following message may be displayed at the bottom of a blank display:

```
Operating system has been installed.  IPL in progress.
```

When the IPL is complete, the IPL Sign On display is shown and the system in ready to complete the IPL. Continue with the next task.

Task 7. Select IPL Options

1. Type `QSECOFR` in the *User* prompt and the password required for that user ID in the *Password* prompt (if password security is active) on the Sign On display.

   Note: If function code 23 was used to restore the Licensed Internal Code, before restoring the operating system, the password is the special one you assigned to QSECOFR user profile after the system was installed.

   If function code 24 was used to install the Licensed Internal Code, no password is required at this time. The system security level will be restored after the operating system is installed and the IPL completes.

2. Press the Enter key. Informational messages are displayed.

3. If the Select Product to Work with PTFs display appears, press F3 (Exit) to continue.

```
                Select Product to Work with PTFs
                                                              RCHASTTX
Position to . . . . . .____________ Product


Type options, press Enter.  Press F21 to select all.
   1=Select


              Product
Opt  Product  Option  Release
 _    5738999  *BASE   V2R3M0
 _    5738SS1  *BASE   V2R3M0
```

4. When the IPL Options display is shown, respond to the prompts using the following information.

```
                                IPL Options

Type choices, press Enter.

  System date  . . . . . .  . . .  07 / 26 / 88   MM / DD / YY
  System time  . . . . . .  . . .  12 : 00 : 00   HH : MM : SS
  Clear job queues . . . . . .  .  N              Y=Yes,  N=No
  Clear output queues  . . . . .   N              Y=Yes,  N=No
  Clear incomplete job logs  . . .  N             Y=Yes,  N=No
  Start print writers  . . . . . .  N             Y=Yes,  N=No
  Start this device only . . . . . .  Y           Y=Yes,  N=No


  Set major system options . . . . . .  Y         Y=Yes,  N=No
  Define or change system at IPL . . . . .  N     Y=Yes,  N=No
```

Figure 7-3. IPL Options Display

5. Enter the value for the system date.

   The date is displayed. The system date format shown can be YY/MM/DD, DD/MM/YY, or MM/DD/YY where MM means month, DD means day, and YY means year. For English, the default is MM/DD/YY; the default value differs according to the primary language

   If the date is not correct, you can type over the date to change it. The system date must have a year value in the range of 87 to 99, or 00 to 22.

6. Enter the value for system time.

   The current time is displayed. The time format is HH : MM : SS; HH means hour, MM means minutes, and SS means seconds. If you want to change the time, type it in accordance with the 24-hour clock. For example, for an IPL at 4:30 p.m., type 163000 for the time.

7. Enter the value for start print writers.

   If you are going to continue restoring user profiles, device configuration objects, user libraries, and authorities, type an N to not start the print writers. Otherwise, type a Y to start print writers.

8. Enter the value for start this device only.

   If you are going to continue restoring user profiles, device configuration objects, user libraries, and authorities, type a Y to start this device only. Otherwise, type an N to start all devices.

9. Enter the value for set major system options.

   The default is different, depending on the type of restore operation. If you restored the Licensed Internal Code using function code 23 (Restore), the default value is set to N. If you installed the Licensed Internal Code using function code 24 (Install), the default value is set to Y.

   Type a Y to set the major system options.

10. Enter the value for the define or change system at IPL.

    If the System/36 environment is your main operating system environment or you are restoring from the distribution tapes, then type a Y to define or change the system at IPL.

    If you are doing a partial recovery, you must ensure that all the libraries in the library list (QSYSLIBL and QUSRLIBL system values) are restored or remove

the libraries not being restored from the library list. Type a Y to define or change the system at IPL.

11. Press the Enter key.

    The Set Major System Options display is shown.

    Ensure enable automatic configuration is set to Yes, unless you are using the System/36 environment as your main operating environment. If you are using the System/36 environment as your main operating environment, ensure enable automatic configuration is set to No.

    If enable automatic configuration is set to No, you will receive SRC A900 2000 on the control panel later in the restore operation. The instructions to recover from SRC A900 2000 are provided, if necessary.

12. Press the Enter key.

    The Define or Change System at IPL is shown if you specified a *Y* for define or change system at IPL on the IPL options display.

    Was enable automatic configuration set to No?

    If it is set to No, do the following:

    a. Select option 3 (System value commands) and press the Enter key.

    b. Select option 3 (Work with system values) and press the Enter key.

    c. Type a 2 in the *Option* column next to the system value QIPLTYPE and press the Enter key.

    d. Change the value to *2* and press the Enter key. Press F12 until you return to the Define of Change the System at IPL menu.

    e. Press F3 (Exit) to continue the IPL.

    Are you restoring from the distribution tapes?

    If you are restoring the operating system from the distribution tapes, the system has reset some values back to the IBM-supplied defaults. These values must be changed back to the values that were in effect at the time of save operation. You should have lists of these values that were created at the time you performed a complete save operation.

    The following may need to be changed:

    - System values
    - Network attributes
    - Configuration lists
    - Edit descriptions
    - Reply list entries
    - IBM-supplied subsystem descriptions

      Note: Configuration lists, edit descriptions, reply list entries, and IBM-supplied subsystem descriptions can be changed after the operating system is restored.

    To change the system values, do the following:

    a. Type a Y to display the Define or Change the System menu.

    b. Select option 3 (System value commands) and press the Enter key.

    c. Select option 3 (Work with system values) and press the Enter key.

d. Type a 2 in the *Option* column next to the system value values you want to change and press the Enter key.

e. Change the values to the correct values and press the Enter key. Press F12 to return to the Define or Change the System at IPL menu.

If you had changed the network attributes from the IBM-supplied defaults, do the following:

a. Select option 4 (Network attributes commands) and press the Enter key.

b. Select option 2 (Change network attributes) and press the Enter key to display a list of network attributes.

c. Change the values to the correct network attributes and press the Enter key.

d. Press F12 (Cancel) to return to the Define or Change the System at IPL menu.

e. Press F3 to continue the IPL.

The following display is shown during the IPL process (attended mode) when there are system access paths marked for rebuild:

```
                    Edit Rebuild of Access Paths                RCHAS331
                                                      05/12/90  13:49:34

IPL threshold . . . . . . . . . . . . . 50   0-99

Type sequence, press Enter.
Sequence:  1-99, *OPN, *HLD

                    ------------Access Paths----------  Unique  Rebuild
Seq   Status        File        Library     Member      Keyed   Time
25__  IPL           QAPZSYM2    QSYS        QAPZSYM2    NO      00:00:01
25__  IPL           QAPZREQ2    QSYS        QAPZREQ2    NO      00:00:01
25__  IPL           QAPZPTF3    QSYS        QAPZPTF3    NO      00:00:01
25__  IPL           QAPZPTF2    QSYS        QAPZPTF2    NO      00:00:01
25__  IPL           QAPZOBJ2    QSYS        QAPZOBJ2    NO      00:00:01
*OPN  OPEN          QTWALL      QSYS        QTWALL      NO      00:00:06
*OPN  OPEN          QASULTEL    QSYS        QASULTEL    NO      00:00:01
*OPN  OPEN          QASULE05    QSYS        QASULE05    NO      00:00:01
*OPN  OPEN          QASULE03    QSYS        QASULE03    NO      00:00:01
*OPN  OPEN          QASULE01    QSYS        QASULE01    NO      00:00:01
                                                          More...
F5=Refresh   F11=Display member text   F13=Change multiple  F15=Sort by
F16=Repeat position to   F17=Position to
```

This display does not support the F3 and F12 keys.

- A status message is sent to notify the user that the system is performing access path recovery.
- The *IPL threshold* is a value from 1 through 99 that can be set by the user (default is 50), which indicates that access paths with a sequence less than or equal to the number specified will be rebuilt at IPL time. If the IPL threshold changes, all access paths with a status of IPL and AFTIPL will be changed to reflect the new status of the IPL threshold.
- Sequence
  - *IPL threshold*-1 represents the sequence of the access paths that are to be rebuilt prior to completion of the IPL. A rebuild sequence of 25 is

initially set by the system to set the sequence of access paths for the files that have MAINT(*IMMED) and RECOV(*IPL) specified. The access paths with the same sequence are built first according to rebuild time (the access paths that take the longest to rebuild are rebuilt first if the priorities are the same). The access paths are displayed in the same order.

  - *IPL threshold*-99 represents the sequence in which the access paths are rebuilt after the IPL. A rebuild sequence of 75 is initially set by the system to set the sequence of the access paths for the files that have MAINT(*IMMED) and RECOV(*AFTIPL) specified.
  - *OPN indicates the access path is to be rebuilt when the file is opened. The *OPN must be changed to 1 through 99 before the system job will initiate the rebuild. The system initially sets the sequence to *OPN for the files that have MAINT(*IMMED) and RECOV(*NO) specified.
  - *HLD indicates the access path is not to be rebuilt until the user changes the sequence from *HLD to a *OPN, or 1 through 99. *HLD will also cancel the rebuilding of any access path.

- Status
  - RUN indicates that the access path is being rebuilt.
  - IPL indicates that the access path is to be rebuilt before the system completes the IPL process.
  - AFTIPL indicates that the access path is to be rebuilt after the system completes the IPL process.
  - HELD indicates that the access path is not to be rebuilt until the user changes the sequence from *HLD to a *OPN, or 1 through 99.
  - OPEN indicates that the access path is to rebuilt when the file is opened.
- Rebuild Time
  - The time the access path will take to be rebuilt when the system is running without any other jobs on the system. For example, at IPL time. This is an estimate of rebuild time based on the file size and key length. No time for journaled access paths is displayed.

13. Do one of the following:
    - Make changes and press the Enter key. After changing the fields on the display and pressing the Enter key, the change to be made, if possible. For example, if the user attempts to change the sequence from 9 to 50, but the sequence cannot be changed because the access path has already been rebuilt, the user is sent an error message for each improper input.
    - Press the Enter key. If you press the Enter key without making any changes to the display, the Display Access Path Status display is shown (only if access paths remain to be rebuilt). If no access paths need to be rebuilt, the status display is not shown and the IPL continues.

    The following display is shown during the IPL process when the user finishes with the Edit Rebuild of Access Paths display.

```
                    Display Access Path Status

IPL Threshold . . . . . : 50

          ----------Access Paths----------   Rebuild     Current
Status    File        Library    Member      Build Time  Run Time
RUN       QAPZSYM2    QSYS       QAPZSYM2    00:00:01    00:00:01
JRN       QAPZREQ2    QSYS       QAPZREQ2    00:00:01
JRN       QAPZPTF3    QSYS       QAPZPTF3    00:00:01
JRN       QAPZPTF2    QSYS       QAPZPTF2    00:00:01
JRN       QAPZOBJ2    QSYS       QAPZOBJ2    00:00:07
JRN       QTWALL      QSYS       QTWALL      00:00:01
JRN       QASULTEL    QSYS       QASULTEL    00:00:01
SYS       QASULE05    QSYS       QASULE05    00:00:01
SYS       QASULE03    QSYS       QASULE03    00:00:01
IPL       QASULE01    QSYS       QASULE01    00:00:01
                                                        More...



F3=Exit and continue IPL   F12=Cancel
```

Every 5 seconds the display is updated with the current run time.

After all the access paths have been rebuilt (access paths with a sequence less than or equal to the *IPL threshold*), the IPL process continues and this display is removed.

F12 (Cancel) calls the Edit Rebuild of Access Paths display. If the user returns to the Edit Rebuild of Access Paths display using F12 (Cancel), the user must exit the Edit Rebuild of Access Paths again. Even if all the access paths are rebuilt, the user remains at the Edit Rebuild of Access Paths display until the user exits the display.

If F12 (Cancel) is pressed and there are only SYS/JRN access paths to be recovered, the edit display is shown without any access paths to be edited.

Status

- RUN–indicates that the access path is being rebuilt.
- SYS–indicates the access path is a system access path and is waiting to be rebuilt.
- JRN–indicates that the access path is being recovered from its associated journal.
- IPL–indicates that the access path will be rebuilt before the system completes the IPL and is waiting to be rebuilt.

14. Press F3 (Exit and continue the IPL) to continue.
15. Press the Enter key to continue.

Task 8. Recovery From SRC A900 2000, If Necessary

If function code 24 (Install) was used to restore the Licensed Internal Code and automatic configuration was turned off during the restore operation of the operating system (SRC A900 2000 is displayed on the console), you must create a tape description and possibly a controller description to finish the restore operation. **Do not** create a user-defined device description for the console display.

Do the following:

1. If your tape unit is a 3422, 3430, 3480, or a 3490, do the following:
   a. Use the Work with Hardware Resource (WRKHDWRSC) command to determine the location of the tape controller.

      ```
      WRKHDWRSC TYPE(*STG)
      ```

   b. Locate the resource name for the tape controller on the Work with Storage Resources display. 2604, 2622, or 2644 will be displayed in the *Type* column.
   c. Type a 9 (Work with) in the *OPT* column next to name and press the Enter key. The Work with Storage Controller Resources display is shown.
   d. Locate the resource for the tape controller (for example, TAPCTL01).
   e. Type a 7 (Create description) in the *Opt* column next to the name and press the Enter key. The Create Controller Description display is shown.
   f. Enter a tape controller name (such as TAPCTL01) in the *New device description* prompt and press the Enter key.
   g. On the Create CTL Desc (Tape) (CRTDEVCTL) display, enter a controller description name and controller type and model.
   h. Use the Create Device Description command to create a device description for each tape unit attached to the controller. For example:

      ```
      CRTDEVTAP DEVD(TAPxx) TYPE(34xx) MODEL(model-number)
           CTLD(tape-controller-description) TEXT('text description')
      ```

      where DEVD is the name of the description, TYPE is tape unit type, MODEL is the model (or *ANY), and CTLD is the name of the controller description created above.
   i. Use the Work with Configuration Status command to vary on the controller and tape unit.

      ```
      WRKCFGSTS *CTL *TAP
      ```

   j. Find the controller description and type a 1 in the *Opt* column next to the name. Press the Enter key. This will vary on the controller and any tape units attached to the controller.
2. If you are not using a 34xx tape unit, do the following:
   a. Use the Work with Hardware Resource (WRKHDWRSC) command to determine tape controller name.

      ```
      WRKHDWRSC TYPE(*STG)
      ```

   b. Locate the tape controller.
   c. Type a 9 (Work with) next to tape controller name and press the Enter key.
   d. Locate the resource name for the tape unit (for example, TAP01).
   e. Enter a 7 (Add configuration) in the *Opt* column next to the tape resource name and press the Enter key.
   f. Enter a tape device description name (for example, TAP01) in the *New device description* prompt and press the Enter key.
   g. Use the Work with Configuration Status command to vary on the tape unit.

      ```
      WRKCFGSTS *DEV *TAP
      ```

h. Find the tape device description and type a 1 in the *Opt* column next to the name. Press the Enter key. This will vary on the tape unit attached to the controller.

SRC A900 2000 remains displayed on the control panel throughout the remaining restore operations. When the final IPL of the system is complete, SRC A900 2000 disappears. The user-defined device description for the console display will be restored when the Restore Configuration (RSTCFG) command is run later in the recovery.

Task 9. Restore the Remaining Parts of the System

Before You Begin. . .

__ Clean the read and write head of the tape unit.

__ Find the tape volume that contains the user profiles.

__ Use the VRYCFG command to vary off the device configuration objects to be restored. The device configuration objects cannot be active when being restored.

__ If you do not know where the user profiles are stored on tape, use DSPTAP and specify DATA(*LABELS) to determine on which volume:

- File QFILEUPR is located. (This step is not necessary if you are restoring user profiles from the SAVSECDTA media.)
- File QFILEIOC is located. This file contains object types *CFGL, *CNNL, *COSD, *CTLD, *DEVD, *LIND, *MODD, *NWID, and *SRMSPC. This step is not necessary if you are restoring the configuration information from a SAVCFG tape.

Considerations: There are two options you can use to restore the user profiles, device configuration objects, user libraries, document library objects, and authority:

1. If you are restoring the user profiles from a SAVSYS tape and the following considerations do not apply, go to "Option 1. Using Option 21 on the Restore Menu" in this procedure.
2. If any of the following considerations apply, go to "Option 2. Using the Restore Commands" (not option 21 on the Restore menu) in this procedure.

Use the restore commands (not option 21 on the Restore menu) if:

- You prefer to enter the commands manually.
- You saved changed objects or have journal changes to apply.

Attention

To ensure the journaling environment is restored correctly, the libraries containing the journals must be restored before the libraries containing the journaled files. If the journaled files are restored before the journals, journaling is not started again for the files.

- You performed individual save operations instead of using the SAVLIB LIB(*NONSYS) command. You must use a RSTLIB command for each saved library. If you saved individual objects using the SAVOBJ or SAVCHGOBJ command, you must use a RSTOBJ command for each group of saved objects.

- You performed a SAVLIB LIB(*IBM) and a SAVLIB LIB(*ALLUSR). You need to do a RSTLIB SAVLIB(*IBM) and RSTLIB SAVLIB(*ALLUSR).
- You saved the security information with the Save Security Data (SAVSECDTA) command. You must restore the information using the restore commands.
- You saved logical file access paths using either the SAVOBJ or SAVCHGOBJ command. You must restore the logical files the same way you restored the physical files using the RSTOBJ command.

Method 1. Using Option 21 (The System) on the Restore Menu

To restore user profiles, configuration objects, system resource management information, user libraries, document library objects, and authority, use the following steps:

1. Sign on the system as the security officer; type QSECOFR in the user prompt and the password for QSECOFR in the *Password* prompt.

   Note: If you restored the Licensed Internal Code (function code 23), it is the user-assigned password. If you installed the Licensed Internal Code (function code 24), it is the default password QSECOFR.

2. Press the Enter key.
3. Ensure that the correct volume of your last set of save tapes is loaded and make the tape device ready. The tape should contain the file labeled QFILEUPR. Run the DSPTAP command and specify DATA(*LABELS) to find the file labeled QFILEUPR.
4. Ensure that any device configuration objects not used in the restore operation are varied off.
5. Ensure that the devices you are using for the restore operation (workstations, tape devices, and tape controllers) are varied on. These configuration objects are excluded from the restore operation (message CPF379C in the job log).
6. Go to the Restore menu:

   GO RESTORE

   The Restore menu is shown.

```
RESTORE                              Restore
                                                      System:   RCHASLLZ
Select one of the following:

  Restore Data
     1. Files
     2. Libraries
     3. Documents and folders
     4. Programs
     5. Other objects
     6. Licensed programs
     7. Configuration
     8. User profiles

  Restore System Data
    20. All libraries other than system library
    21. The system
```

Doing an Unattended Restore

To prevent an interrupted restore caused by incomplete restore messages, run the following commands before selecting option 21 from the Restore menu.

1. To display the reply list sequence numbers currently being used, type the following and press the Enter key.

   ```
   WRKRPYLE
   ```

2. To add message CPA3709 to the reply list, type the following (where xxxx is an unused sequence number 1-9999) and press the Enter key.

   ```
   ADDRPYLE SEQNBR(xxxx) MSGID(CPA3709) RPY('G')
   ```

3. To change the job, type the following and press the Enter key.

   ```
   CHGJOB INQMSGRPY(*SYSRPYL)
   ```

   Note: Communications messages with a severity of 99 that require a reply can stop an unattended restore operation. If you have communication messages that can stop an unattended restore operation, you can specify *NOTIFY for the *Message queue delivery* prompt on the Specify Command Defaults display. This sends the communication messages to the QSYSOPR message queue without interrupting the restore operation.

   After running these commands, the following messages will be displayed:

   a. `CPF0994  ENDSBS(*ALL) command being processed`

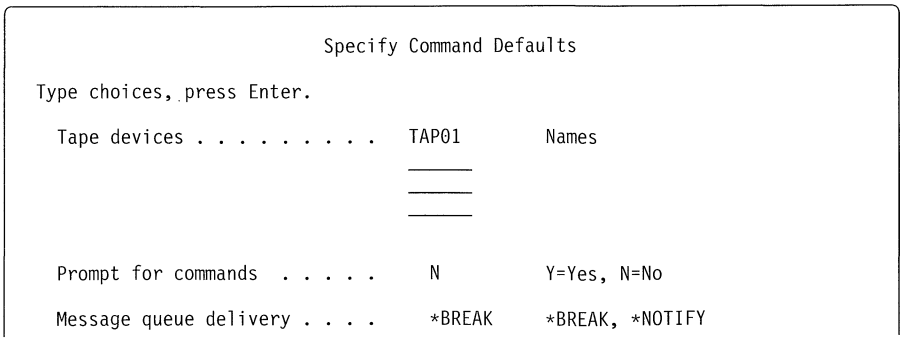
   b. Press the Enter key.

   c. `CPF0968  System ended to restricted condition`

   d. Press the Enter key.

After performing step d, the first message, `ENDSBS(*ALL) command being processed`, will return to the screen. Repeat steps b through d before moving on to select option 21.

7. Select option 21 (The system) on the Restore menu and press the Enter key. The following display is shown.



```
                          Specify Command Defaults

Type choices, press Enter.

  Tape devices . . . . . .  TAP01       Names
                            _____
                            _____
                            _____


  Prompt for commands  . . . . .   N          Y=Yes, N=No

  Message queue delivery . . . .   *BREAK     *BREAK, *NOTIFY
```

Tape devices

You can specify up to four tape device names. If you specify more than one device, the system automatically switches to the next tape device after the current tape is read.

Prompt for commands

You can specify whether or not you want to be prompted for the commands. If you specify Y=Yes, the prompt display is shown and you can change the defaults on the commands. If you specify N=No, the system runs the commands without prompting and uses the default values.

Message queue delivery

You can specify whether or not you want messages sent in *BREAK or *NOTIFY mode to the QSYSOPR message queue. If *BREAK is specified, any message of severity 99 that requires a reply interrupts the restore operation. If *NOTIFY is specified, severity 99 messages that are not are associated with restore operation, do not interrupt the restore process. For example, messages that request a new volume to be loaded interrupt the restore operation because they are associated with the job. You cannot continue until the you reply to these messages.

Note: If you are doing an unattended restore operation and communications is active, change the message queue delivery to *NOTIFY mode.

Option 21 will guide you through the following if you selected `Y` for the *Prompt for commands* prompt on the Specify Command Defaults display.

a. ENDSBS SBS(*ALL) OPTION(*IMMED)

b. RSTUSRPRF USRPRF(*ALL)

c. RSTCFG OBJ(*ALL)

d. RSTLIB SAVLIB(*NONSYS)

e. RSTDLO DLO(*ALL) SAVFLR(*ANY)

Note: If an unrecoverable error occurs when running the RSTDLO DLO(*ALL) SAVFLR(*ANY) command, refer to the topic "Media or Device Error When Running the RSTDLO Command" in the *Basic Backup and Recovery Guide*.

8. Press the Enter Key.
9. Continue loading the save tapes in sequence when the system sends a message to load the next volume.

If a media error occurs....

If an unrecoverable media error occurs during the RSTLIB procedure, you can restart the procedure using the STRLIB parameter on the RSTLIB command. The STRLIB parameter is valid only when *NONSYS, *ALLUSR, or *IBM is specified for the restore operation,

The basic recovery steps for a restore operation are:

1. Check the job log to determine the library where the previous RSTLIB SAVLIB(*NONSYS, *IBM, or *ALLUSR) command failed. Find the library that follows the last successfully restored library. It is indicated by a successful restore completion message in the job log. If the library is not identified in the job log, look at the output from the save operation or display the tape file QFILE to determine which library is next.
2. Load the first tape of the SAVLIB LIB(*NONSYS, *ALLUSR, or *IBM) media.
3. Type the following and press the Enter key:

   ```
   RSTLIB SAVLIB(*NONSYS, *IBM or *ALLUSR) DEV(tape-name)
       ENDOPT(*LEAVE) STRLIB(library-name) OMITLIB(library-name)
   ```

   where the *library-name* for the STRLIB and the OMITLIB parameters is the library where the RSTLIB failed. This starts the restore operation on the library after the library where the RSTLIB failed.
4. You will be asked to load the volume containing the starting library.
5. After the restore operation is complete, restore the library that failed using the media from a previous save operation.

   Note: Consider eliminating the tape with the media error from the next save rotation cycle to avoid a tape error again.

10. This completes the restore operation.
11. If you are unsure what the QSECOFR password is, change it now. To see if the password has expired, type the following:

    ```
    DSPUSRPRF QSECOFR
    ```

    If the password expiration is active for the QSECOFR user profile, you will see the expiration date on the Date password expired field. If the date is the current system date or prior, change the password for the QSECOFR user profile.
12. Check the job log to ensure all objects were restored.

    The job log contains information about the restore operation. To verify that all objects were restored, you should spool the job log for printing, along with the job's remaining spooled output, if any.

    ```
    DSPJOBLOG * *PRINT
    ```

    Or

    ```
    SIGNOFF *LIST
    ```

    Message CPC3703 is sent to the job log for each library that was successfully restored. Message CPF3773 is sent to tell you how many objects were restored. It also tells you how many objects were not restored. Objects are not restored for various reasons. Check for any error messages, correct the

errors, and then restore those objects from the media. After running these commands, the following messages will be displayed:

a. `CPF0994 ENDSBS(*ALL) command being processed`

b. Press the Enter key.

c. `CPF0968 System ended to restricted condition`

d. Press the Enter key.

After performing step d, the first message, `ENDSBS(*ALL) command being processed`, will return to the screen. Repeat steps b through d before moving on to select option 21.

Method 2. Using the Restore Commands

To use the commands to restore the system, do the following:

1. Sign on the system as the security officer; type `QSECOFR` in the User prompt and the password for QSECOFR in the Password prompt.

   Note: If you restored the Licensed Internal Code (function code 23), it is the user-assigned password. If you installed the Licensed Internal Code (function code 24), it is the default password QSECOFR.

2. Press the Enter key.

3. Type the following command on the command line and press the Enter key.

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

4. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Messages are sent indicating when the subsystems have ended and the system is in a restricted state.

5. Change the QSYSOPR message queue.

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(99)
   ```

   Note: Communications messages with a severity of 99 and that require a reply can stop an unattended restore operation. If you are using communications, you may need to identify the messages that may require a reply and then add them to the reply list or change the delivery of the QSYSOPR message queue to *NOTIFY with a severity of 99.

6. Ensure that the correct volume of your last set of save tapes is loaded and make the tape device ready. The tape should contain file labeled QFILEUPR. Run the DSPTAP command and specify DATA(*LABELS) to find the file labeled QFILEUPR.

   Note: Use the tapes from the most recent complete save operation (SAVSYS), or if the security data was saved since the last complete save operation, use the SAVSECDTA tapes.

   If the SAVSYS tape is used, type the following:

   ```
   RSTUSRPRF DEV(tape-device-name) USRPRF(*ALL) ENDOPT(*LEAVE)
   ```

   If the save security data tape (SAVSECDTA) is used, type the following:

   ```
   RSTUSRPRF DEV(tape-device-name) USRPRF(*ALL) ENDOPT(*UNLOAD)
   ```

   The time that this takes can vary significantly.

7. Ensure any device configuration objects not used in the restore operation are varied off.
8. Ensure that the devices you are using for the restore operation (workstations, tape devices, and tape controllers) are varied on. These configuration objects will be excluded from the restore operation (message CPF379C in the job log).
9. If you used the distribution tapes to restore the operating system, some information was not restored. You must create or change this information again. You should have lists of this information that were created at the time you performed your save operation.

   The following may need to be created or changed:

   - Configuration lists
   - Edit descriptions
   - Reply list entries
   - IBM-supplied subsystem descriptions

   a. For the configuration lists, do the following:

      Use the Work Configuration List (WRKCFGL CFGL(*ALL)) command to create the configuration lists to match the information in your list.

   b. For edit descriptions, do the following:

      Use the Work with Edit Descriptions (WRKEDTD EDTD(*ALL)) command to create edit descriptions to match the information in your list.

   c. For reply list entries, do the following:

      Use the Add Reply List Entry (ADDRPYLE) command to create reply list entries to match the information in your list.

   d. For IBM-supplied subsystem descriptions, do the following:

      Use the Work with Subsystem Descriptions (WRKSBSD SBSD(*ALL)) command to change the IBM-supplied subsystem descriptions to match the information in your list.

10. Restore the device configuration objects from your most recent SAVSYS tape or SAVCFG tape:

    If the SAVSYS media is used, type the following:

    ```
    RSTCFG OBJ(*ALL) DEV(tape-device-name) OBJTYPE(*ALL) ENDOPT(*LEAVE)
    ```

    If the SAVCFG media is used, type the following:

    ```
    RSTCFG OBJ(*ALL) DEV(tape-device-name) OBJTYPE(*ALL) ENDOPT(*UNLOAD)
    ```

    The time that this takes can vary significantly.

11. Restore the IBM and user libraries in one of the following ways:

    If you used SAVLIB LIB(*NONSYS) to save the IBM-supplied and user libraries, load the correct volume and then type the following:

    ```
    RSTLIB SAVLIB(*NONSYS) DEV(tape-device-name) ENDOPT(*LEAVE)
    ```

    Or, if you used SAVLIB LIB(*IBM) and SAVLIB LIB(*ALLUSR) to save the IBM and user libraries, load the correct tape and then type the following two commands. The first command must complete before entering the second command.

    ```
    RSTLIB SAVLIB(*IBM) DEV(tape-device-name) ENDOPT(*LEAVE)
          MBROPT(*ALL)
    ```

```
RSTLIB SAVLIB(*ALLUSR) DEV(tape-device-name) ENDOPT(*LEAVE)
      MBROPT(*ALL)
```

Note: If you saved individual libraries and objects with the SAVLIB, SAVOBJ, and SAVCHGOBJ commands, then you will have to restore the individual libraries and objects with the RSTLIB command (not RSTLIB SAVLIB(*NONSYS)) and the RSTOBJ command.

Attention

To ensure the journaling environment is restored correctly, the libraries containing the journals must be restored before the libraries containing the journaled files. If the journaled files are restored before the journals, journaling is not started again for the files.

If a media error occurs....

If an unrecoverable media error occurs during the RSTLIB procedure, you can restart the procedure using the STRLIB parameter on the RSTLIB command. The STRLIB parameter is valid only when *NONSYS, *ALLUSR, or *IBM is specified for the restore operation,

The basic recovery steps for a restore operation are:

1. Check the job log to determine the library where the previous RSTLIB SAVLIB(*NONSYS, *IBM, or *ALLUSR) command failed. Find the library that follows the last successfully restored library. It is indicated by a successful restore completion message in the job log. If the library is not identified in the job log, look at the output from the save operation or display the tape file QFILE to determine which library is next.
2. Load the first tape of the SAVLIB LIB(*NONSYS, *ALLUSR, or *IBM) media.
3. Type the following and press the Enter key:

   ```
   RSTLIB SAVLIB(*NONSYS, *IBM or *ALLUSR) DEV(tape-name)
      ENDOPT(*LEAVE) STRLIB(library-name) OMITLIB(library-name)
   ```

   where the *library-name* for the STRLIB and the OMITLIB parameters is where the RSTLIB failed. This starts the restore operation on the next library after the library where the RSTLIB failed.
4. You will be asked to load the volume containing the starting library.
5. After the restore operation is complete, restore the library that failed using the media from a previous save operation.

   Note: Consider eliminating the tape with the media error from the next save rotation cycle to avoid a tape error again.

12. If you have documents, folders, and mail to restore, load the correct tape and type the following:

    ```
    RSTDLO DLO(*ALL) SAVFLR(*ANY) DEV(tape-device-name) ENDOPT(*UNLOAD)
    ```

Notes:

a. If an unrecoverable error occurs when running the RSTDLO DLO(*ALL) SAVFLR(*ANY) command, see the topic "Media or Device Error When Running the RSTDLO Command" in the *Basic Backup and Recovery Guide*.

b. If you are not using journaling, or do not have changed objects to restore, continue with the next step. Otherwise, continue with the task "Restore Changed Objects."

13. To restore the authority, type the following:

    ```
    RSTAUT
    ```

    Note: If users have private authority to many objects, the RSTAUT command can take a very long time to run.

    This completes the restore operation.

14. If you are unsure what the QSECOFR password is, change it now. To see if the password has expired, type the following:

    ```
    DSPUSRPRF QSECOFR
    ```

    If the password expiration is active for the QSECOFR user profile, you will see the expiration date on the Date password expired field. If the date is the current system date or prior, change the password for the QSECOFR user profile.

15. Check the job log to ensure all objects were restored.

    The job log contains information about the restore operation. To verify that all objects were restored, you should spool the job log for printing, along with the job's remaining spooled output, if any.

    ```
    DSPJOBLOG * *PRINT
    ```

    Or

    ```
    SIGNOFF *LIST
    ```

    Message CPC3703 is sent to the job log for each library that was successfully restored. Message CPF3773 is sent to tell you how many objects were restored. It also tells you how many objects were not restored. Objects are not restored for various reasons. Check for any error messages, correct the errors, and then restore those objects from the media.

Task 10. Restore Changed Objects

Attention!

If you are using journaling and need to apply journaled changes, continue with the following steps. Use these recommended steps to avoid a failed restore operation caused by restoring journal receivers with names that conflict with the journal receivers currently attached to the restored journals. Otherwise, ignore these steps and continue with "Restoring Changed Objects."

Note: If you are using OfficeVision/400 or PC Support/400 and are performing daily save operations using SAVDLO and SAVCHGOBJ LIB(QUSRSYS) OBJJRN(*NO) commands, you must perform the steps in "Working with Journals" for the system supplied journal QUSRSYS/QAOSDIAJRN. If you

specified OBJJRN(*YES) on the SAVCHGOBJ command, you do not need to apply journal changes.

Working with Journals

1. Type the following and press the Enter key:

   ```
   WRKJRN
   ```

2. The Specify Journal Name display is shown. Specify *ALL for the *Library name* prompt and press the Enter key.

3. The Work with Journals display is shown. To display the name of the currently attached journal receiver, type a 5 (Display journal status) in the *Opt* field for each journal on which you want to apply changes. Write down all the names of the journals and their currently attached journal receivers.

   Notes:

   a. You only need to perform the following steps for those journals you plan to use for recovering journaled files by performing the APYJRNCHG command. If no database files have been journaled to a journal, the the system cannot apply any journaled changes using the journal.

   b. If you are using OfficeVision/400 or PC Support/400, you must apply journaled changes to the files journaled to the system-supplied journal QUSRSYS/QAOSDIAJRN.

4. You cannot restore journal receivers from the SAVLIB, SAVOBJ, or SAVCHGOBJ media if they have the same names as the journal receivers that are attached. To later apply all journaled changes that have occurred since the last complete save operation, you must restore the receivers to the system from the save media.

   To avoid a failed restore operation of saved journal receivers because of name conflicts, do the following for each journal identified in the previous step.

   a. Create a journal receiver that will be used as a temporary receiver. Give it a name that will identify it as a temporary receiver, for example, TEMPnn. You can enter a description in the text (TEXT parameter) that identifies it as a temporary receiver for disaster recovery.

      ```
      CRTJRNRCV JRNRCV(library-name/TEMPnn)
            TEXT('temporary journal receiver for journal xxx')
      ```

      Repeat this step for each journal found in step 3.

   b. To detach the current receiver and attach the new TEMPnn receiver, type the following and press the Enter key.

      ```
      CHGJRN JRN(library-name/journal-name) JRNRCV(library-name/TEMPnn)
      ```

      Repeat this step for each journal found in step 3.

   c. Delete the detached journal receiver (identified in step 3 where you wrote down the name of the journal and journal receiver) using the Delete Journal Receiver (DLTJRNRCV) command.

      ```
      DLTJRNRCV JRNRCV(library-name/journal-receiver)
      ```

      Repeat this step for each journal found in step 3.

      If you receive message CPA7025 *Receiver never fully saved*, enter an I to ignore and press Enter to continue the delete.

This allows the journal receivers on the save media to be restored successfully.

Restoring Changed Objects

1. Load the SAVCHGOBJ tape.

   Note: Several commands found in library QUSRTOOL can help you during save and restore operations. If you create the RSTALLCHG command in library QUSRTOOL, you can use the RSTALLCHG command to restore the libraries with changed objects from the SAVCHGOBJ or SAVALLCHG media without the need to know the names of the saved libraries. You need to run only one RSTALLCHG command to restore all the libraries. For more information about these commands, see Appendix D of the *Basic Backup and Recovery Guide*.

2. If you do not use the RSTALLCHG in library QUSRTOOL and you specified SAVCHGOBJ LIB(*ALLUSR), type the following to determine the libraries that were saved:

   ```
   DSPTAP DEV(device-name) OUTPUT(*PRINT)
   ```

3. To restore changed objects, type the following and press the Enter key:

   ```
   RSTOBJ OBJ(*ALL) DEV(tape-device) SAVLIB(library-name)
             OBJTYPE(*ALL) ENDOPT(*LEAVE) MBROPT(*ALL)
   ```

   You must repeat this step for every library saved using the SAVCHGOBJ command.

4. Do one of the following:

   - If you are using journaling, perform the steps in "Applying Journal Changes" for each journal.
   - If you need to restore changed documents and folders, with the steps in "Restoring Changed Documents and Folders."
   - If you do not have no other restore steps to perform, continue with the following step.

5. Restore users' authority by entering:

   ```
   RSTAUT
   ```

   The time it takes for the RSTAUT command to complete can vary significantly. The time depends on the number of user profiles and private authorities that were saved during the save operation.

6. This completes the restore operation.

Task 11. Apply Journaled Changes

Ensure that all the journal receivers required for the apply journaled changes operation are available on the system. In general, you will need all journal receivers that were attached to the journal for the length of time for which journaled changes are now to be applied to the restored files. Restore all necessary journal receivers, including ones that might have not been restored earlier because of name conflicts with the receivers attached to the restored journals. Use the Display Journal Receiver Attributes (DSPJRNRCVA) command to determine when a journal receiver was attached to and detached from a journal.

1. Determine the name of the last journal receiver (the last receiver restored) by entering the following:

```
WRKJRNA JRN(library-name/journal-name)
```

2. Press the Enter key.
3. Press F15 (Work with receiver directory) from the Work with Journal Attributes display to show the last journal receiver with a status of SAVED or PARTIAL. Write down the name of the receiver.
4. Determine the chain of receivers to be used in the APYJRNCHG command from the Work with Receiver Directory display. Write down the first and last receiver that you restored (last receiver is prior to the TEMPnn receiver). Notice that the first and last receiver are the same if only one journal receiver was restored.

   Note: While looking at the receiver directory, you should also look for any receiver chain breaks. You can determine a chain break by looking at the first two digits in the *Number* column on the Work with Receiver Directory display. You cannot apply journaled changes across receiver chain breaks. Therefore, you must write down the beginning and ending receiver names for each receiver chain. Then you need to run a series of apply journaled changes operations, one for each chain using these receivers. The *Advanced Backup and Recovery Guide* has more information about receiver chain breaks.

5. When applying journal changes, if the ending receiver has a status of PARTIAL (saved-while-attached), the TOENT parameter requires a sequence number to be specified on the APYJRNCHG command. Determine the last entry to be applied for the last receiver (identified in previous step).

   To determine the last receiver in the receiver range, type an 8 (Display attributes) in the *Opt* field next to the receiver name on the Work with Receiver Directory display.

   Write down the value for the *Last Sequence Number* field.

6. To ensure that the files are currently being journaled, do the following from the Work with Journal Attributes display:

   a. Press F13 (Display journaled files) from the Work with Journal Attributes display to show the list of files currently being journaled. To start journaling for a physical file that should be in the list, run the STRJRNPF command for each physical file not in the list.

   b. Press F14 (Display Journaled Access Paths) from the Work with Journal Attributes main display to display the list of currently journaled access paths. To start journaling access paths for a physical or logical file that should be in the list, run the STRJRNAP command for the physical or logical file that is not in the list.

   Notice that before journaling an access path, all physical files over which the access path is built must first be journaled to this journal. When you have ensured all files are correctly journaled, continue with the next step.

7. To continue the naming convention for your journal receivers, create a receiver that follows the same naming convention as the last receiver but assign it a number of one greater.

   ```
   CRTJRNRCV JRNRCV(library-name/journal-receiver-nameNN)
   ```

By doing this, you are doing what the CHGJRN command would normally do if the last receiver saved was the current receiver being detached with a new receiver name being created.

8. Use the CHGJRN command to detach the temporary receiver and attach the new receiver you just created.

```
CHGJRN JRN(library-name/journal-name)
       JRNRCV(library-name/journal-receiver-nameNN)
```

9. Enter the following command to apply the journaled changes using the first and last journal receivers identified on the Work with Receiver Directory display.

```
APYJRNCHG JRN(library-name/journal-name)
          FILE((library-name/*ALL))
          RCVRNG(lib-name/first-receiver lib-name/last-receiver)
          FROMENT(*LASTSAVE) TOENT(last-entry)
```

Note: If you determined in step 4 that this journal had receiver chain breaks, then you must run an APYJRNCHG command for each chain instead of one command as shown. For the RCVRNG parameter, specify the first and last receiver for each chain. For the FROMENT and TOENT parameters, specify:

a. FROMENT(*LASTSAVE) and TOENT(*LAST) for the first receiver chain.

b. FROMENT(*FIRST) and TOENT(*LAST) for the middle receiver chains.

c. FROMENT(*FIRST) and TOENT(last-entry) for the last receiver chain.

Attention

You must specify individual files on the FILE parameter instead of *ALL for the QAOSDIAJRN journal in library QUSRSYS. Do not apply journal changes to the document and folder search index database files (QAOSSS10 through QAOSSS15, QAOSSS17, and QAOSSS18) for journal QAOSDIAJRN in library QUSRSYS.

```
APYJRNCHG JRN(QUSRSYS/QAOSDIAJRN)
     FILE((QUSRSYS/QAOKPLCA) (QUSRSYS/QAOSAY05)
          (QUSRSYS/QAOKPX4A) (QUSRSYS/QAOSAY07)
          (QUSRSYS/QAOKP01A) (QUSRSYS/QAOKP02A)
          (QUSRSYS/QAOKP03A) (QUSRSYS/QAOKP04A)
          (QUSRSYS/QAOKP05A) (QUSRSYS/QAOKP06A)
          (QUSRSYS/QAOKP08A) (QUSRSYS/QAOKP09A))
      RCVRNG(lib-name/first-receiver lib-name/last-receiver)
      FROMENT(*LASTSAVE) TOENT(last-entry)
```

If you need to restore changed documents and folders, with the steps in "Restoring Changed Documents and Folders." Otherwise, continue with the next step.

10. Restore users' authority by entering:

```
RSTAUT
```

The time it takes for the RSTAUT command to complete can vary significantly. The time depends on the number of user profiles and private authorities that were saved during the save operation.

11. This completes the restore operation.
12. Perform a normal IPL and return the system to normal operations:
    a. Turn the keylock switch to the Normal position.
    b. Type the following on a command line and press the Enter key.

       ```
       PWRDWNSYS OPTION(*IMMED) RESTART(*YES))
       ```
13. When the IPL is complete, sign on the system.
14. Start any other subsystems that need to be started, such as QTCP or QSNADS.

    ```
    STRSBS SBSD(subsystem-name)
    ```

Task 13. Restore Changed Documents and Folders

If you performed daily save operations for documents and folders, do the following steps. Otherwise, continue with the RSTAUT command.

1. Load the last daily SAVDLO tape.
2. If you performed daily save (SAVDLO DLO(*CHG)) operations to back up all new folders, new and changed documents, and mail since the last complete SAVDLO DLO(*ALL) FLR(*ANY) operation, type the following and press the Enter key.

   ```
   RSTDLO DLO(*ALL) DEV(TAP01) SAVFLR(*ANY) ALWOBJDIF(*ALL)
   ```

   Note: If an unrecoverable error occurs when running the RSTDLO DLO(*ALL) SAVFLR(*ANY) command, see the topic "Media or Device Error When Running the RSTDLO Command" in the *Basic Backup and Recovery Guide*.
3. Restore users' authority by entering:

   ```
   RSTAUT
   ```

   The time it takes for the RSTAUT command to complete can vary significantly. The time depends on the number of user profiles and private authorities that were saved during the save operation.
4. This completes the restore operation.
5. Perform a normal IPL and return the system to normal operations:
   a. Turn the keylock switch to the Normal position.
   b. Type the following on a command line and press the Enter key.

      ```
      PWRDWNSYS OPTION(*IMMED) RESTART(*YES))
      ```
6. When the IPL is complete, sign on the system.
7. Start any other subsystems that need to be started, such as QTCP or QSNADS.

   ```
   STRSBS SBSD(subsystem-name)
   ```

Recovering Devices That Will Not Vary On

If you have a problem with your devices, such as not being able to vary on a device, it may be because the system resource management (SRM) database that was restored does not match the device descriptions on the system.

Tape Controller - Tape Unit Types 3422, 3430, 3480, and 3490: To correct the problem for a tape controller, do the following:

1. Type the following and press the Enter key to display the Work with Storage Resources display.

   ```
   WRKHDWRSC TYPE(*STG)
   ```

2. Find the correct storage controller for the device that would not vary on.
3. Type a 9 (Work with resource) in the *Opt* column next to the resource name. The Work with Storage Controller Resources display is shown.
4. Find the valid resource name for the device type and model you tried to vary on.
5. Press F12 (Cancel) until you return to a display with a command line.
6. Type the following and press the Enter key to display the device description for the device that would not vary on.

   ```
   WRKCTLD CTL(controller-name)
   ```

   The Work with Device Descriptions display is shown.
7. Type a 2 (Change) in the *Opt* column next to the device description you want to change and press the Enter key. The Change Device Description display is shown.
8. Change the name in the *Resource name* prompt to the correct name for the resource and press the Enter key. You will return to the Work with Device Descriptions display.
9. Type an 8 (Work with status) in the *Opt* column next to the device description you changed and press the Enter key. The Work with Configuration Status display is shown.
10. Type a 1 (Vary on) in the *Opt* column next to the device description name and press the Enter key to vary on the device.

Tape Units Other Than Types 3422, 3430, 3480, and 3490: To correct the problem for a tape unit, do the following:

1. Type the following and press the Enter key to display the Work with Storage Resources display.

   ```
   WRKHDWRSC TYPE(*STG)
   ```

2. Find the correct storage controller for the device that would not vary on.
3. Type a 9 (Work with resource) in the *Opt* column next to the resource name. The Work with Storage Controller Resources display is shown.
4. Find the valid resource name for the device type and model you tried to vary on.
5. Press F12 (Cancel) until you return to a display with a command line.
6. Type the following and press the Enter key to display the device description for the device that would not vary on.

```
WRKDEVD DEV(device-name)
```

The Work with Device Descriptions display is shown.

7. Type a 2 (Change) in the *Opt* column next to the device description you want to change and press the Enter key. The Change Device Description display is shown.
8. Change the name in the *Resource name* prompt to the correct name for the resource and press the Enter key. You will return to the Work with Device Descriptions display.
9. Type an 8 (Work with status) in the *Opt* column next to the device description you changed and press the Enter key. The Work with Configuration Status display is shown.
10. Type a 1 (Vary on) in the *Opt* column next to the device description name and press the Enter key to vary on the device.

Local Work Station Controller: To correct the problem for a work station, do the following:

1. Type the following and press the Enter key to display the Work with Local Workstation Resources display.

   ```
   WRKHDWRSC TYPE(*LWS)
   ```

2. Find the correct controller description for the device that would not vary on.
3. Type a 5 (Work with configuration description) in the *Opt* column next to the controller description name and press the Enter key. The Work with Configuration Description display is shown.
4. Type a 5 (Display) in the *Opt* column to display the valid resource name for the work station controller.
5. Press F12 (Cancel) until you return to a display with a command line.
6. Type the following and press the Enter key to display the device description for the device that would not vary on.

   ```
   WRKCTLD CTLD(controller-name)
   ```

   The Work with Controller Descriptions display is shown.
7. Type a 2 (Change) in the *Opt* column next to the controller description you want to change and press the Enter key. The Change Controller Description display is shown.
8. Change the name in the *Resource name* prompt to the correct name for the resource and press the Enter key. You will return to the Work with Controller Descriptions display.
9. Type an 8 (Work with status) in the *Opt* column next to the controller description you changed and press the Enter key. The Work with Configuration Status display is shown.
10. Type a 1 (Vary on) in the *Opt* column next to the controller description name and press the Enter key to vary on the device.

    Note: It is possible that another device description is varied on for this resource. Vary off the device first and then vary on the changed device description. This situation can happen to the console device.

Recovering the System/36 Environment Configuration

If you are experiencing a problem with the System/36 environment after restoring the system, it may be caused by the locking rules used during the installation process. The QS36ENV configuration object in library #LIBRARY may have been locked by the System/36 environment.

This object contains the System/36 environment names for the work station, printer, tape and diskette units on the system and default System/36 environment values used for all users. This object may have been modified by the Change S/36 Environment Configuration (CHGS36) command to customize the System/36 environment.

When the first subsystem is started on the system after the installation process is complete, a new #LIBRARY and a new QS36ENV object in #LIBRARY is created with the AS/400 system defaults. In addition to the creating the new objects, each subsystem holds a lock on the QS36ENV configuration object to ensure that it is not deleted. This lock will not allow the saved QS36ENV configuration object to be restored.

If the QS36ENV configuration object did not restore, start with step 1. If the configuration object did restore but you are experiencing problems with the System/36 environment configuration, go to step 5.

1. Rename the newly created #LIBRARY to something else (for example, #LIBNEW).
   - The locks held on QS36ENV object remain with the renamed library. This allows the saved System/36 environment configuration object to be restored.
2. Restore the saved copy of library #LIBRARY. This library was saved using SAVLIB LIB(*NONSYS) or SAVLIB LIB(*ALLUSR).
3. Perform and IPL of the system.
   - The QS36ENV object in the restored copy of #LIBRARY is the System/36 environment configuration again.
4. Delete the earlier renamed version of #LIBRARY (for example, #LIBNEW).
5. Use the Change S/36 Environment Configuration (CHGS36) command to refresh the configuration object.
   a. Select each of the device types you want to change.
      - Work station devices
      - Printer devices
      - Tape devices
      - Diskette devices

   b. Do the following for each device type you want to change:
      1) Press the F5 key to ensure the configuration object matches the device descriptions on the system.
      2) Do one of the following if any System/36 names are not specified:
         - Press the F10 key to use the AS/400 defaults for the System/36 names for those devices.
         - Update the System/36 names manually.

c. Save the changes to the configuration object.

See the topic on configuring the System/36 environment in the *Concepts and Programmer's Guide for the System/36 Environment* for more information about configuring the System/36 environment.

Considerations for Recovering an Overflowed User ASP

If a user ASP exceeds its storage capacity, data for objects in the user ASP overflows into the system ASP. Consequently, the recovery benefits of separating auxiliary storage into the system and user ASPs are no longer guaranteed.

You can monitor user ASP storage use to avoid overflowing storage capacity by setting the storage threshold value for the user ASP. When the ASP storage usage goes past the threshold, the system operator receives messages warning of an impending user ASP overflow. If the user ASP is overflowed, it is necessary to recover the overflowed condition, use the DSPOBJD command to determine which objects in the ASP have overflowed. Recovery the overflow status using one of the following methods:

1. Delete all overflowed objects that are no longer needed.
2. Request the system recover the overflowed ASP during the IPL if you have freed enough space by deleting objects no longer needed. The objects being deleted in the ASP must not be in overflowed status.
3. Delete all objects in the user ASP, perform an IPL, and recover the overflow status.

If you are using journaling and have journals or journal receivers in the overflowed ASP, see "Moving Journal Receivers From an Overflowed User ASP to a Different ASP" on page 7-97 or "Moving a Journal From an Overflowed User ASP to a Different ASP" on page 7-98 for the procedures to save and restore the journals and journal receivers and start journaling again.

Avoid restoring all saved objects to the same ASP or data may overflow again. Consider restoring some of the objects to another user ASP by using the RSTASP parameter on the restore commands. You may also want to consider adding an additional unit to the user ASP that has overflowed.

Determining the Amount of Overflowed Storage

1. Type the following to display the System Service Tools (SST) menu.

   ```
   STRSST
   ```

   Press the Enter key. The following display is shown.

```
                        System Service Tools (SST)

Select one of the following:

     1. Start a service tool
     2. Work with active service tools
     3. Work with disk units
     4. Work with diskette data recovery



Select one of the following:
```

2. Select option 3 (Work with Disk Units) and press the Enter key. The following display is shown.

```
                              Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

3. Select option 1 (Display disk configuration) on the Work with Disk Units display and press the Enter key. The following display is shown.

```
                           Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units
     5. Display device parity status
```

4. Select option 2 (Display disk configuration capacity) and press the Enter key.

```
                    Display Disk Configuration Capacity

                                                      --Protected--  --Unprotected--
ASP  Unit  Type  Model  Threshold  Overflow   Size  %Used    Size  %Used
 1                          90%    No            0   0.00%   1400   8.22%
       1   9332   400                           0   0.00%    200  17.97%
       2   9332   400                           0   0.00%    200   6.60%
       3   9332   400                           0   0.00%    200   6.59%
       4   9332   400                           0   0.00%    200   6.62%
       5   9332   400                           0   0.00%    200   6.60%
       6   9332   400                           0   0.00%    200   6.57%
       7   9332   400                           0   0.00%    200   6.59%
 2                                Yes           0   0.00%    200  99.99%
       8   9332   200       90%                 0   0.00%    200  99.99%
 3                                Yes           0   0.00%    200  99.99%
       9   9332   200       90%                 0   0.00%    200  99.99%
 4                                No            0   0.00%    200   6.61%
      10   9332   200       90%                 0   0.00%    200   6.61%

Press Enter to continue.

F3=Exit F5=Refresh  F9=Overflow information  F11=Non-configured units
F12=Cancel
```

Figure 7-4. Display Disk Configuration Capacity Display

5. Press F9 (Display ASP Overflow information) to display the overflow amount and the amount of storage needed to recover the overflowed objects.

```
                    Display ASP Overflow Information

                  Overflow      ----Amount Needed to Recover----
ASP  Threshold     Amount         To Capacity     To Threshold
 2       90%          14                0               0
 3       90%          25               25              45
```

Figure 7-5. Display ASP Overflow Information

6. The following information is used to determine the overflow amount and the amount of storage needed to recover the overflowed objects.
   - *ASP:* The auxiliary storage pool number.
   - *Threshold:* The current threshold setting.
   - *Overflow amount:* The amount of storage that has overflowed from the user ASP into the system in millions of bytes. An asterisk (*) indicates that the amount is not known. The possible cause is that the overflow has occurred before Version 2 Release 2.
   - *Amount Needed to Recover:* The amount of storage needed in the ASP the recover from the overflow condition.
   - *To Capacity:* The minimum amount of storage that must be made available to get all the overflowed data back into the user ASP.
   - *To Threshold:* The minimum amount of storage that needs to be made available before the recover overflowed ASP operation can move the over-

flowed data from the system ASP back to the user ASP and remain within the storage threshold of the ASP.

Recovering an Overflowed User ASP by Deleting Overflowed Objects

You can reset an overflowed user ASP by determining which objects have overflowed and then deleting them. Use this procedure only if the value shown in the *Overflow Amount* field on the Display ASP Overflow Information display is something other than an asterisk (*).

Task 1. Determine the Overflowed Objects

Use the Display Object Description (DSPOBJD) command to determine if objects are overflowed. To display the objects in a user ASP, see "Displaying Objects in a User ASP" on page 7-94 or see the appendix on tips and techniques in the *Basic Backup and Recovery Guide* for information about the Display Overflowed Objects command in library QUSRTOOL.

Task 2. Save and Delete the Objects in the User ASP

Do one or more of the following:

1. If detached journal receivers are in the ASP, save them and then delete them.

   To save the journal receivers, do the following:

   a. Load a save tape

   b. Save the journal receivers contained in the user ASP:

   ```
   SAVOBJ OBJ(object-name) LIB(library-name) DEV(tape-device-name)
       OBJTYPE(*JRNRCV) VOL(*MOUNTED) ENDOPT(*UNLOAD)
   ```

   c. To delete the journal receivers, do the following:

   ```
   WRKLIB LIB(library-name)
   ```

   d. Type a 12 (Work with objects) in the *Opt* column and press the Enter key.

   e. Find the objects to be deleted in the *Object* column.

   f. Type a 4 (Delete) in the *Opt* column for each object you want to delete.

   g. Press the Enter key.

2. If some of the objects in the user ASP can be deleted because they are no longer needed, delete them.

   To delete the objects, do the following:

   a. Display the library in the user ASP or the library associated with the objects in the user ASP:

   ```
   WRKLIB LIB(library-name)
   ```

   b. Type a 12 (Work with objects) in the *Opt* column and press the Enter key.

   c. Find the objects to be deleted in the *Object* column.

   d. Type a 4 (Delete) in the *Opt* column for each object you want to delete.

   e. Press the Enter key.

3. Save all overflowed objects and then delete them:

To save the objects in the ASP that are needed on the system, do the following:

a. Save the objects needed on the system. If the objects you are saving include one or more active journals, consider saving the associated database files. Saving the database files allows you to start journaling again by deleting the old files and restoring the saved files.

   To save the objects to a save file:

   ```
   SAVOBJ OBJ(object-name) DEV(*SAVF) SAVF(save-file-name)
          OBJTYPE(*ALL)
   ```

   Or, save the objects to tape:

   ```
   SAVOBJ OBJ(object-name) LIB(library-name)
         DEV(TAP01) OBJTYPE(*ALL) ENDOPT(*UNLOAD)
   ```

   Journaling Consideration

   If the objects you want to save and delete include one or more journals, do the following:

   1. Determine all physical files currently being journaled and all files with the access paths being journaled using the WRKJRNA command.
   2. Save all the files that have their access paths being journaled by specifying ACCPTH(*YES) on the appropriate save command (SAVOBJ or SAVLIB command).
   3. Save all the physical files being journaled.

      Note: These files may or may not be in overflowed status.

b. To delete the objects that have been saved, do the following:

   1) Display the library in the user ASP or the library associated with the objects in the user ASP:

      ```
      WRKLIB LIB(library-name)
      ```

   2) Type a 12 (Work with objects) in the *Opt* column and press the Enter key.

   3) Find the objects to be deleted in the *Object* column.

   4) Type a 4 (Delete) in the *Opt* column for each object you want to delete.

   5) Press the Enter key.

   Journaling Consideration

   A journal that has files journaled to it cannot be deleted. To delete a journal, do the following:

   1. Delete the files that have their access paths journaled.
   2. Delete the journaled physical files.
   3. Delete the journal.

4. Verify the overflow condition is recovered by displaying the QSYSOPR message queue (DSPMSG QSYSOPR), or by using the System Service Tools (STRSST) command. If the user ASP is still in overflowed status, consider

using the procedures explained in "Recovering an Overflowed User ASP During an IPL" on page 7-80.

Task 3. Restore Objects to the User ASP

Restore the objects to the user ASP by using the appropriate restore commands (RSTOBJ or RSTLIB commands).

Journaling Consideration

To restore your journaling environment, do the following:

1. Restore the journal.
2. Restore the journaled physical files saved in the step where you saved and deleted the objects.
3. Restore the files that had their access paths journaled that were saved in the step where you saved and deleted the objects.

Recovering an Overflowed User ASP During an IPL

You can reset an overflowed user ASP by requesting the system to recover overflowed objects. If there is sufficient storage in the overflowed user ASP, the system moves all overflowed objects from the system ASP back to the user ASP. To make storage available, do the following tasks:

Task 1. Determine the Amount of Overflowed Storage

1. Type the following to display the System Service Tools (SST) menu.

   ```
   STRSST
   ```

   Press the Enter key. The following display is shown.

```
                    System Service Tools (SST)

Select one of the following:

     1. Start a service tool
     2. Work with active service tools
     3. Work with disk units
     4. Work with diskette data recovery



Select one of the following:
```

2. Select option 3 (Work with Disk Units) and press the Enter key. The following display is shown.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

3. Select option 1 (Display disk configuration) on the Work with Disk Units display and press the Enter key. The following display is shown.

```
                        Display Disk Configuration

 Select one of the following:

      1. Display disk configuration status
      2. Display disk configuration capacity
      3. Display disk configuration protection
      4. Display non-configured units
      5. Display device parity status
```

4. Select option 2 (Display disk configuration capacity) and press the Enter key.

```
                      Display Disk Configuration Capacity

                                                    --Protected--  --Unprotected--
ASP  Unit  Type  Model  Threshold  Overflow   Size  %Used    Size  %Used
 1                          90%    No            0   0.00%   1400   8.22%
       1   9332   400                           0   0.00%    200  17.97%
       2   9332   400                           0   0.00%    200   6.60%
       3   9332   400                           0   0.00%    200   6.59%
       4   9332   400                           0   0.00%    200   6.62%
       5   9332   400                           0   0.00%    200   6.60%
       6   9332   400                           0   0.00%    200   6.57%
       7   9332   400                           0   0.00%    200   6.59%
 2                                Yes           0   0.00%    200  99.99%
       8   9332   200       90%                 0   0.00%    200  99.99%
 3                                Yes           0   0.00%    200  99.99%
       9   9332   200       90%                 0   0.00%    200  99.99%
 4                                No            0   0.00%    200   6.61%
      10   9332   200       90%                 0   0.00%    200   6.61%

Press Enter to continue.

F3=Exit F5=Refresh  F9=Overflow information  F11=Non-configured units
F12=Cancel
```

Figure 7-6. Display Disk Configuration Capacity Display

5. Press F9 (Display ASP Overflow information) to display the overflow amount and the amount of storage needed to recover the overflowed objects.

```
                    Display ASP Overflow Information

                    Overflow      ----Amount Needed to Recover----
ASP  Threshold      Amount          To Capacity     To Threshold
 2      90%           14                0               0
 3      90%           25               25              45
```

Figure 7-7. Display ASP Overflow Information

6. The following information is used to determine the overflow amount and the amount of storage needed to recover the overflowed objects.
   - *ASP:* The auxiliary storage pool number.
   - *Threshold:* The current threshold setting.
   - *Overflow amount:* The amount of storage that has overflowed from the user ASP into the system in millions of bytes. An asterisk (*) indicates that the amount is not known. The possible cause is that the overflow has occurred before Version 2 Release 2.
   - *Amount Needed to Recover:* The amount of storage needed in the ASP the recover from the overflow condition.
   - *to Capacity:* The minimum amount of storage that must be made available to get all the overflowed data back into the user ASP.
   - *to Threshold:* The minimum amount of storage that needs to be made available before the recover overflowed ASP operation can move the overflowed data from the system ASP back to the user ASP and remain within the storage threshold of the ASP.

Task 2. Save and Delete the Objects in the User ASP

Do one or more of the following:

1. If there are detached journal receivers in the ASP, save them and then delete them.

   To save the journal receivers, do the following:

   a. Load a save tape

   b. Save the journal receivers contained in the user ASP:

   ```
   SAVOBJ OBJ(object-name) LIB(library-name) DEV(TAP01)
         OBJTYPE(*JRNRCV) VOL(*MOUNTED) ENDOPT(*UNLOAD)
   ```

   c. To delete the journal receivers, do the following:

   ```
   WRKLIB LIB(library-name)
   ```

   d. Type a 12 (Work with objects) in the *Opt* column and press the Enter key.

   e. Find the objects to be deleted in the *Object* column.

   f. Type a 4 (Delete) in the *Opt* column for each object you want to delete.

   g. Press the Enter key.

2. If some of the objects in the user ASP can be deleted because they are no longer needed, delete them.

   To delete the objects, do the following:

   a. Display the library in the user ASP or the library associated with the objects in the user ASP:

      ```
      WRKLIB LIB(library-name)
      ```

   b. Type a 12 (Work with objects) in the *Opt* column and press the Enter key.

   c. Find the objects to be deleted in the *Object* column.

   d. Type a 4 (Delete) in the *Opt* column for each object you want to delete.

   e. Press the Enter key.

3. If some of the objects are needed on the system, save the objects to a save file and then delete them:

   To save the objects in the ASP that are needed on the system, do the following:

   a. Save all the overflowed objects and then delete them.

      ```
      SAVOBJ OBJ(object-name) DEV(*SAVF) SAVF(save-file-name)
             OBJTYPE(*ALL)
      ```

      Or

      ```
      SAVLIB LIB(library-name) DEV(*SAVF) SAVF(save-file-name)
      ```

   Journaling Consideration

   If the objects you want to save and delete include one or more journals, do the following:

   1. Determine all physical files currently being journaled and all files with the access paths being journaled using the WRKJRNA command.
   2. Save all the files that have their access paths being journaled by specifying ACCPTH(*YES) on the appropriate save command (SAVOBJ or SAVLIB command).
   3. Save all the physical files being journaled.

      Note: These files may or may not be in overflowed status.

4. To delete objects that have been saved, do the following:

   a. Display the library in the user ASP or the library associated with the objects in the user ASP:

      ```
      WRKLIB LIB(library-name)
      ```

   b. Type a 12 (Work with objects) in the *Opt* column and press the Enter key.

   c. Find the objects to be deleted in the *Object* column.

   d. Type a 4 (Delete) in the *Opt* column for each object you want to delete.

   e. Press the Enter key.

Journaling Consideration

A journal that has files journaled to it cannot be deleted. To delete a journal, do the following:

1. Delete the files that have their access paths journaled.
2. Delete the journaled physical files.
3. Delete the journal.

5. Verify that there is sufficient storage to recover the overflowed objects by looking at the *Amount Needed* column on the Display ASP Overflow Information display. If the amount is not zero (0), then you still do not have sufficient storage to recover the overflowed objects. Repeat steps 3 through 4. If the amount is zero (0), you can request to recover the overflowed amount.

You can request the system to recover the overflow condition by adding or moving disk units to the overflowed ASP. For the procedures to add or move disk units to an ASP, see "Adding Units to an Existing ASP" on page 7-14 or "Moving a Disk Unit from an Existing ASP that Has Sufficient Storage" on page 7-28.

Task 3. Access DST Options

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .   _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

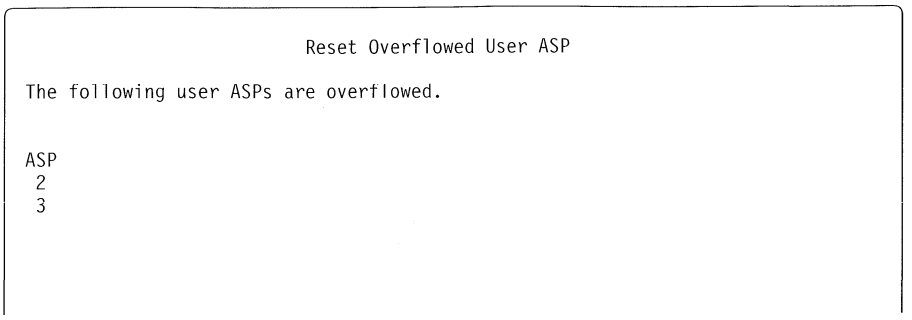
   The Use Dedicated Service Tools (DST) menu is shown.

```
                         Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     _

F3=Exit            F12=Cancel
```

Task 4. Recover Overflowed ASP

1. If you decide to add or move disk units to the overflowed ASP, see "Adding Units to an Existing ASP" on page 7-14 or "Moving a Disk Unit from an Existing ASP that Has Sufficient Storage" on page 7-28 for the procedures to add or move disk units to an ASP. Otherwise, continue with the next step.
2. Select option 1 (Perform an IPL) on the IPL or Install the System menu and press the Enter key.

   The Recover Overflowed User ASP display is show.



3. Press the Enter key to request recovery of the overflowed user ASPs. The recovery takes place during the storage management recovery phase of the IPL. The operation takes from several minutes to a few hours, depending on the number of objects on the system and the amount of data to be recovered.
4. When the IPL of the system is complete, the Sign On display is shown.
5. Verify the results of the ASP overflow recovery operation by displaying the QSYSOPR message queue:

   ```
   DSPMSG MSGQ(QSYSOPR)
   ```

Task 5. Restore Objects to the User ASP

Restore the objects to the user ASP using the appropriate restore commands (RSTOBJ or RSTLIB commands). Ensure there is sufficient storage in the ASP before restoring the objects.

Journaling Consideration

To restore your journaling environment, do the following:

1. Restore the journal.
2. Restore the journaled physical files saved in the step where you saved and deleted the objects.
3. Restore the files that had their access paths journaled that were saved in the step where you saved and deleted the objects.

Recovering an Overflowed User ASP by Deleting the User ASP Data

Use this procedure if you do not need any of the data in the user ASP on the system. For example, save files can be saved to tape and then deleted.

Task 1. Save the Security Data

Use the Save Security Data (SAVSECDTA) command to save the security data. By entering this command now, you can save all private object authorities. If you do not do this now, you must manually change private authorities to every object (using EDTOBJAUT command) that resides in the user ASP.

To save the security data, do the following:

1. To change the system operator message queue so all messages will appear on the display, type the following and press the Enter key.

```
CHGMSGQ QSYSOPR *BREAK SEV(60)
```

2. Load the first tape, and make the tape device ready.
3. To save the security data, type the following and press the Enter key.

   ```
   SAVSECDTA DEV(TAP01)
   ```

   The following may occur:

Task 2. Save the Objects in the User ASP

1. Save all the objects in the user ASP with the appropriate save command:

   ```
   SAVLIB LIB(user-ASP-library-name) DEV(TAP01) LABEL(label-name)
   ```

   Or,

   ```
   SAVOBJ OBJ(object-name) LIB(library-name) DEV(TAP01) OBJTYPE(*ALL)
          VOL(*MOUNTED) ENDOPT(*LEAVE)
   ```

2. If the overflowed user ASP contains journals and the files being journaled, save the library containing the journals and journaled files.

   ```
   SAVLIB LIB(library-name) DEV(TAP01) OBJTYPE(*FILE)
          VOL(*MOUNTED) ENDOPT(*UNLOAD)
   ```

Journaling Consideration

If the objects you want to save and delete include one or more journals, do the following:

1. Determine all physical files currently being journaled and all files with the access paths being journaled using the WRKJRNA command.
2. Save all the files that have their access paths being journaled by specifying ACCPTH(*YES) on the appropriate save command (SAVOBJ or SAVLIB command).
3. Save all the physical files being journaled.

   Note: These files may or may not be in overflowed status.

Task 3. Delete the Objects in the User ASP

Delete all objects in the user ASP that has overflowed before deleting the ASP data. This avoids having any pointer in the system ASP to objects that are destroyed in the user ASP, and having partial objects in the system ASP.

1. Display the library in the user ASP:

   ```
   WRKLIB LIB(library-name)
   ```

2. Type a 12 (Work with objects) in the *Opt* column.
3. Press the Enter key.
4. Find the objects to be deleted in the *Object* column.
5. Type a 4 (Delete) in the *Opt* column for each object you want to delete.
6. Press the Enter key.

Journaling Consideration

A journal that has files journaled to it cannot be deleted. To delete a journal, do the following:

1. Delete the files that have their access paths journaled.
2. Delete the journaled physical files.
3. Delete the journal.

Task 4. Access DST Options

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                        Use Dedicated Service Tools (DST)

Select one of the following:

           1. Perform an IPL
           2. Install the operating system
           3. Work with licensed internal code
           4. Work with disk units
           5. Work with DST environment
           6. Select DST console mode
           7. Start a service tool
           8. Perform automatic installation of the operating system
           9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Task 5. Delete the ASP Data

1. Select option 4 (Work with disk unit) and press the Enter key.

```
                            Work with Disk Units

 Select one of the following:

      1. Work with disk configuration
      2. Analyze disk device problem
      3. Work with disk unit recovery
      4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display and press the Enter key.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display and press the Enter key.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 5 (Delete ASP data) on the Work with ASP Configuration display and press the Enter key.

```
                    Select ASP to Delete Data From

Type options, press Enter
  4=Delete ASP data

Option  ASP  Threshold  Overflow    --Protected--  --Unprotected
                                      Size  %Used     Size  %Used
          1       90%       No        0.00  0.00%    1200  74.84%
          2       90%      Yes        0.00  0.00%     200  99.99%
          3       90%      Yes        0.00  0.00%     200  99.99%
```

5. Type a 4 in the *Option* column to select the ASP you want to delete the data from and press the Enter key. The following display is shown.

```
                         Confirm Delete ASP Data

Warning:  All data will be deleted from the selected ASPs.

Press F10 to confirm your choice for 4=Delete ASP data.
Press F12=Cancel to return to change your choice.

                                    --Protected--  --Unprotected--
Option  ASP  Threshold  Overflow     Size  %Used     Size  %Used
  4       2      90%       Yes         0   0.00      200  0.53%
  4       3      90%       Yes         0   0.00      200  0.53%
```

6. Press F10 (Confirm) to confirm your choice to delete the ASP data.

   The system deletes all the data in all the disk units in this user ASP and resets the overflow status for the user ASP.

   Note: This option does not delete any data that has overflowed from the user ASP into the system ASP.

7. When the ASP data is deleted, you return to the Use Dedicated Service Tools (DST) menu.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```


Task 6. Restore the Objects to the User ASP

1. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools (DST) menu.
2. On the IPL or Install the System menu, select option 1 (Perform an IPL).

   After the IPL is complete, the command entry display appears and you can proceed to the next step.

3. Load the correct volume of the SAVSECDTA tapes.
4. Restore the user profiles.

   ```
   RSTUSRPRF USRPRF(*ALL)
   ```

5. Load the correct volume of the RSTLIB or RSTOBJ tapes.
6. When restoring the saved objects after the overflow condition has been reset, restore the journals before restoring the journaled files to ensure that journaling is started again for the files. Use the appropriate restore command to restore the desired objects to the ASP.

   ```
   RSTLIB SAVLIB(library-name) DEV(TAP01) VOL(*MOUNTED)
   ```

   Or,

   ```
   RSTOBJ OBJ(*ALL) LIB(library-name) OBJTYPE(*ALL)
          ENDOPT(*REWIND) MBROPT(*ALL)
   ```

7. Restore the private authorities.

   RSTAUT

   If any objects were restored into a library other than the library from which they were saved, you must manually grant private authorities for the restored objects using the EDTOBJAUT command.

8. If the user ASP contained journal receivers associated with journals in another ASP, create a new journal receiver for each of these journals and run the CHGJRN command to attach the new receiver to the journal.

   Journaling Consideration

   To restore your journaling environment, do the following:

   1. Restore the journal.
   2. Restore the journaled physical files saved in the step where you saved and deleted the objects.
   3. Restore the files that had their access paths journaled that were saved in the step where you saved and deleted the objects.

9. This completes the steps to reset the overflowed status of a user ASP.

Working with Objects in User ASPs

The following ASP operations are described in this topic:

- Creating objects in a user ASP
- Transferring journals, journal receivers, or save files between ASPs
- Deleting objects in a user ASP
- Displaying objects in a user ASP
- Transferring existing journals into a user ASP
- Transferring existing journal receivers into a user ASP
- Moving journal receivers from an overflowed user ASP
- Moving journals from an overflowed user ASP
- Moving journals in an overflowed user ASP to a different ASP
- Moving journal receivers in an overflowed user ASP to a different ASP

The following command language (CL) commands support these ASP operations by providing an ASP or RSTASP parameter:

- Create Journal (CRTJRN)
- Create Journal Receiver (CRTJRNRCV)
- Create Library (CRTLIB)
- Create Save File (CRTSAVF)
- Restore Object (RSTOBJ)
- Restore Library (RSTLIB)
- Display Object Description (DSPOBJD)
- Display File Description (DSPFD)
- Work with Journal Attributes (WRKJRNA)
- Display Journal Receiver Attributes (DSPJRNRCVA)

For additional information on these commands, refer to the *CL Reference*.

Creating Objects in a User ASP

You may find it easier to display the contents of a user ASP and to save and restore it by creating a separate library for each user ASP on your system (such as ASP2LIB in the following example).

Once your user ASPs are configured, you can place libraries or objects in them as follows:

1. Create the libraries directly in the ASP by specifying a value on the ASP parameter on the Create Library (CRTLIB) command and then create the objects in the library. All objects created in a library in a user ASP are automatically created in the same user ASP.

   For example, to create a journal receiver in ASP 2, do the following:

   ```
   CRTLIB LIB(ASP2LIB) ASP(2)

   CRTJRNRCV  JRNRCV(ASP2LIB/RCVINASP2)
   ```

   This is the recommended way of creating objects in a user ASP.

2. Objects can be created directly in an ASP by specifying a value on the ASP parameter on the Create command for some objects. However, the library that the object is being created in must be in either the system ASP or in the same user ASP that the object is being created in. A user ASP cannot contain a library if it contains journals, journal receivers, or save files whose library is in the system ASP. If a library exists in the user ASP, you cannot create journals, journal receivers, and save files in the same ASP unless the library for the journals, journal receivers, or save files is in the same ASP. For example,

   CRTSAVF FILE(ASP2LIB/TEST) ASP(2)

   where (2) is the number of the user ASP where you are placing the save file. The library for the save file is in the system ASP and ASP 2 does not contain any libraries.

   Once the object is created, all storage for the object resides in the designated user ASP. Changes and additions to that object are also made in the user ASP. If the ASP becomes full, it overflows into the system ASP. For additional information, see "Considerations for Recovering an Overflowed User ASP" on page 7-75.

   It is recommended that all journals and journal receivers on the system have unique names. RCLSTG renames them if duplicate names are found when objects are placed in library QRCL and the user cannot rename them to their original name.

   Monitor the size of objects to prevent them from overflowing into the system ASP with the MAXRCDS parameter on the CRTSAVF command, and the THRESHOLD parameter on the CRTJRNRCV command.

Transferring Objects between ASPs

You cannot directly move objects between ASPs. However, you can transfer them from one ASP into another ASP by using the save and restore commands (the RSTASP parameter on the RSTOBJ and RSTLIB commands).

If you try to restore an object to a user ASP by explicitly specifying the desired user ASP for the RSTASP parameter and the designated user ASP does not exist, or it does exist but contains a library, a message is sent to the user indicating that the

object is not restored. However, if the object was originally in a user ASP and the user ASP does not exist at the time of the restore operation, and the default value RSTASP(*SAVASP) is specified, the object is restored to the system ASP and an informational message is sent to the user. The object is restored to the user ASP from which it was saved (as long as that ASP exists on your system) unless you specify a different ASP number on the restore command (with the RSTASP parameter).

If you attempt to restore objects (other than journals, journal receivers, and save files) to a user ASP other than the user ASP that contains their library, the objects are automatically restored to the ASP where their libraries are found. An informational message is sent to the user.

Private authorizations cannot be saved or restored with an object. You must manually grant private authorities to your objects after they are restored. If you have several objects to be moved that have many private authorities, consider using the Save System (SAVSYS) or the Save Security Data (SAVSECDTA) command. This command saves all private authorities and user profiles that are restored with the Restore User Profile (RSTUSRPRF) and Restore Authority (RSTAUT) commands.

To transfer a library or an object from one ASP to another ASP, do the following:

1. Save the library or the object using either the SAVLIB or SAVOBJ command.
2. Delete the library or the object.
3. Restore the library or the object to the new ASP by specifying:

   ```
   RSTLIB(ASPNLIB) RSTASP(N)
   ```

   where (N) specifies the number of the ASP to which you want the library restored.

   or

   ```
   RSTOBJ OBJ(OBJNAME) RSTASP(N)
   ```

Deleting Objects in a User ASP

To delete libraries or objects from a user ASP, use the appropriate delete commands to delete or clear the library that contains the objects.

Displaying Objects in a User ASP

There is no specific CL command to display objects contained in a user ASP. If you want to display a list:

1. Do one of the following (the first method is recommended):
   a. Create the library in the user ASP and then create the objects in the library. For example:

   ```
   CRTLIB LIB(ASP2LIB) ASP(2)

   CRTSRCPF FILE(ASP2LIB/SRC)

   CRTSAVF FILE(ASP2LIB/SAVF)
             .
             .
             .
   ```

b. Create a separate library in the system ASP for objects you want to create in the user ASP. When you create an object, be sure that the object is created in that library. For example:

```
CRTLIB LIB(LIBFOR2)

CRTSAVF FILE(LIBFOR2/SAVF) ASP(2)
          .
          .
          .

CRTLIB LIB(LIBFOR15)

CRTJRNRCV JRNRCV(LIBFOR15) ASP(15)

CRTJRN JRN(LIBFOR15/JOURNAL) JRNRCV(LIBFOR15/RECEIVER)
       ASP(15)
```

2. To display these objects, enter the Display Library (DSPLIB) command. For example, to display all objects created in LIBFOR15 in ASP 15, do the following:

```
DSPLIB LIB(LIBFOR15)
```

If you do not create separate libraries for your user ASPs, you cannot directly display objects in a user ASP. There are indirect methods that enable you to identify objects:

1. You can determine which ASP an object is in by using the DSPOBJD command and looking at the number shown on the *Auxiliary storage pool* field.
2. You can display all the object descriptions on your system to an output file using the DSPOBJD command, query the file, and search for objects in the ASP.

Transferring Existing Journals and Files into a User ASP

If you use the recommended type of user ASP, the files being journaled and the journal should be in the same ASP. For the purpose of recovery as well as performance, it is recommended that the journal receiver be placed in a different user ASP. If a failure occurs in the ASP that contains the files and the journal, you do not lose both the files and the journaled changes which are in the receiver. For maximum system performance, do not place your files and journal receiver in the same user ASP. To do so causes contention between access to the file and access to the journal receiver.

There are two methods for transferring journals and journal receivers into a user ASP. The first method is recommended. This involves creating the library for the journal (or journal receiver) in the user ASP. The objects can either be restored or newly created in the library. The second method is to keep the library for the journal (or journal receiver) in the system ASP and either restore or create the objects in the user ASP. Notice that a user ASP can contain either isolated objects (journals, journal receivers, and save files) or libraries, but not both.

Method 1: To transfer an existing journal to a user ASP using the first method, do the following:

1. If you are going to restore the journal, use the SAVOBJ or the SAVLIB command to save the journal.

2. Because the library for the journal is restored or re-created in the user ASP, the files must also be moved to the same user ASP before you can resume journaling the files and access paths after the move.
3. Save any physical files being journaled and any logical files that have their access paths journaled.
4. Delete the physical files and logical files using the Delete File (DLTF) command.
5. Delete the journal using the Delete Journal (DLTJRN) command.
6. Create the library for the journal in the user ASP using the Create Library (CRTLIB) command and specify the desired ASP on the ASP parameter, or restore the library for the journal to the user ASP using the Restore Library (RSTLIB) command and specify the desired user ASP on the RSTASP parameter.
7. If you used the RSTLIB command to restore the library for the journal in step 6 you can skip this step. Otherwise, create the journal in the ASP using the Create Journal (CRTJRN) command, or restore the journal to the library in the user ASP using the Restore Object (RSTOBJ) command.
8. Restore the physical files and logical files to the libraries in the user ASP. If you want to restore the files to their original libraries, you must first move those libraries to the user ASP. Restoring the files automatically resumes journaling for the files if the journal already exists.
9. Save the files so that the journaled changes can be applied, if necessary. To save the files, see "Saving Journaled Files" on page 2-7.

Method 2: To transfer an existing journal into a user ASP using the second method, do the following:

1. If you are going to restore the journal, save the journal with either the SAVOBJ or SAVLIB command.
2. Optionally, save any physical files being journaled and any logical files whose access paths are being journaled. This allows you to begin journaling later by simply restoring the files.
3. You may want to use the SAVSYS command to save the system. By using this command now, you can save all private object authorities. If you do not do this, you must manually restore private authorities with the EDTOBJAUT command.
4. You can end journaling of access paths using the ENDJRNAP command. For physical files, use the ENDJRNPF command.

   If you saved the physical and logical files, delete those files with the DLTF command.
5. Delete the journal with the DLTJRN command.
6. Restore the journal to the user ASP by specifying:

   RSTOBJ OBJ(XXX) OBJTYPE(*JRN) RSTASP(N)

   where (N) is the number of the ASP to which the journal is restored. You can also create a journal with the ASP parameter on the Create Journal (CRTJRN) command.

7. If you deleted the physical and logical files in step 4, restore these files to begin journaling using the RSTOBJ or RSTLIB command. Otherwise, start journaling with the STRJRNPF and STRJRNAP commands.
8. Reestablish private authorities to the journal and the database files, if they were deleted, with the EDTOBJAUT command. If you saved the system, restore your user profiles and authorities with the RSTUSRPRF and RSTAUT commands.
9. This completes the steps to transfer an existing journal to a user ASP.

Changing to Journal Receiver on a User ASP

There are two methods you can use to change to journal receivers on user ASPs. The first method is recommended.

Method 1:

1. Create a library in the user ASP using the Create Library (CRTLIB) command and specify the desired user ASP on the ASP parameter.
2. Create a new journal receiver in the library in a user ASP using the Create Journal Receiver (CRTJRNRCV) command.
3. Change journal receivers so the new journal receiver is attached and actively receiving journal entries by specifying:

   CHGJRN JRN(XX) JRNRCV(YY)

   where (XX) is the name of the journal, and (YY) is the name of the new journal receiver.

   You can save the detached journal receiver and delete it from the system.

Method 2:

1. Create a new receiver (CRTJRNRCV command) in a user ASP by specifying a user ASP name on the ASP parameter when the library for the journal receiver is in the system ASP.
2. Change journal receivers so the new journal receiver is attached and actively receiving journal entries by specifying:

   CHGJRN JRN(XX) JRNRCV(YY)

   where (XX) is the name of the journal, and (YY) is the name of the new journal receiver.

Moving Journal Receivers From an Overflowed User ASP to a Different ASP

To maintain journaling for the files, do the following steps:

1. Use the Display Journal Receiver Attributes (DSPJRNRCVA) command to determine the names of the journal receivers associated with the journal.
2. If the journal receivers to be moved are attached to a journal, create a new journal receiver on a different ASP using the CRTJRNRCV command. Consider using a name for the journal receiver that continues your naming conventions.
3. Change the journal using the Change Journal (CHGJRN) command, and specify the newly created journal receiver on the JRNRCV parameter.

4. Save the journal receivers from the overflowed user ASP that are associated with the journal using the Save Library (SAVLIB) or the Save Object (SAVOBJ) command.
5. Delete the library from the overflowed user ASP using the DLTLIB or DLTJRNRCV command.
6. Restore the library and receivers to the different ASP using the RSTLIB or RSTOBJ command.

Use the Work Journal Attributes (WRKJRNA) command to verify the restored journal receivers are now associated with the same journal and that the files are being journaled correctly.

Moving a Journal From an Overflowed User ASP to a Different ASP

To maintain journaling for the files, do the following steps:

1. Save the journal that is being moved to a different ASP using the Save Library (SAVLIB) command
2. Save the files that are currently being journaled using the SAVLIB command.

   By saving the files while they are still being journaled, the system will automatically start journaling again when you restore the files after restoring the journal.
3. End journaling for any access paths being journaled by using the End Journaling Access Path (ENDJRNAP) command.
4. End journaling for any physical files being journaled by using the End Journaling Physical Files (ENDJRNPF) command.
5. Use the Work Journal Attributes (WRKJRNA) command to determine the names of the journal receivers associated with the journal.
6. Delete the journal using the Delete Journal (DLTJRN) command.
7. Delete all the journaled files and access paths using the Delete File (DLTF) command.
8. Save the journal receivers that were associated with the journal being moved using the Save Object (SAVOBJ) command.
9. Delete the library for the journal using the Delete Library (DLTLIB) command.
10. Restore the library for the journal to the different ASP using the Restore Library (RSTLIB) command. This will create a new journal receiver and journal.
11. Restore the journaled files using the Restore Library (RSTLIB) command.
12. Restore the receivers saved in step 1 using the Restore Object (RSTOBJ) command.

Use the Work Journal Attributes (WRKJRNA) command to verify you have the same set of journal receivers associated with the journal in the different ASP and that the files are being journaled correctly. There will be a chain break between the newly attached receivers and the receivers that were just restored.

Part 4. Checksum Protection

Chapter 8. Introduction to Checksum Protection

Checksum protection is an option you can use to recover from a disk unit media failure in an ASP without having to reload the entire system. This option minimizes your vulnerability to data loss and reduces recovery time from disk unit media failures.

Note: It is important to understand checksum protection and how it might fit in your recovery strategy before you start using it.

Checksum protection should not be used as a replacement for system backup procedures. You should continue to back up your system on a regular basis because there are instances where checksum protection cannot be used to reconstruct the lost data.

How Checksum Protection Works

When checksum protection is configured, the system automatically groups the disk units in an ASP into checksum sets. Space equivalent to approximately one disk unit in each set is used to store checksum data that provides protection for the user data stored on the other units in the set. Checksum protection requires additional storage capacity (both main storage and auxiliary storage), as well as added processing unit capacity to achieve comparable performance to a system without checksum protection.

Checksum protection takes the data residing on a storage unit and distributes the data onto other storage units in the checksum set. The data is distributed in such a way that if any one of the units in the checksum set fails, its contents may be recovered by recombining the data on remaining units. Checksum protection operates like a parity bit. The parity bit is normally set based on the contents of a byte of data and allows the recovery of any bit within the byte that cannot be read. Checksum protection operates in the same manner except that the parity is dependent on input from more than one disk unit.

When a disk unit fails, the system becomes unusable regardless of whether checksum protection is active or not. Checksum protection does not prevent the system from ending abnormally. Rather, its key advantage is that it helps avoid restoring the licensed internal code, the operating system, and user data if the data on a failed unit is lost and the unit must be replaced.

When you first establish the checksum environment, some of the disk capacity is reserved for redundant checksum data. The system moves any current data from the reserved areas of the disk unit. Any changes you make to permanent objects are automatically updated and maintained in the checksum data (associated with the disk extents allocated to the objects). If any single storage unit in a checksum set is lost, the system reconstructs the contents of the lost storage unit after it is replaced. The reconstructed data reflects the most up-to-date information on the disk at the time of the failure.

Figure 8-1 on page 8-2 illustrates a system ASP configuration with checksum protection. In this example, assume that two of the three devices contain user application data, and the other device contains checksum data.

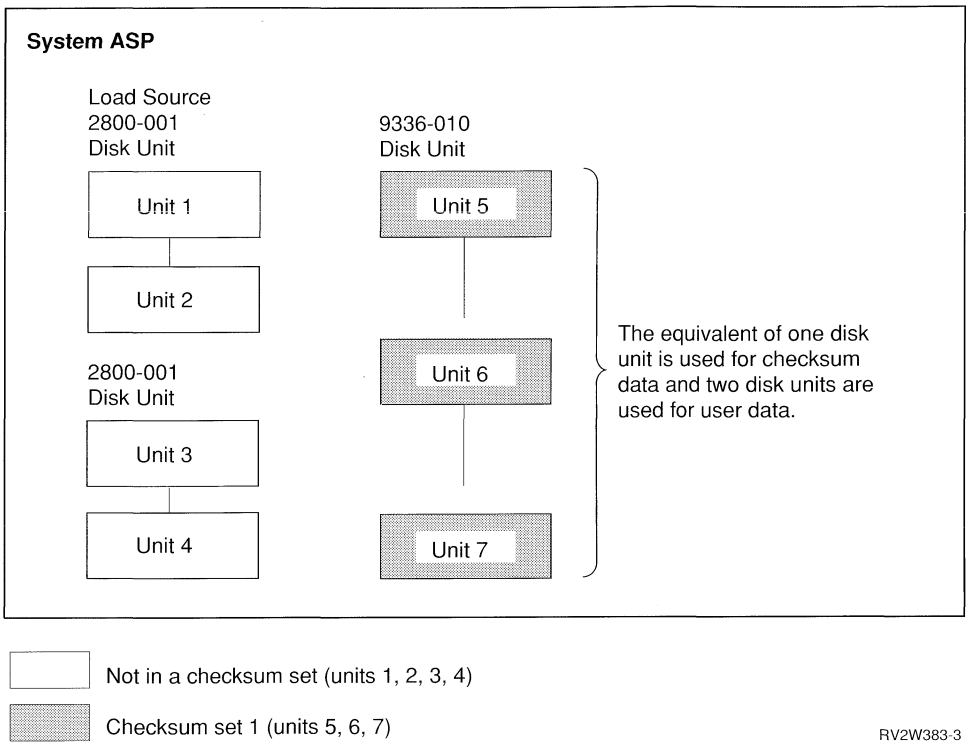


Figure 8-1. System ASP with Checksum Protection

The assignment of storage units to checksum sets is shown for example purposes only. The actual assignment of checksum sets can vary.

The implementation of checksum protection is more complex because there are multiple checksum areas spread across all units in a checksum set to distribute the disk activity more evenly over all units in a checksum set.

If the user data on storage unit 6 is lost, the system automatically reconstructs that data from the checksum data on storage units 5 and 7 after unit 6 is replaced.

Additional Considerations

Checksum protection gives you the advantage of being able to recover from a disk unit failure, but there is a performance and resource cost. You may need to do one or more of the following before you start using checksum protection:

- Add more disk units.
- Add more main storage.
- Upgrade to a higher performance processor model.

Checksum Recovery Limitations:

Although checksum protection can fully recover from several disk unit media failures, it is not a substitute for save procedures. There are some types of disk unit failures that do require backup media. For example:

- Checksum protection cannot recover data if a second storage unit in the same checksum set fails and the data cannot be recovered before the data from the first failure is fully recovered.

  In this case, all units in the ASP must be cleared. You must be prepared to restore the ASP from your backup media and apply your journal changes. The *Basic Backup and Recovery Guide* has more information about how to reload the system.

- If the system abnormally ends without saving main storage, disk sectors affected by I/O operations in progress may not be recovered, and you may have some damaged objects.

After the system recovers data using checksum protection, message queue QHST contains messages identifying any objects that could not be completely recovered. This is especially useful if a power loss prevents the saving of main storage.

System Performance When Using Checksum Protection: When checksum protection is active, additional storage capacity (both main and auxiliary storage) and added processing unit capacity are required to achieve performance comparable to a system without checksum protection. Even with the addition of more resources to your system, you can still experience a decrease in performance with checksum protection.

Because of the additional requirements, you must plan carefully to determine the additional system resources that are required to get a similar level of performance. Planning for additional auxiliary storage requirements should be done using the information provided later in this chapter. Use the capacity planning tool to plan for additional main storage and requirements processing which is a part of the AS/400 Performance Tools package. You can determine the effect checksum protection does have on system performance by using the capacity planning tool to predict the effect that checksum protection will have.

In addition to the capacity planner, the following are some general guidelines that can help determine what effect checksum protection does have on main storage and processor utilization. However, these guidelines should not be used as a replacement for the capacity planning tool. The capacity planning tool should still be your primary source for obtaining information about checksum protection.

- Checksum protection uses approximately 5% of main storage for internal operations. This amount should be added to the machine pool size. In addition, checksum protection uses an additional 40KB of main storage for each storage unit that is part of a checksum set.
- Checksum protection requires at least 10% additional system processing power or increased system utilization. Because checksum protection can slow system performance, you may need additional processing capacity.

  For example, if you monitor your current processor usage and find that it is running at approximately 60% of its capability during heavy usage, you can expect to add at least another 10% with checksum protection. Your system would run at 70% of its processing capability during heavy usage with checksum protection specified. This percentage leaves little room for additional usage or growth. In this situation, you may decide a faster processing unit is needed.

It is important to understand that, even with the addition of more resource to your system, you can still experience a decrease in performance because of checksum protection. The amount can vary for each job, depending on how many disk write operations are done by the job. The more write operations done by a job, the greater the decrease in performance for the job when checksum protection is in effect. Some of the operations that generally experience a decrease in performance are:

- Extensive updating of the database
- Journaling
- Commitment control
- Copying a file (Copy File (CPYF) command)
- Restoring
- Saving to save files
- Replacing edited source members
- Reorganizing physical file members (Reorganize Physical File Member (RGZPFM) command)

Because performance is reduced as the number of write operations increases, a restore of the system ASP or of all the libraries in a user ASP can take two to three times longer than your normal restore time.

It is recommended that you implement user ASPs without checksum protection for journaling, commitment control, and saving to save files. Because checksum protection is done by ASP, the unprotected ASPs will not be affected by the performance impact of checksum protection. Journals, journal receivers, and save files are all forms of protection from data loss. It is not necessary to protect an ASP containing these objects using checksum protection.

Batch jobs must also be given special consideration when implementing checksum protection. Because batch jobs are often disk input/output intensive, they can perform many more write operations than an interactive job. Therefore batch jobs can experience a significant decrease in performance. Batch jobs can take two to three times longer, depending on the number of write oper-

ations. To predict more accurately the effects of checksum protection for interactive and batch jobs, use the *Performance Tools/400 Guide*.

Additional IPL Time: In some cases, if checksum protection is active and the last system end was abnormal, the system cannot be sure that the most up-to-date information was written to disk before the system ended. In that case, the system does read all the data in the ASP to verify checksum data. This validation process occurs during the IPL following the abnormal end, and it can be a lengthy process. If the system can save a copy of main storage before it ends, the lengthy checksum validation process may be avoided.

If the system loses power frequently, it is strongly recommended that you install an uninterruptible power supply when you start checksum protection. Should main power be lost, an uninterruptible power supply allows the system to stop normally, preventing most long checksum validations.

Planning Storage Capacity

This section provides information on your main storage requirements, understanding your current storage use, determining the amount of unprotected and protected storage you need, and disk unit requirements.

Main Storage Requirements

Checksum protection uses approximately 40 000 bytes of main storage for each storage unit in a checksum set.

Adding additional main storage may lessen, but not avoid, slower performance. Generally, operations that do several write operations (such as updating database records) slow system performance on a checksum-protected system unless extra main storage is added.

Moving your journal receivers or selected user libraries to an unprotected or mirror protected user ASP can reduce the number of write operations required for checksum protected data. This reduces the amount of main storage required to provide for acceptable performance. However, it is not recommended to use both checksum and mirrored protection on the same system.

Understanding Current Storage Use

After checksum protection is started, portions of auxiliary storage are used for checksum redundancy information, reducing the overall storage capacity of your disk configuration. Accordingly, you must understand how much auxiliary storage space you are using for your normal operations before you start checksum protection.

Auxiliary storage for the system ASP with checksum protection is divided into two distinct areas of protected storage (for permanent data) and unprotected storage (for temporary data):

- Protected storage contains permanent objects, such as database files and program objects.
- Unprotected storage contains machine data and temporary objects, such as file open data paths and compiler work areas.

Note: When checksum protection is started for the system ASP, you define the amount of unprotected storage for each storage unit in a checksum set.

Auxiliary storage in a user ASP that has checksum protection does not have this division of protected and unprotected storage. All available storage is protected storage.

Before configuring checksum protection for an ASP, monitor your system for a representative period to determine the total amount of protected and unprotected storage you are currently using. To determine the amount of unprotected storage, see the topic "Determining the Amount of Unprotected Storage You Need for the System ASP" on page 9-4.

The value shown on the Work with System Status display when calculating the maximum amount of unprotected storage allocated since the last IPL is useful only if it is a representative example of your system's work. Also consider any special applications you may run, as well as peak activity needs, when planning for your system storage.

It is recommended that, when deciding how to configure your system, you allow a large buffer of space for unprotected storage for growth. It is better to overestimate the amount of space

needed. If you incorrectly estimate the amount, it is easier to decrease the unprotected storage space than it is to increase it. Increasing the amount causes the ASP to be cleared of all data. You must save and restore all data in the ASP in this case.

How Unprotected Storage Is Used

To determine the amount of space to allocate for unprotected storage, you must first consider how the space is to be used.

Note: Unprotected storage has meaning only for the system ASP that has checksum protection.

Unprotected storage is used for:

- Running jobs. The Work with Active Jobs (WRKACTJOB) display shows, at the top of the display, the number of jobs currently active in the system. Each job requires unprotected storage for internal work areas. You can display this value for a job using the Work with Jobs (WRKJOB) command and selecting the Job Run Attributes display. The temporary (unprotected) storage used value appears at the bottom of the Job Run Attributes display, and is shown as kilobytes.

  Sample several jobs to determine the averages for your system. Certain jobs (for example, certain office products) may require a large amount of temporary storage.
- Storing machine data. Unit 1 is used mainly for licensed internal code and machine data. The entire size of unit 1 is included in the unprotected amount used values shown on the Work with System Status (WRKSYSSTS command) display (when checksum protection is not active).

Your *unprotected storage* allocation should be the total of:

- The amount of unprotected storage already used immediately after you do an IPL, and before other users sign on.
- The average amount of temporary (unprotected) storage used per job multiplied by the total number of jobs you expect to support.

To verify that the value you calculate is appropriate for your needs, use the Work with System Status (WRKSYSSTS) command to show the current amount of unprotected storage used and the maximum amount of unprotected storage used since the last IPL of the system.

In addition to growth and peak levels of activity, also consider whether you now use, or plan to use, group jobs, advanced program-to-program communications (APPC), Structured Query Language/400 (SQL/400*), AS/400 Query, or the dynamic selection (DYNSEL) attribute on any of your files. Group jobs and APPC increase the number of active jobs for your system. SQL/400, Query, and the DYNSEL attribute increase the amount of temporary storage used by the system.

Disk Unit Requirements

After you have assessed your storage capacity requirements, you must determine the equipment you need to satisfy those requirements. Use the SST Calculate Checksum Configuration display to determine how much protected (permanent) and unprotected (temporary) storage in the system ASP will be available for a given disk configuration when checksum protection is started. For more information about doing this, see the topic "Calculating a Disk Configuration for Checksum Protection" on page 9-4.

You can evaluate whether the amount of auxiliary storage on your system is sufficient to meet your permanent and temporary storage requirements. If your current disk configuration does not support a checksum environment, use this option to help determine the disk configuration that does support that environment.

Using the checksum protection function requires that you have enough storage for the checksum data area for each checksum set. The amount of additional storage capacity required depends on the size and number of the checksum sets created. Each checksum set requires the equivalent of approximately one unit for checksum protection purposes.

For example, a checksum set with three storage units requires the equivalent storage of one of those units to be used for the checksum data. Therefore, only 67 percent of the storage space is actually available for protected and unprotected storage. The other 33 percent of the storage space contains the checksum data.

Table 8-1 on page 8-6 illustrates the space available for protected and unprotected storage as additional storage units are added.

Table 8-1. Space Available per Number of Units

Number of Storage Units in a Checksum Set	Storage Available per Checksum Set	Checksum Storage Used per Checksum Set
2	50%	50%
3	67%	33%
4	75%	25%
5	80%	20%
6	84%	16%
7	86%	14%
8	88%	12%

The following rules and restrictions determine the checksum set configuration:

- The requirement that each checksum set contains from two to eight units.
- The similarity of the disk unit sizes. Each storage unit in the checksum set must be of the same size. For example, you could have one checksum set with four 9332 model 200 disk unit. You cannot, however, have a checksum set with one 9332 and one 9335 disk unit.

  The 9332 models 200 and 400 disk units can be in the same checksum set. You can have multiple sets of different disk unit types (for example, you can have three 9332 units in a checksum set and four 9335 units in a different checksum set in the same ASP).
- The restriction that no two storage units of a checksum set can be on the same replaceable disk unit applies to the 9332 and 9335 disk units only.
- The restriction that an ASP with checksum protection cannot contain disk units that use device parity protection.

  Note: Storage units in the same replaceable disk unit have the same serial number.
- The restriction that unit 1 and all other 2800 model 001 storage units are not eligible for checksum protection.

For example, if you currently have one 9335 (two storage units total), you need at least one additional 9335 disk unit to begin checksum protection. This additional disk unit must be added because a minimum of two units must be in each checksum set, and because no two units can be on the same replaceable disk unit.

The 9336 model 010 disk unit contains from two to four storage units that can be replaced separately within the same disk unit. Therefore, the storage units within the disk unit can be in the same checksum set.

Because the 2800 model 001 units cannot be placed in a checksum set, the additional capacity they provide (beyond that needed for the licensed internal code and data areas) can only be used for the storage of unprotected (temporary) objects. The 2800-001 units that are not used for the licensed internal code and data areas can be assigned to a user ASP.

Warning: See note 3 on page 6-10 for restrictions.

When checksum protection is in effect, storage units in the system ASP that are not in a checksum set are used for storage of unprotected data. For example, if you have only one disk unit of a particular type, the disk unit is not checksum protected. Only unprotected data (such as temporary objects) is stored on that device.

Storage units that are not in a checksum set in a user ASP that has checksum protection are never used. These storage units should be either moved or removed from the user ASP. Before this type of configuration is allowed to occur, a warning is sent informing you of this situation.

Disk Configuration for Checksum Protection

You can use the SST Calculate Checksum Configuration display to calculate the amount of protected and unprotected storage capacity that would be provided if checksum protection were in effect for a particular disk configuration. You can calculate the storage capacity of your current system configuration or the storage capacity of additional disk units that you may add later by using a different configuration.

To calculate disk storage capacity, see the topic "Calculating a Disk Configuration for Checksum Protection" on page 9-4.

Examples of Checksum Protection Configurations

The following examples are provided to illustrate how different combinations of disk units can result in different checksum set configurations. The examples do not necessarily match the configuration put into effect on your system for the same set of disk units because the system uses an algorithm to determine the checksum sets. The unit numbers may be different.

Example of Checksum Configuration Using Three Checksum Sets: Assume your system ASP has two 2800 model 001 disk units configured, three additional 9335 disk units, and a 9336 model 010 disk unit configured for a total of fourteen storage units. After checksum protection is configured, the system ASP could have the configuration shown in Figure 8-2 on page 8-8.

Storage units 5, 6, 7, and 8 are configured in checksum set 1, and storage units 9, 11, and 13 are configured in checksum set 2, and storage units 10, 12, and 14 are configured in checksum set 3. Thus, if the data in storage unit 12 is lost, the system automatically reconstructs the lost data from the checksum data on storage units 10 and 14. The 2800 Model 001 storage units (1, 2, 3, and 4) that contain the licensed internal code and data areas cannot be placed in a checksum set.

Example of Checksum Configuration Using Two User ASPs: Assume your system ASP has two 2800 Model 001 disk units configured, that have a total of four storage units to contain the licensed internal code and data areas, and has a 9336 disk unit configured, with 4 storage units, for a total of 8 storage units.

Also, assume that you have three additional 9335 Model B01 and 9332 Model 400 disk units configured into two user ASPs. Assume ASP 3 contains a journal object, journal receivers, or one or more save files. ASP 2 contains other object types allowed in user ASPs.

The system could have the configuration shown in Figure 8-3 on page 8-9.

Placing the journal receiver in a user ASP that does not have checksum protection gives you two performance benefits:

- If extensive journaling is done, journal performance is better because the actuator arm is dedicated to the journal receiver.
- Write operations to a user ASP without checksum protection do not have the performance overhead of checksum protection.

Dual journal receivers can further improve the recovery capabilities. If you use this approach, the best performance occurs if the active journal receivers are in different user ASPs that do not have checksum protection. The best recovery occurs if the active journal receivers are not on the same replaceable disk unit.

Placing the save file in a user ASP also reduces the number of write operations in the system ASP that has checksum protection.

Example of an Ineligible Configuration for Checksum Protection: Assume your system ASP has the required 2800 model 001 units. In addition, you have the following configured disk units:

- One 9332 Model 400 disk unit (two storage units)
- One 9335 B01 disk unit (two storage units)

This configuration shown in Figure 8-4 on page 8-10 is not eligible for checksum configuration because you must have at least two disk units containing the same sized storage units allocated to your system ASP. To qualify for checksum protection, add a second 9335 or 9332 disk unit.

The 2800-001 units cannot be in a checksum set because they are reserved for use by the system for the licensed internal code, dump space, and so forth.

If you try to configure the units in this example, you will receive an error message when you use the DST Start Checksum Protection option.

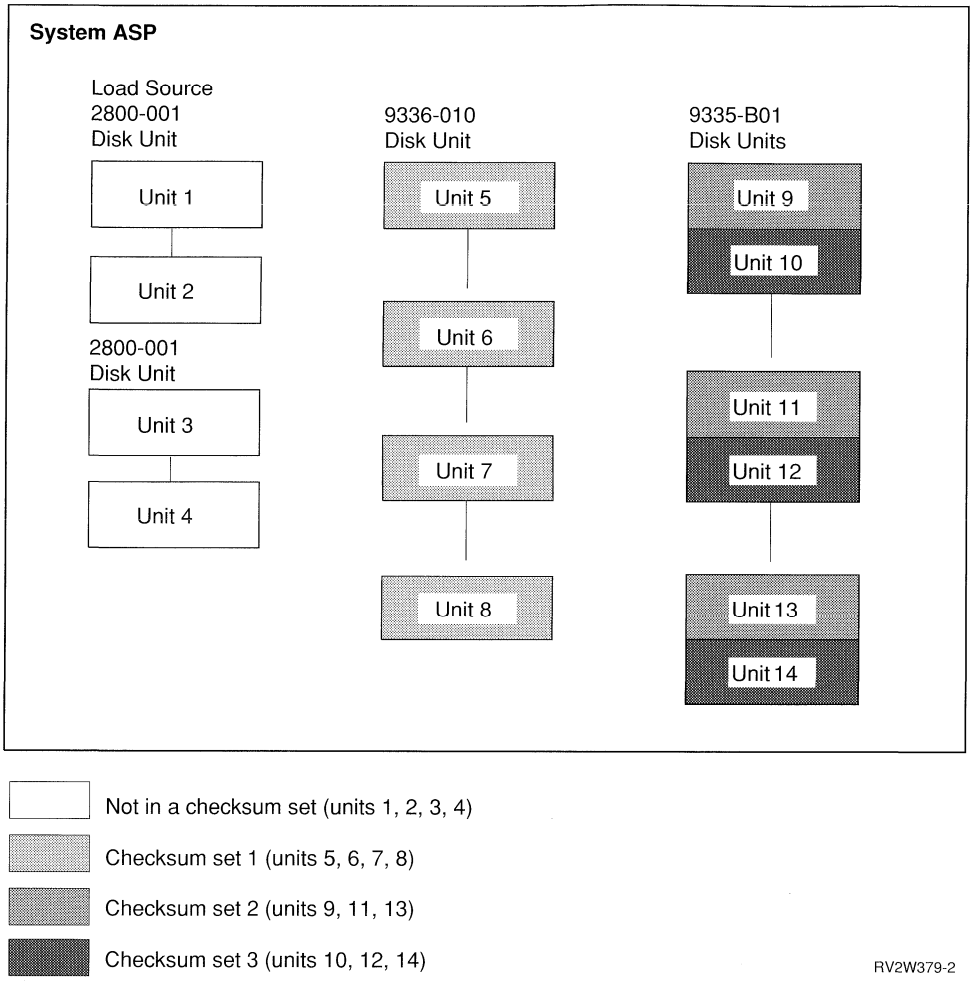


Figure 8-2. Example of Checksum Configuration Using Three Checksum Sets

Example of an Inefficient Checksum Configuration: Assume your system has the required 2800 model 001 storage units configured for the licensed internal code and data areas in addition to the following:

In ASP 1:

- Three 9332 Model 400 disk units (six storage units)
- One 9335 Model B01 disk unit (two storage units)

In ASP 2:

- One 9332 Model 400 disk unit (two storage units)
- One 9332 Model 200 disk unit (one storage unit)

Although this is a valid checksum configuration, this is not efficient use of storage because the space on the 9335 model B01 disk unit is used only for unprotected storage (your permanent user objects are not stored on the 9335 disk unit). Also, the 2800 model 001 storage units that are not used for the licensed internal code, data areas, and dump space provide unprotected storage. This can be inefficient in disk accessing performance because it can allow too much temporary data in one storage unit, risking contention. You could let the system use the space for unprotected storage, or you could place the 9335 disk unit into a separate user ASP.

In Figure 8-5 on page 8-11, unit 14 in the user ASP is not included in the checksum set because it is in the same replaceable disk unit as storage unit 13, in checksum set 3. There are no permanent user objects or temporary data stored on unit 14. You will get a warning message if you attempt to configure a user ASP with a storage unit that cannot be in a checksum set. If your configuration is already in this state, you may

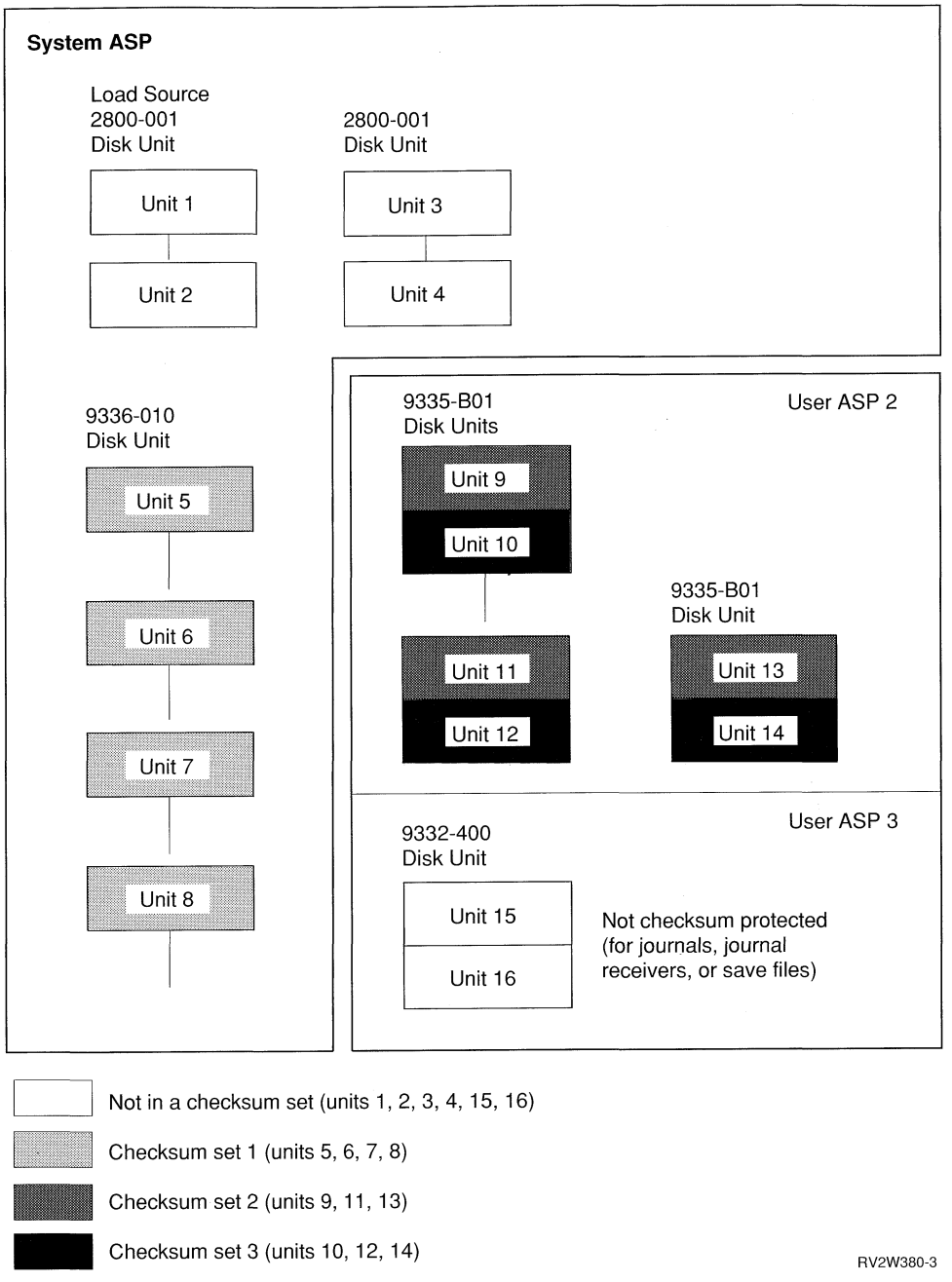


Figure 8-3. Example of Checksum Configuration Using Two Checksum Sets with Two User ASPs

remove the unit using DST without clearing the ASP.

Changing an Existing Checksum Configuration

When checksum protection is in effect, you can change your existing configuration in several ways. Any time you make configuration changes to your system, you should save your system to protect against unanticipated problems or errors that can occur during the reconfiguration.

The following changes do not require that you save and restore the ASP:

- Decreasing the amount of unprotected storage in the system ASP.
- Adding units to the system ASP or a user ASP while checksum protection is in effect.
- Moving a storage unit that is not in a checksum set from an ASP that has checksum protection to a different ASP, or

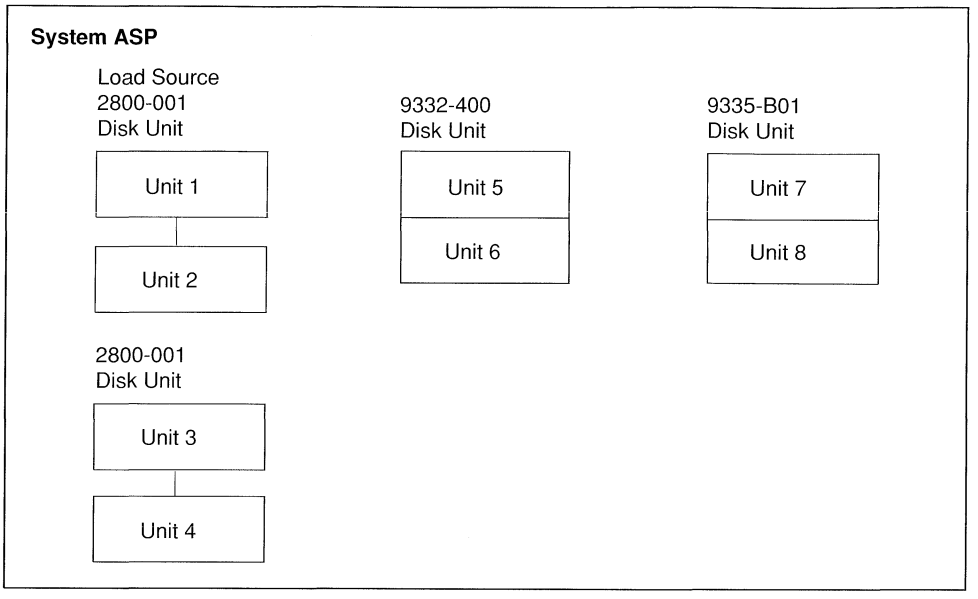


Figure 8-4. Example of an Ineligible Checksum Configuration

removing such a unit from an ASP that has checksum protection.

- Moving or removing disk units in a checksum set requires that checksum protection be stopped before the move or remove operation is allowed.
- Replacing one storage unit in a checksum set with a nonconfigured unit.

| Increasing the amount of unprotected storage in the system ASP stops and starts checksum protection.

Changing the Amount of Unprotected Storage in the System ASP

Assume your current unprotected storage amount is 10MB per unit, and you want to decrease it to 9MB. To do this use the DST Change Unprotected Storage for ASP1 option and specify the new value of 9 for the unprotected storage per unit value.

- Decreasing this value causes some additional processing during the next IPL.
- Increasing this value takes much longer because checksum protection is stopped and then started again (see "Handling Unprotected Storage Overflows" on page 9-20).

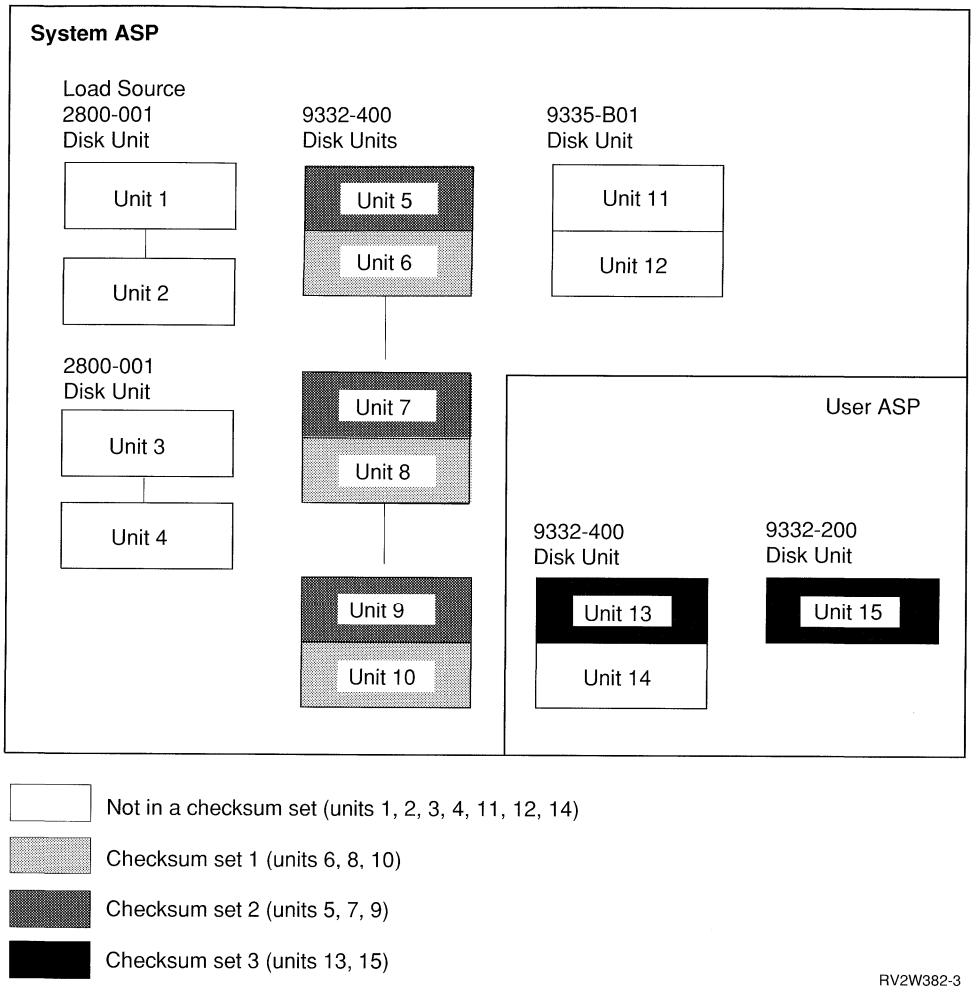


Figure 8-5. Example of an Inefficient Checksum Configuration

Chapter 9. Working with Checksum Protection

Before configuring checksum protection, monitor your system for a representative period to determine the total amount of protected and unprotected storage you are currently using:

1. Enter the Work With System Status (WRKSYSSTS) command to monitor system storage. (For additional information on this command, refer to the *CL Reference*.) This display shows the storage capacity in megabytes. Megabytes is represented by M on the display.
   - *System*: The capacity of auxiliary storage in the system ASP.
   - *% Used:* The current percentage of system ASP storage capacity that is allocated.

     When checksum protection is not in effect, these fields relate to the capacity in the system ASP for storage of both permanent and temporary objects.

     When checksum protection is in effect, these fields relate to the capacity in the system ASP for the storage of permanent objects only.
   - *Current unprotected used*: All temporary objects and machine data currently used on the system.
   - *Max unprotected*: The maximum amount of unprotected storage allocated since the last IPL of the system.
2. Use the values in the unprotected used fields to determine your current storage use for what would be unprotected storage under checksum protection:
   - When checksum protection is not in effect, these fields show the amount of unprotected storage that would be used if checksum protection were in effect.
   - When checksum protection is in effect, these fields show the amount of unprotected storage actually used.

   The value shown for the maximum amount of unprotected storage allocated since the last IPL is useful only if it is a representative example of your system's work. Also consider any special applications you may run as well as peak activity needs when planning for your system storage.

It is recommended that, when deciding how to configure your system, you allow a large buffer of space for unprotected storage for growth. It is better to overestimate the amount of space needed. If you incorrectly estimate the amount, it is easier to decrease the unprotected storage space than it is to increase it. Increasing the amount causes the ASP to be cleared of all data. You must save and restore all data in the ASP in this case.

Warning: When considering the removal of units from the system ASP (ASP 1), remember to leave at least two 2800-001 storage units in the system ASP for the 9406 model D70 and D80 system units that contain 224MB or more of main storage.

Determining the Amount of Protected Storage You Need for Checksum Protection in the System ASP

In the following example, the system ASP storage value is 1803M, 60.63% of the storage is being used, and 262M is the current amount of unprotected storage being used.

To determine the amount of protected storage needed, do the following:

1. Look at the Work with System Status (WRKSYSSTS command) display.

```
                        Work with System Status              RCH38342
                                                          02/26/90  14:31:46
% CPU used . . . . . . . :       29.5  Auxiliary storage:
Elapsed time . . . . . . :   00:00:00   System . . . . . . . . . :  1803 M
Jobs in system . . . . . :        159   % used . . . . . . . . . : 60.6241
% addresses used:                       Total  . . . . . . . . . :  1803 M
  Permanent  . . . . . . :      2.125   Current unprotect used :   262 M
  Temporary  . . . . . . :       .304   Maximum unprotect  . . :   265 M

Type pool size and activity level changes, press Enter.

System    Pool     Reserved    Max   -----DB-----  ---Non-DB---
 Pool   Size (K)  Size (K)  Active  Fault  Pages  Fault  Pages
   1       9183      4392     +++      .0     .0     .0     .0
   2       4000         0       6      .0     .0     .0     .0
   3      35619         0      59      .0     .0    7.0   16.3
   4        350         0       5      .0     .0     .0     .0

                                                                    Bottom
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve      F10=Restart
F11=Transition data   F12=Cancel   F14=Subsystems   F24=More keys
```

2. Determine the system ASP storage amount from the value in the *System* field (1803M), and multiply that value by the value in the *% used* field. The *% uses* is the storage currently used.

```
WRKSYSSTS---System ASP Storage  =    1803M
           Storage being used   =    x60.63

                                     ------
                                     109,316
```

3. Divide the value calculated in step 2 by 100.

```
Divide 109,316 by 100           =    1093.16

Round this value to             =    1093
```

4. Look at the Work with Disk Status (WRKDSKSTS command) display.

```
                              Work with Disk Status              RCH38342
                                                              02/26/90  14:30:54
Elapsed time:   00:00:00

                Size     %   I/O   Request  Read  Write   Read  Write     %
Unit  Type      (M)   Used   Rqs  Size (K)  Rqs    Rqs     (K)   (K)   Busy
  1   9332      200   98.2   .0       .0     .0     .0      .0    .0      0
  2   9332      200   57.4   .0       .0     .0     .0      .0    .0      0
  3   9332      200   58.6   .0       .0     .0     .0      .0    .0      0
  4   9332      200   58.0   .0       .0     .0     .0      .0    .0      0
  5   9332      200   59.0   .0       .0     .0     .0      .0    .0      0
  6   9332      200   57.0   .0       .0     .0     .0      .0    .0      0
  7   9332      200   59.8   .0       .0     .0     .0      .0    .0      0
  8   9332      200   48.7   .0       .0     .0     .0      .0    .0      0
  9   9332      200   48.6   .0       .0     .0     .0      .0    .0      0



                                                                      Bottom
Command
===>
F3=Exit    F5=Refresh    F12=Cancel    F24=More keys
```

5. Determine the size of the load source unit (unit 1).
6. Subtract the value of unit 1 from the value in the *Maximum unprotected used* field on the Work with System Status display.

   ```
   Maximum unprotected storage use =   265M
   Size of unit 1                  = - 200M

   Net unprotected storage         =   65M
   ```

7. Subtract the value in step 6 from the value calculated in step 3, since the *% used* value includes the total amount of protected and unprotected storage when checksum protection is not in effect.

   ```
   Total storage used              =   1093M
   Net unprotected storage used        - 65M

   Protected storage used              1028M
   ```

8. Multiply this value by a factor (for example, 1.5) to allow for growth and peak storage needs. You may want to use a larger value, depending on your future storage needs.

   ```
   Protected storage used              1028M
   Buffer for future growth            x1.5

   Protected storage needed            1542M
   ```

Determining the Amount of Protected Storage You Need for Checksum Protection in a User ASP

User ASPs that have checksum protection do not reserve unprotected storage for temporary objects. You only need to calculate the protected storage needed in the ASP.

Use the Display Disk Configuration Capacity option in SST or DST to determine the amount of storage currently used in the user ASP. To allow for growth and peak storage needs, multiply this value by a factor (for example 1.5), depending on your

future storage needs. Use this value when calculating a disk configuration for checksum protection for a user ASP.

Determining the Amount of Unprotected Storage You Need for the System ASP

To determine the amount of space to allocate for unprotected storage, you must first consider how the space is to be used. Unprotected storage is used for:

- Running jobs. The Work with Active Jobs (WRKACTJOB) display shows, at the top of the display, the number of jobs currently active in the system. Each job requires unprotected storage for internal work areas. You can display this value for a job using the Work with Jobs (WRKJOB) command and selecting the Job Run Attributes display. The temporary (unprotected) storage used value appears at the bottom of the Job Run Attributes display, and is shown as kilobytes.

  Sample several jobs to determine the averages for your system. Certain jobs (for example, certain office products) may require a large amount of temporary storage.

- Storing machine data. The unit 1 is used mainly for licensed internal code and the system data. The entire size of the unit 1 is included in the unprotected used values shown by WRKSYSSTS when checksum protection is not active.

Your *unprotected storage* allocation should be the total of:

- The amount of unprotected storage already used immediately after you IPL your system, and before other users sign on.
- The average amount of temporary (unprotected) storage used per job multiplied by the total number of jobs you expect to support.

To verify that the value you calculate is appropriate for your needs, use the Work with System Status (WRKSYSSTS) command to show the current amount of unprotected storage used and the maximum amount of unprotected storage used since the last IPL of the system.

Calculating a Disk Configuration for Checksum Protection

Use the SST/DST Calculate Checksum Configuration display to calculate the amount of protected and unprotected storage capacity that would be provided if checksum protection were in effect for a particular disk configuration. You can calculate the storage capacity of your current system configuration or the storage capacity of additional disk units that you may add later by using a different configuration.

To calculate disk storage capacity, perform the following steps:

1. After you have determined your protected and unprotected storage needs (see "Determining the Amount of Protected Storage You Need for Checksum Protection in the System ASP" on page 9-2 and "Determining the Amount of Unprotected Storage You Need for the System ASP"), use the option 5 (Calculate checksum configuration display) on the Work with Checksum Protection display to display the capacities of protected and unprotected storage that would be provided by the disk units currently in your system ASP, or a set of disk units of your choosing.
2. This step applies only to the system ASP.

Assume that:

- You need 560MB of storage allocated for unprotected storage.
- You have two 2800 Model 001 disk units, each with 2 storage units configured to ASP 1.
- You have four other disk units, each with 2 storage units.

Calculate the unprotected storage per unit value needed as input to the Calculate Checksum Configuration function by:

a. Subtracting the capacity of unit 1 (since it is assigned to unprotected storage primarily for system data) from the amount of unprotected storage that you have determined you need.

   Subtract 320MB from the unprotected storage (560MB) to get the adjusted amount of unprotected storage.

```
(unprotected storage) - (unit 1)          = X

                560MB  - 320MB            = 240MB
```

b. Divide the adjusted amount of unprotected storage by the number of storage units that have checksum protection. Do not include the load source unit or any 2800-001 units because they do not have checksum protection.

   Divide 240M (560MB needed less the 320MB for the 2800-001 units) by 8.

```
(X)     divided by (checksum units) = (unprotected storage per unit)

240MB   divided by       8          = 30
```

Note: Storage per unit cannot be more than 100. If your calculated value is more than 100, use 100.

3. Select an option on the Calculate Checksum Configuration display:
   - Existing ASP
   - Existing ASP and user-specified units
   - User-specified units
4. Enter the number of the ASP and the amount of unprotected storage per unit. Enter the information in the remaining fields.
5. Press the Enter key.
6. When the Possible Checksum Configuration display is shown, count the total number of units contained in all checksum sets. If this number is different from the number you divided by in your original calculation (8 in the above example), figure your calculation for the unprotected storage per unit value again, using the new number and repeating step 2 on page 9-4 above.

   This must be done to ensure adequate performance of disk accesses for the temporary objects stored in unprotected storage by evenly spreading the space allocations for unprotected storage across all the units in checksum sets. Otherwise, if space for unprotected storage is allocated on just a few storage units, contention is possible and system performance will suffer.

   If the value shown for protected storage available under that checksum configuration is adequate for your needs, the number of disk units you specified in the calculation should be adequate.

If the value shown for protected storage is less than the value for your protected storage needs, you have to add disk units to the ASP.

Determining the Number of Additional Disk Units You Need: If you determine that you need additional disk units, specify different disk units on the DST Calculate Checksum Configuration display to try different combinations. Use the calculation examples in the previous section to determine the amount of unprotected storage per unit to specify on the display.

For example, assume you have two 9335 disk units and unit 1 configured on your system. You have decided to add an additional 9335 disk unit and want to determine if there will be adequate protected and unprotected storage. Also, assume that you determined you will need 60M of unprotected storage.

To determine the amount of protected and unprotected storage that would be provided by these disk units if checksum protection were in effect:

1. Use the DST Calculate Checksum Configuration display to request a calculation for your base configuration plus the 9335 disk unit to be added. Specify 10 for the unprotected storage per unit value (60M divided by 6 units).
2. You then see the Calculate Storage Configuration display for ASP 1. It shows what the total protected and unprotected storage capacity per storage unit would be if checksum protection were in effect for the specified devices. Checksum set assignments for each unit are also shown.

Based on your earlier calculations ("Determining the Amount of Unprotected Storage You Need for the System ASP" on page 9-4) you can determine whether this amount of storage is adequate for your needs. Refer to the disk capacity information in Table 6-1 on page 6-6 to help determine which devices to add to your system.

Starting Checksum Protection for the System ASP

Read the topic entitled "Calculating a Disk Configuration for Checksum Protection" on page 9-4 to verify storage capacity needs before you perform the following.

Task Overview

1. Access DST
2. Add disk units (optional)
3. Start checksum protection
4. Perform an IPL

It is recommended that you save the entire system if you do not have a current backup copy of the system.

Task 1. Access DST

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Task 2. Add Disk Units to the System ASP (Optional)

If you have units to add to the system ASP, do the following. Otherwise continue with "Task 3. Start Checksum Protection" on page 9-10.

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                      Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                          Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 4 (Add units to existing ASP) on the Work with ASP Configuration display.

```
                       Specify ASPs to Add Units to

Specify the ASP to add each unit to.

Specify  Serial
  ASP    Number      Type  Model  Address
   __    10-00A7503   9332    400    0010-0100FFFF
   __    10-00A7503   9332    400    0010-0101FFFF
   __    10-00A3651   9332    400    0010-0400FFFF
   __    10-00A3651   9332    400    0010-0401FFFF
```

5. Type the number of the ASP in the *Specify ASP* column to select the disk units to place in the specified ASP and press the Enter key.

   The Confirm Add Units display shows what the entire system configuration will be when you add the units. Use the serial number of the disk unit to verify that you are selecting the correct disk units to add to the ASP.

```
                              Confirm Add Units

Add will take several minutes for each unit.  The system will
have the displayed protection after the unit(s) are added.

Press F9=Capacity Information to display the resulting capacity.
Press Enter to confirm your choice for 1=Add units.
Press F12=Cancel to return and change your choice.


           Serial
 ASP  Unit Number      Type  Model  Address         Protection       CSS
  2                                                 Unprotected
        9  10-00A3651  9332   400   0010-0400FFFF   Unprotected
       10  10-00A3651  9332   400   0010-0401FFFF   Unprotected
```

6. If you are satisfied with the configuration, press the Enter key to add the disk units to the ASP.

Adding units takes several minutes. The time it takes depends on the size of each unit being added and the ability of the system to add more than one unit at the same time.

Task 3. Start Checksum Protection

1. Return to the Work with Disk Unit display by pressing F3 (Exit) two times.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                       Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 4 (Work with checksum protection) on the Work with Disk Configuration display.

```
                       Work with Checksum Protection

Select one of the following:

     1. Display checksum configuration
     2. Start checksum protection
     3. Stop checksum protection
     4. Change unprotected storage
     5. Calculate checksum configuration
```

4. Select option 2 (Start checksum protection) on the Work with Checksum Protection display to start checksum protection for the ASP.

```
                    Select ASP for Start Checksum Protection

Type option, press Enter.
  1=Select

Option  ASP    Protection
  _       1    Unprotected
  _       2    Unprotected
```

5. Select the ASP by typing a 1 in the *Option* and pressing the Enter key.

   The Specify Unprotected Storage display is shown.

```
                          Specify Unprotected Storage

ASP number. . . . . . . . . . . .:

      Serial
Unit  Number       Type  Model   Address
 1    10-00A5307   9332   200    0010-0100FFFF
 2    10-00A4937   9332   200    0010-0101FFFF
 5    10-00A3651   9332   400    0010-0400FFFF
 6    10-00A3651   9332   400    0010-0401FFFF




Type Choice, Press Enter.

   Unprotected storage per unit: . . .  ___  5-150 Megabytes

F3=Exit          F12=Cancel
```

6. Specify the desired value for the unprotected storage per unit and press the Enter key.

```
                    Confirm Start of Checksum Protection

A subsequent stop checksum protection request on this ASP
before IPL will ignore this start checksum request.

Press Enter to confirm your choice for 1=Start checksum.
Press F12=Cancel to return and change your choice.

ASP . . . . . . . . . . :  __
```

7. Press the Enter key to confirm your choice of 1=Start Checksum.

   The Display Checksum Configuration display is shown. Press the Enter key to continue.

8. Press F3 (Exit) until you return to the Use Dedicated Service Tools (DST) menu.

Task 4. Perform an IPL

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

1. Select option 1 (Perform an IPL) on the Use Dedicated Service tools menu. The IPL or Install the System menu is shown.
2. Select option 1 (Perform an IPL) on the IPL or Install the System Menu.

This completes the steps to start checksum protection. The system will move data from sectors on each storage unit to other sectors in the ASP. The move of data is done to allow space for checksum data on each storage unit. This process can take up to 10 hours for the system ASP.

Starting Checksum Protection for a User ASP

Read the topic "Calculating a Disk Configuration for Checksum Protection" on page 9-4 to verify the storage capacity needs before you perform the following.

Task Overview

1. Access DST
2. Add units to the ASP (optional)
3. Start checksum protection
4. Perform an IPL

If you do not have a current backup copy of the system, it is recommended that you save the security data. Use the Save Security Data (SAVSECDTA) command to save the security data.

If you do not have a current backup copy of the system, it is recommended that you save the objects in the ASP.

Task 1. Access DST

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                  Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

Task 2. Add Units to the ASP (Optional)

If you have units to add to the ASP, continue with the following. Otherwise, continue with task 3.

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                         Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                          Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 4 (Add units to existing ASP) on the Work with ASP Configuration display.

```
                       Specify ASPs to Add Units to

Specify the ASP to add each unit to.

Specify  Serial
  ASP    Number      Type  Model  Address
   __    10-00A7503   9332    400    0010-0100FFFF
   __    10-00A7503   9332    400    0010-0101FFFF
   __    10-00A3651   9332    400    0010-0400FFFF
   __    10-00A3651   9332    400    0010-0401FFFF
```

5. Type the number of the ASP in the *Specify ASP* column to select the disk units to place in the specified ASP and press the Enter key.

   The Confirm Add Units display shows what the entire system configuration will be when you add the units. Use the serial number of the disk unit to verify that you are selecting the correct disk units to add to the ASP.

```
                              Confirm Add Units

Add will take several minutes for each unit.  The system will
have the displayed protection after the unit(s) are added.

Press F9=Capacity Information to display the resulting capacity.
Press Enter to confirm your choice for 1=Add units.
Press F12=Cancel to return and change your choice.


           Serial
ASP  Unit  Number      Type  Model  Address         Protection       CSS
 2                                                 Unprotected
       9   10-00A3651  9332   400   0010-0400FFFF   Unprotected
      10   10-00A3651  9332   400   0010-0401FFFF   Unprotected
```

6. If you are satisfied with the configuration, press the Enter key to add the disk units to the ASP.

Adding units takes several minutes. The time it takes depends on the size of each unit being added and the ability of the system to add more than one unit at the same time.

Task 3. Start Checksum Protection

1. Return to the Work with Disk Unit display by pressing F3 (Exit) two times.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                       Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 4 (Work with checksum protection) on the Work with Disk Configuration display.

```
                       Work with Checksum Protection

Select one of the following:

     1. Display checksum configuration
     2. Start checksum protection
     3. Stop checksum protection
     4. Change unprotected storage
     5. Calculate checksum configuration
```

4. Select option 2 (Start checksum protection) on the Work with Checksum Protection display to start checksum protection for the ASP.

```
                    Select ASP for Start Checksum Protection

Type option, press Enter.
  1=Select

Option  ASP    Protection
  _       1    Unprotected
  _       2    Unprotected
```

5. Select the ASP by typing a 1 in the *Option* and pressing the Enter key.

```
                    Confirm Start of Checksum Protection

A subsequent stop checksum protection request on this ASP
before IPL will ignore this start checksum request.

Press Enter to confirm your choice for 1=Start checksum.
Press F12=Cancel to return and change your choice.

ASP . . . . . . . . . . :  __
```

6. For a user ASP, the confirmation will notify you if any units in the ASP will not be part of a checksum set.
7. Press the Enter key to confirm your choice of 1=Start Checksum.

   The Display Checksum Configuration display is shown Press the Enter key to continue.
8. Press F3 (Exit) until you return to the Use Dedicated Service Tools (DST) menu.

Task 4. Perform an IPL

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit            F12=Cancel
```

1. Select option 1 (Perform an IPL) on the Use Dedicated Service tools menu. The IPL or Install the System menu is shown.
2. Select option 1 (Perform an IPL) on the IPL or Install the System Menu.
3. This completes the steps to start checksum protection. The system will move data from sectors on each storage unit to other sectors in the ASP. The move of data is done to allow space for checksum data on each storage unit.

Adding Storage Units to an ASP While Checksum Is in Effect

You can add units to an ASP while it is under checksum protection to increase the capacity:

1. Use DST Calculate Checksum Configuration display to verify that the new disk will provide the desired capacity.
2. Your service representative attaches the new disk.
3. After the disk is attached, the units for it appear as nonconfigured units on the DST Display Non-configured Units display.
4. Use the DST Add Units to Existing ASP display to select the new units to add to the system ASP.

   The new units are automatically added to the existing checksum sets or a new checksum set is created as part of the add operation.
5. Use the DST Display Checksum Configuration display to verify the new configuration.

Moving a Storage Unit Not in a Checksum Set from the System ASP to a User ASP

Units (other than unit 1) in the system ASP that are not in a checksum set when checksum protection is started do not contain permanent user data. They can be moved without causing destruction of user data in the system ASP.

Warning: When considering the removal of units from the system ASP (ASP 1), remember to leave at least two 2800-001 storage units in the system ASP for the 9406 model D70 and D80 system units that contain 224MB or more of main storage.

To move a unit from the system ASP, perform the following steps:

1. Use the DST Display Checksum Configuration display to determine which units are not in checksum sets. If any units listed under the *Checksum set number* column are marked with an asterisk, they can be moved into a user ASP (with the exception of the unit 1, which must remain in the system ASP).

   Carefully mark down the identification of the units to be moved.

2. Use the DST Move Unit from One ASP to Another display to move the unit from the system ASP to the desired user ASP.
3. Use the DST Display Disk Configuration displays and the Display Checksum Configuration displays to verify the configuration changes.

Replacing One Disk Unit in a Checksum Set with Another Unit

You can take advantage of checksum protection's ability to rebuild the data on a unit in a checksum set when replacing one unit in a checksum set with another unit. For example, when a disk is experiencing intermittent errors that seem to indicate a hard failure might occur, you may want to replace the unit. Your service representative will assist you in analyzing the problem and deciding if the disk should be replaced. For more information about replacing the unit, see "Replacing a Failed Disk Unit in the System ASP" on page 9-23.

Changing the Amount of Unprotected Storage in the System ASP

Assume your current unprotected storage amount is 10MB per unit, and you want to decrease it to 9MB. To do this use the DST Change Unprotected Storage for ASP1 option and specify the new value of 9 for the unprotected storage per unit value.

- Decreasing this value will cause some additional processing during the next IPL.
- Increasing this value will take much longer because checksum protection will be stopped and then started again (see "Handling Unprotected Storage Overflows" on page 9-20).

Handling Unprotected Storage Overflows

If you underestimate the amount of unprotected storage needed, temporary objects automatically overflow into the protected storage area. A message automatically notifies you when this occurs. Overflow slows system performance because checksum protection is performed on these temporary objects, which ordinarily are stored in the unprotected storage area.

When you IPL the system, unprotected objects are always deleted and this overflow condition is reset. You may want to change your checksum configuration and allocate a larger amount of unprotected storage space if this occurs frequently.

You may also consider installing additional devices to give you more unprotected storage. For each unit added, you get the amount of unprotected storage currently specified for the unprotected storage per unit value you specified for the checksum configuration.

1. Select option 4 (Change unprotected storage) on the Work with Checksum Protection display. Use this option to change the amount of unprotected storage for the system after it has checksum protection. This display automatically shows the configuration of ASP 1 if it has checksum protection.

```
                          Change Unprotected Storage

ASP number  . . . . . . . . . . . . :   1
Current unprotected storage per unit :  22    Megabytes

       Checksum  Serial
Unit   Set No.   Number      Type  Model   Address
 1       *       10-0001234  9332   200    0010-100FFFF
 2       1       10-00A7503  9332   200    0010-101FFFF
 3       1       10-0007204  9332   200    0010-500FFFF
 6       2       57-000451   9335   B01    0020-001FFFF
 8       2       57-000461   9335   B01    0020-100FFFF

* - Indicates that unit is not part of any checksum set

Type choice, press Enter.

    New unprotected storage per unit. :  ___   5-150  Megabytes



F3=Exit         F12=Cancel
```

- *Current unprotected storage per unit.* The amount of unprotected storage that exists now.
- *New unprotected storage per unit.* The amount of unprotected storage you want to enter.

Handling Protected Storage That Has Reached Maximum Storage Capacity

If objects placed in the ASP cause the protected storage area to exceed the threshold value specified for the ASP, a message is sent to the system operator every hour warning that the threshold was reached.

When the actual storage space is exceeded in a user ASP, the data overflows into the system ASP.

When the actual storage space is exceeded in the system ASP, and the protected area reaches its maximum capacity, the system completely stops, resulting in a device error. When this occurs, you must do one of the following after you IPL your system:

- Delete objects from auxiliary storage.
- Reduce the amount of unprotected storage.
- Move user ASP devices to the system ASP.
- Stop checksum protection.
- Configure additional devices to increase the amount of protected storage.

It is possible for protected storage to reach its maximum capacity because temporary (unprotected) objects overflow into protected storage. If this occurs, you can IPL your system (which deletes all temporary objects), freeing protected storage. For information on what to do if this occurs, see "Handling Unprotected Storage Overflows" on page 9-20.

If checksum protection is in effect, the system requires at least 2MB of available protected storage to IPL the system. If protected storage reaches its maximum capacity, and you have less than 2MB of protected storage, you must use DST options to increase the protected storage capacity. You should either add units to the system ASP, which can be allocated to checksum sets, or reduce the amount of unprotected storage per unit. While the system is designed to allow for recovery when protected storage becomes full, there may be instances when you are unable to IPL your system.

Stopping Checksum Protection

You do not need to save or restore the system to stop checksum protection. The units you specified for checksum protection are available for normal use and the storage space for checksum data is freed.

To stop checksum protection, perform these steps:

1. Select option 3 (Stop checksum protection) on the Work with Checksum Protection display.
2. Type a 1 in the *Option* column to select the ASP to stop checksum protection.
3. Press the Enter key. The Confirm Stop of Checksum display is shown.
4. Press the Enter key to confirm your choice of 1=Stop Checksum protection.
5. Use the DST Display ASP Configuration display to confirm your changes.
6. Return to the Use Dedicated Service Tools menu.

7. Select option 1 (Perform an IPL) on the Use Dedicated Service tools menu. The IPL or Install the System menu is shown.
8. Select option 1 (Perform an IPL) on the IPL or Install the System Menu.

Checksum Recovery Actions

Recovering from disk unit media failures involves steps the service representative performs as well as steps you perform. When using checksum recovery consider the following.

Checksum Recovery Actions Performed by the Service Representative

The recovery actions performed by the service representative are provided here so you know what happens before you recover from a disk unit media failure. This discussion applies to a disk failure in an ASP where checksum protection is in effect.

If a disk unit in your system fails, your service representative determines whether data loss has occurred. For the purpose of the checksum recovery actions discussed below, any case of data loss is considered a disk unit media failure.

Following are the steps your service representative follows when helping you recover from a disk unit media failure:

1. The service representative attempts to use the DST Save Disk Unit Data function to save the data on the failed units to tape.
   - If all data from the failed unit is saved, the service representative can restore it to the replacement unit. Further recovery steps are not needed beyond those required for other cases (such as power failures or other disk failures) where machine processing ends abnormally without disk media damage.
   - If some of the data is saved and the unit that failed is not unit 1, the chances of partial data loss are reduced and overall recovery time for the replacement unit is reduced because checksum protection will not have to rebuild any data that has been successfully copied.
   - If all data is not saved and the unit that failed is unit 1, your service representative will go through a procedure to rebuild the data stored on unit 1 to its replacement unit rather than restoring the saved data.
2. The service representative attaches the replacement disk unit to the system. If the disk that failed contained unit 1, it will be attached so that it is addressed correctly to indicate it contains unit 1. Otherwise, no specific address is required for the new disk.
3. The following occurs if the disk that failed contained unit 1:
   a. The service representative performs an IPL of the system from your last SAVSYS or SAVSTG tapes and uses the stand-alone Licensed Internal Code install (option 24) utility to format the new unit 1 and copy the system Licensed Internal Code from tape to the new unit 1.

      Other units in the system are not be cleared by this step. However, they are considered nonconfigured at this point. Installing the Licensed Internal Code results in a default disk configuration of just unit 1 being configured into the system.

b. The service representative uses the DST Recover Configuration function to request the system to recover the disk configuration that was in effect prior to the failure by reading information stored on the other disks in the system. This disk configuration information is placed on unit 1 as a result.

   This step is necessary for the system to reestablish which disks were in the system ASP and which were in the user ASPs, and whether or not checksum or mirrored protection was in effect for the ASP.

Certain values that are normally stored on unit 1 cannot be rebuilt and will be lost. Most of these will be put back into effect automatically during IPL processing performed by the OS/400 program. You may have to set other values correctly during the recovery actions you perform.

4. The following occurs if the disk unit that failed contained a unit other than unit 1 and some of the data for the unit was saved to tape:
   a. The service representative uses the DST Restore Disk Unit Data function to restore the saved data onto the replacement unit.
5. The following occurs if the disk unit that failed contained a unit other than unit 1 and none of the data for the unit was saved to tape:
   a. The service representative uses the DST Replace Disk function to indicate to the system that the old (failed) unit is being replaced by the newly attached unit. The newly attached unit is formatted at this point.

When you continue the recovery actions by selecting to IPL the operating system, additional disk recovery processing may be performed, if necessary, to go through checksum recovery processing to rebuild the data previously contained on the failed units.

Replacing a Failed Disk Unit in the System ASP

After the service representative has replaced the disk unit, follow these steps to recover from replacement of failed disk units in the ASP when checksum protection is in effect:

If the Units That Failed Did Not Include Unit 1:

1. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools menu.
2. Select option 1 (Perform an IPL) on the IPL or Install the System menu

   The system will attempt to rebuild the contents of the disk units that were replaced and are members of a checksum set. The IPL may take several hours, but you will not need to reload your system, data, or other objects residing in the ASP. Unit 1 is never a member of a checksum set and is not operated on by this rebuild processing.

If the Units That Failed Did Include Unit 1:

1. If the units that failed included unit 1 and you had previously changed the DST passwords from their defaults to values of your own choosing, use the DST security options to set your password values back into effect. For more information about DST passwords, see *Security Reference*.

   Also, if you had previously changed the system value for the date format from the default, the default format will be in effect until the OS/400 program com-

pletes IPL processing. IPL processing sets the format back to the value that was in effect when you saved your system. It is assumed the Licensed Internal Code has been installed.

a. Perform an abbreviated install of the operating system to reinstall certain data needed by IPL onto unit 1. For more information on install, refer to *Basic Backup and Recovery Guide*. To perform the abbreviated install, follow these steps:

a. Select the option to install the operating system.

b. On the Install Operating System display, select the option named *Change Installing Options*.

c. On the Installing Options display, select the option named *Do not Restore Programs or Language Objects*.

On this display you normally want to specify *Keep* for the option named *Clear Job and Output Queues*.

The system then performs a relatively short amount of processing to load the necessary data from tape onto the system. After this processing is complete, the system automatically continues to the step of doing an IPL of the operating system.

The system attempts to rebuild the contents of the disk units that were replaced and are members of a checksum set. The IPL may take several hours, but you do not need to reload your system, data, or other objects residing in the system ASP. Unit 1 is never a member of a checksum set and is not operated on by this rebuild processing.

2. If the units that failed included unit 1, a message may appear during IPL processing. This message asks you to enter the configuration information again because some hardware configuration information has been lost. To do this, use the Work with Hardware Products (WRKHDWPRD) command.

3. If the units that failed included unit 1, a message (CPI0916) may appear during IPL processing. This message asks you to set the correct values for these attributes on your system because the network attribute information has been lost. Follow the instructions provided in that message to set the correct network attributes at this point.

4. If the units that failed included unit 1 and if you use the AS/400 Cryptographic Support (5728-CR1) licensed program on your system, you must reenter the master cryptographic key. To do this you use the Set Master Key (SETMSTK) command.

Recovering From a Disk Unit Media Failure in a User ASP That Has Checksum Protection

After the service representative has replaced the disk unit, perform the following steps to recover from a replacement of a failed disk unit in the user ASP when checksum is in effect.

1. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools (DST) menu.

2. Select option 1 (Perform an IPL) on the IPL or Install the System menu.

The system attempts to rebuild the contents of the disk unit that was replaced and is a member of a checksum set. The IPL may take several hours, but you

do not need to restore the system, data, or other objects residing in the user ASP.

Estimating Checksum Recovery Time

The number of storage units in the checksum set in which the disk unit failure occurs has a direct bearing on how long it takes to reconstruct the data on the lost unit. To estimate the length of time to reconstruct data:

1. Determine the number of units in the checksum set that contained the unit that failed.
2. Multiply this number by the per-unit rebuild time:

Disk Unit	Recovery Time in Minutes
9332	5 to 10
9335	10 to 15
9336 010	10 to 15
9336 020	20 to 30
6100	12 to 20

   For example, rebuilding a 9335 unit in a 3-unit checksum set should take from 30 to 45 minutes (3 units multiplied by 10 to 15 minutes for each unit).

If a 9332 model 400 or 600, or a 9335 model B01 disk unit fails and is replaced, it usually means that a unit in two different checksum sets must be rebuilt (because these disk units have two storage units). To estimate the time to reconstruct the data on an entire drive (with two storage units), perform the calculation described above and then double the time.

The time estimates given in these examples are just approximate. The actual times experienced on your system could vary significantly, depending on the particular characteristics of your disk configuration.

These time estimates do not include the time for other recovery processing, such as rebuilding access paths, which may be needed when the system ends abnormally.

Part 5. Device Parity Protection

Chapter 10. Device Parity Protection and Mixed ASP Support

This chapter provides information about disk subsystems with device parity protection and about mixing these units in an auxiliary storage pool (ASP) that has mirrored protection.

Differences between V2R2 with Software Feature 1982 and V2R3

- V2R2 with Software Feature 1982 installed, or V2R3, is required to support the 9337 2XX models.
- V2R2 with Software Feature 1982 installed, or V2R3, is required to support mixing device parity protection disk units within mirrored ASPs.
- V2R3 is required in order to use the Display Device Parity Status function under SST or DST.
- V2R3 provides a new Dedicated Service Tools option to change the data-protection mode of a 9337-2xx without having to move the data off the disk unit or perform a restore operation, if you have enough storage.
- V2R2 with Software Feature 1982 installed, or V2R3, allows a 9337-0xx to be upgraded to a 9337-2xx in base mode without having to perform a restore operation.
- V2R2 with Software Feature 1982 installed, or V2R3, allows a 9337-1xx to be upgraded to a 9337-2xx in high-availability mode without having to perform a restore operation.

Device Parity Protection

Device parity protection is a function that protects data from being lost because of a disk unit failure or because of damage to a disk. Logically, the implementation of device parity protection is similar to checksum protection provided by the system software. However, device parity protection is built into the disk array subsystem.

Unlike system checksum protection, device parity protection cannot be started or stopped solely by selecting an option on a system menu. Device parity protection is built into the high-availability models of the IBM 9337 Disk Array Subsystems.

IBM Disk Array Subsystems with Device Parity Protection

The disk array subsystems supplied by IBM* enhance the selection of recovery options available on the AS/400* system. This method of protection is based on the Redundant Array of Independent Disks (RAID) specifications published by the University of California in 1987. The high-availability models with device parity protection use the RAID-5 data-redundancy technology to protect the data.

The 9337-1xx models are referred to as the high-availability models of the 9337 Disk Array Subsystems. These models contain an implementation of a RAID-5 technology. For the high-availability models, one-quarter of the capacity of the first four 9337-1xx disk units is used for the checksum data and is unavailable for user data. All of the space on disk units 5 through 7 is available for data. Therefore, 25% of the space is used for checksum data on a subsystem configured with four disk units. Fourteen percent of the space is used for checksum data on a subsystem configured with seven disk units.

The 9337-2xx models can be set to high-availability mode. These models contain an implementation of a RAID-5 technology. For the high-availability models, one-quarter of the capacity of the first four 9338-2xx disk units is used for the checksum data and is unavailable for user data. All of the space on disk units 5 through 8 is available for data. Therefore, 25% of the space is used for checksum data on a subsystem configured with four disk units. Twelve and one-half percent of the space is used for checksum data on a subsystem configured with eight disk units.

For the high-availability 9337-1xx models, one additional disk unit (referred to as the write-assist device) is included in the disk array subsystem. This disk unit is not available to store user data. This disk unit is used to improve performance of interactive write scenarios. More information on

the write-assist device is found in the topic "Write-Assist Device (WAD)" on page 10-4.

For the 9337-2xx models, a write cache is included in the disk array subsystem. This write cache is used to improve performance of interactive write scenarios. More information on the write cache is found in the topic, "Write Cache" on page 10-4.

The IBM 9337 Disk Array Subsystems come in several models. The disk units in each model must all be of the same type. Table 10-1 shows the models and capacity for each model. In the table, *Unit* refers to a disk unit.

Table 10-1 (Page 1 of 2). Models and Capacity for the 9337 Disk Array Subsystems

9337 Model	MB/Unit	MB/Number of Units						
		2	3	4	5	6	7	8
010 (base)[1]	542	1084	1626	2168	2710	3252	3794	N/A
015 (base)[1,2]	542	1084	1626	2168	2710	3252	3794	N/A
020 (base)[1]	970	1940	2910	3880	4850	5820	6790	N/A
025 (base)[1,2]	970	1940	2910	3880	4850	5820	6790	N/A
040 (base)[3]	1967	N/A	N/A	7868	9835	11802	13769	N/A
110 (HA)[4]	406.5/542[5]	N/A	N/A	1626	2168	2710	3252	6
115 (HA)[2,4]	406.5/542[5]	N/A	N/A	1626	2168	2710	3252	6
120 (HA)[4]	727.5/970[5]	N/A	N/A	2910	3880	4850	5820	6
125 (HA)[2,4]	727.5/970[5]	N/A	N/A	2910	3880	4850	5820	6
140 (HA)[4]	1475/1967[5]	N/A	N/A	5901	7867	9834	11801	6
210 (base)[7]	542	1084	1626	2168	2710	3252	3794	4336
210 (HA)[7,8]	406.5/542[5]	N/A	N/A	1626	2168	2710	3252	3794
215 (base)[2,7]	542	1084	1626	2168	2710	3252	3794	4336
215 (HA)[2,7,8]	406.5/542[5]	N/A	N/A	1626	2168	2710	3252	3794
220 (base)[7]	970	1940	2910	3880	4850	5820	6794	7760
220 (HA)[7,8]	727.5/970[5]	N/A	N/A	2910	3880	4850	5820	6794
225 (base)[2,7]	970	1940	2910	3880	4850	5820	6794	7760
225 (HA)[2,7,8]	727.5/970[5]	N/A	N/A	2910	3880	4850	5820	6794
240 (base)[7]	1967	N/A	N/A	7868	9835	11802	13769	15736

Table 10-1 (Page 2 of 2). Models and Capacity for the 9337 Disk Array Subsystems

9337 Model	MB/Unit	MB/Number of Units						
		2	3	4	5	6	7	8
240 (HA)[7]	1475/1967[5]	N/A	N/A	5895	7868	9835	11802	13769

Notes:

1 This base model comes with two disk units without device parity protection as the base configuration. This model can be upgraded to a 1xx model that uses device parity protection.

2 This model reports to the system as an equivalent-capacity 0x0, 1x0, or 2x0 model.

3 This model comes with four disk units without device parity protection as the base configuration. It can be upgraded to a 140 model that uses high-availability protection.

4 This model comes with a minimum of four disk drives and parity protection.

5 Twenty-five percent of the first four disk units for this model is reserved for parity information and is unavailable for user data. Additional units of the same type can be added, and all storage on those units will be available for user data.

6 This unit is called the write-assist device (WAD). The WAD is an integral part of the 9337-1xx models. This unit is used by the 9337 and is not available for user data.

7 All 2xx models have the write cache feature.

8 This model comes with two disk units as the base configuration. It must have a minimum of two feature disk units in order to support device parity protection.

Device Parity Protection Considerations: The following are some things to remember about device parity protection on the AS/400 system.

- Device parity protection or system checksum protection requires one I/O operation for every read operation, and four I/O operations (two reads and two writes) for a write operation. The I/O operations are no different from checksum protection or any RAID-5 technique implementation. The number of I/O operations is not unique to the 9337 high-availability models with device parity protection.
- Different disk unit types and models can be configured on the system. Some disk subsystems require a separate disk I/O processor. However, you cannot configure different disk unit types within the same disk array subsystem.
- Some of the high-availability models have an additional disk unit included in the disk array subsystem. This disk unit does not show on the displays when using WRKDSKSTS, STRSST, or the DST menus. See "Write-Assist Device (WAD)" on page 10-4 for more information about this disk unit.
- The 9337-2xx models have a write cache. To ensure data integrity, the system is not notified that a write operation is complete until a copy of the data is stored in the controller memory and in the write cache.
- Concurrent maintenance can occur on the failed disk unit in the disk array subsystem with device parity protection. The system continues to run during a single disk unit failure and during the synchronization of data after the failed disk unit is replaced. However, the system is not protected during this period. There may be a performance impact to read and write operations. See "Disk Failure in a Device Parity Protection Configuration" on page 10-6 for more information about disk unit failures.
- In some situations, if more than one disk unit fails in a disk array subsystem with device parity protection, it is possible to recover without loss of customer data. In other situations, the system may not be usable. The data must then be restored from the backup media. Because of the write penalty, restoring data to an ASP that has disk units with device parity protection may take longer than an ASP that contains only unprotected disk units.

- Device parity protection does not protect against system outages caused by failures in other disk-related hardware (for example, a disk controller, a disk I/O processor, or a system bus).
- Units that have device parity protection are prevented from becoming part of an ASP that has checksum protection.

Elements of a 9337 Disk Array Subsystem with Device Parity Protection:

The following diagram illustrates the elements of a disk array subsystem with device parity protection.

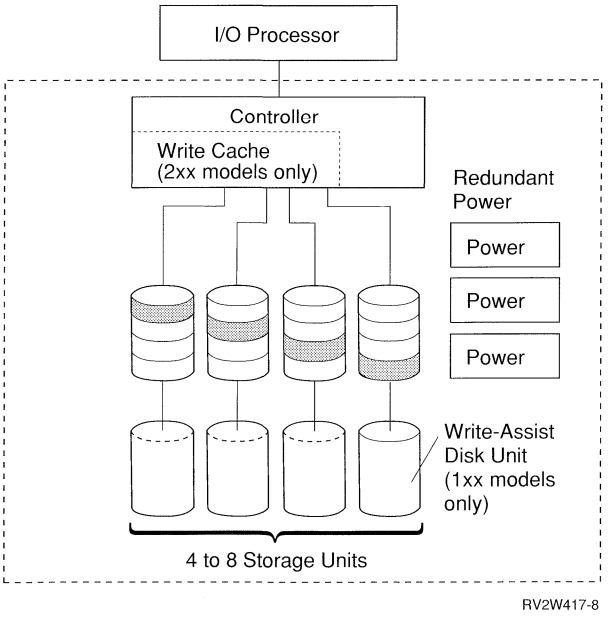


Figure 10-1. Elements of the 9337 Disk Array Subsystem

Write Cache

The 9337-2xx models contain a write cache that is integrated into the controller card. The write cache is implemented in two different areas of the controller:

- The primary cache is part of the controller memory. This memory is very fast and operates without battery backup.
- The secondary cache is a separate component on the controller card. This cache is nonvolatile (it has a battery backup) and is removable. The removable cache provides improved data integrity because the cache (and data) can be moved to a new controller card if a failure occurs in the controller card.

The combination of these two caches provides greater data integrity and improved performance. When a write operation is sent from the AS/400, the data is written to both cache areas and then a write-completion message is sent back to the AS/400. Later, the data is written to the disk. The cache provides the faster write capability and ensures data integrity.

Write-Assist Device (WAD)

The write-assist device (WAD) is an additional disk unit that is available only on 9337-1xx models of the disk array subsystems that have device parity protection. This disk unit is used to improve performance of interactive write operations, and to ensure data integrity if a power loss occurs. No parity calculations are required for write operations to the write-assist device.

Notes:

1. Read requests do not use the write-assist device. Read operations to the WAD are always transparent to the system.
2. If the write-assist device fails, no write operations occur to this disk unit. Application programs wait until a write operation on the disk unit (four I/O operations and parity calculations) is successful before continuing.
3. If a disk unit other than the WAD fails in a 9337 Disk Array Subsystem with device parity protection, write operations to the write-assist device are suspended.
4. If device parity protection is suspended, the write-assist device remains in the disk array subsystem. However, the write-assist device is not addressed by the disk controller or the system.

Disk Controller and the Write-Assist Device:

The disk controller for the subsystems with device parity protection performs an important function for write operations. The controller keeps a list of all uncommitted data written to the write-assist device that has not been written to the data disk or the parity disk. This list is used during a power failure on the AS/400 system. See "Utility

Power Failure and the Write-Assist Device" on page 10-5 for more information.

Write Requests and the Write-Assist Device: A write request to the subsystems with device parity protection starts three write operations. Data to be written to the disk units is first stored in the buffer in the disk controller. From this buffer, the data is sent to the write-assist device, the data disk, and the parity disk.

The following occurs for a write request:

1. A write operation to the write-assist device:
   - Data is written to the write-assist device sequentially. No parity calculation is required for a write operation to the write-assist device.
   - Header information is added by the disk controller (such as identifier and disk address). Trailing information is added for the data before it is written to the write-assist device. The use of the header information is described in "Utility Power Failure and the Write-Assist Device."
   - Normally, the write operations to the write-assist device are completed before the write operations to the disk units. The disk controller sends a completion message to storage management that allows the application to continue. The data written on the write-assist device is marked as uncommitted on the disk controller.

   Note: The write operation to the data disk and the parity disk continues in the background until the data is successfully written and is marked as committed in the disk controller.
2. A write operation to the disk unit:
   - For data:
     - Reads the original data.
     - Writes the new data.
   - For parity data:
     - Reads the original parity information.
     - Compares the new data with the original data and the original parity to calculate the new parity.
     - Writes the new parity information.

   The write operation to the data disk is usually completed before the write operation to the parity disk. The write operation to the data disk does not have to wait for the parity calculation. The delay between the writing of new data and the writing of the new parity information is known as *delayed parity*.
3. Data is marked as committed data when it is successfully written to both the data disk unit and the parity disk unit.
4. A completion message is sent to storage management only if the write operation on the write-assist device or the data disk unit has not already sent a message.

The performance for this type of write operation is dependent on disk contention and the time needed to calculate the parity information. This is similar to the implementation of checksum protection provided by the system software.

Utility Power Failure and the Write-Assist Device: The write-assist device also helps in maintaining data integrity if utility power fails to the 9337 Disk Array Subsystem. Two situations can occur during a power failure:

1. The write request is not written to the write-assist device or the disk units in the subsystem with device parity protection.

   The subsystem is not able to complete the write operation and cannot send the completion message to the host. It is the responsibility of the application user to send the write request again. An uninterruptible power supply connected to the AS/400 system can provide protection against this type of situation.
2. The write request is written to the write-assist device and is marked as uncommitted. However, the write operation to the data disk and the parity disk units is not complete.

   The disk array subsystem with device parity protection recovers the data when power is restored.

   As previously explained, the disk controller keeps a list of uncommitted data. From this list, the disk controller gets the addresses of the records on the write-assist device. Uncommitted data is read from the write-assist device and written to the disk units. If a controller fails and the list of uncommitted data is lost, the header record is read from the begin-

ning to the end for the write transactions recorded. The header record shows that a particular write transaction occurred. A successful write operation to the disk units marks the data as committed. This functions like journaling using forward recovery.

Device Parity Protection and Performance

Write-intensive applications can affect performance. For example, batch programs that have many write requests in a short period of time can affect performance.

A single disk unit failure can affect the performance for both read and write operations.

Disk Failure in a Device Parity Protection Configuration: Device parity protection allows the AS/400 system to continue to operate when a single disk unit failure occurs in a 9337 Disk Array Subsystem with device parity protection. The system continues to run in an exposed mode until the repair operation is complete and the data is synchronized. With system checksum protection, a disk unit failure causes the system to stop until the repair operation is completed. The system performs an IPL to synchronize the data.

The write-assist device is suspended when a disk unit failure occurs in a subsystem with device parity protection. If the write-assist device fails, it is not used again until the repair operation is completed. The performance advantage of the write-assist device is lost until the disk unit is repaired.

The subsystems with device parity protection are considered exposed until the synchronization process completes after replacing the failed disk unit. During the time the disk unit is considered exposed, all read operations to the disk array subsystem with device parity protection may require additional I/O operations because:

- A read operation from a failed disk unit has to read all disk units. Read operations from other disk array subsystems with device parity protection are not affected. For more information, see "Read Operations on a Failed Disk Unit" on page 10-7.
- Write operations depend on where the data is to be written. There are three scenarios:
  1. Write operations to a disk unit that does not involve the failed disk unit requires two read operations and two write operations. This is normal operation.
  2. Write operations to a failed disk unit require N-1 read operations (where N equals the number of disk units) and one write operation.
  3. Parity information on the write operation on the failed disk unit requires 1 write operation.

     See "Write Operations on a Failed Disk Unit" on page 10-7 for more information.

  I/O operations during the rebuild (synchronization) process of the failed disk unit may not require additional disk I/O requests. This depends on where the data is read from or written to on the disk unit that is in the synchronization process. For example:

  - A read operation from the disk area that has already been rebuilt requires one read operation.
  - A read operation from the disk area that has not been rebuilt is treated as a read operation on a failed disk unit. See "Read Operations on a Failed Disk Unit" on page 10-7 for more information.
  - A write operation to the disk that has already been rebuilt requires normal read and write operations (two reads and two writes).
  - A write operation to the disk area that has not been rebuilt is treated as a write operation to a failed disk unit. See "Write Operations on a Failed Disk Unit" on page 10-7 for more information.

Note: Read and write operations to a replaced disk unit may affect the rebuild performance for a disk. Every read or write request interrupts the rebuild process to perform the necessary I/O operations.

When the disk unit is considered exposed, the fewer the number of disk units in the 9337 Disk Array Subsystem, the smaller the decrease in performance. The overhead associated with a disk failure in a subsystem can be significant

depending on the number of disk units in the subsystem. If the decrease in performance is not acceptable when using device parity protection, consider using mirrored protection.

Read Operations on a Failed Disk Unit:

To get the data that was contained on a failed disk unit, device parity protection must read each disk unit in the 9337 Disk Array Subsystem that contains the failed disk unit. For subsystems with device parity protection, this means from three to seven total read operations.

A fully configured subsystem with device parity protection can have a significant effect on response time, depending on the workload. The decrease in performance is in effect until the failed unit is repaired or replaced and until the rebuild process is complete.

Note: Because the failure of a single disk unit with device parity protection may contain only a small portion of user data, it is possible that only a few users will be affected by the decrease in performance.

Write Operations on a Failed Disk Unit:

The following examples show what happens to write operations when a single disk unit fails in a 9337 Disk Array Subsystem with device parity protection. Figure 10-2 on page 10-8 is used for all three examples.

Example 1: A write request from the AS/400 system is to DS1, disk unit 3. The write operation does not involve disk unit 5 that has failed. The following occurs:

1. The system reads the original data from disk subsystem DS1, disk unit 3.
2. The system reads the original parity information from PS1, disk unit 1.
3. New parity information is calculated.
4. New data is written to DS1, disk unit 3.
5. New parity information is written to PS1, disk unit 1.

   Write operations are handled as normal write operations to disk units with device parity protection (2 read and 2 write operations).

Example 2: A write operation from the AS/400 system is to DS2, disk unit 5. The write operation detects that disk unit 5 has failed. The following occurs:

1. The original data is lost on DS2, disk unit 5, because of the failure.
2. The original data is created again by reading DS2, disk unit 1; PS2, disk unit 3; and DS2, disk unit 7.
3. New parity information is calculated.
4. New data cannot be written to DS2, disk unit 5, because of the failure.
5. New parity information is written to PS2, disk unit 3.

   Write operations require multiple reads (N-1 reads, where N is the number of disk units) and only one write operation for the new parity information. Data from disk unit 5 will be rebuilt during synchronization after disk unit 5 is replaced.

Example 3: The write request from the AS/400 system is to DS3, disk unit 7. Parity information for DS3, disk unit 7, is on failed disk unit 5. The following occurs:

1. The disk array subsystem with device parity protection detects a disk failure on disk unit 5.
2. Calculating parity information is not required because it cannot write to PS3, disk unit 5. Therefore, there is no requirement to read the original data and the parity information.
3. Data is written to DS3, disk unit 7.

   A write operation requires only one write for the new data. Parity data for PS3 will be rebuilt during synchronization after disk unit 5 is replaced.

Device Parity Protection and Mixed ASP Support

The AS/400 system recognizes the 9337 Disk Array Subsystem the same as any other auxiliary storage device on the system. Table 10-2 on page 10-8 shows the use of these subsystems in unprotected and protected ASPs.

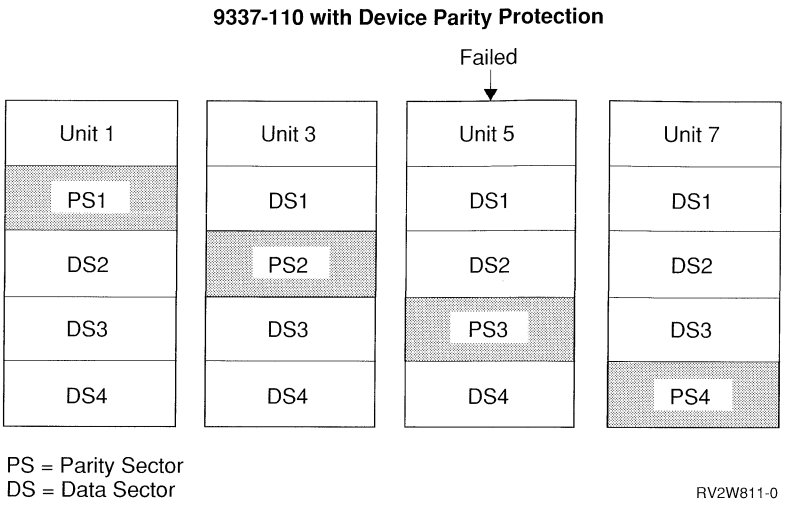


Figure 10-2. Failed Unit in a 9337-110 with Device Parity Protection

Table 10-2. Adding 9337 Disk Units to the AS/400 System

Current Configuration	Standard Models	High-Availability Models
2800, 2801, 9332, 9335, or 9336 in an unprotected system ASP	Yes[1]	Yes[1]
Mirrored protection	Yes[2]	Yes[3]
Checksum protection	Yes[4]	Not allowed
Unprotected user ASP	Yes	Yes[5]

Notes:

1. Disk units without device parity protection in the system ASP remain unprotected unless protected by checksum or mirroring. Disk units with device parity protection are protected by the 9337 Disk Array Subsystem.
2. Disk units without device parity protection are protected by mirrored protection provided by the system software.
3. Disk units with device parity protection are protected by the subsystem. The other units in the ASP are protected by mirrored protection provided by the system software.
4. These disk units become part of checksum sets. Internal not protected.
5. Device parity protection is maintained within the subsystem even if the disk units are divided into separate user ASPs. However, the ASP is considered unprotected.

Device Parity Protection and Mirrored Protection

9337 Disk Array Subsystems without device parity protection can be added to an ASP with mirrored protection. If this is done, the disk units participate in the mirrored protection configuration. The same mirrored protection implementation rules apply to these disk units.

Disk array subsystems with device parity protection do not become part of mirrored protection. All units without device parity protection are mirrored. All disk units with device parity protection are protected by the subsystem. For example, if the system ASP contains only the 2800s or the 2801s, these devices can be mirrored. If you add subsystems with device parity protection to the system ASP, those units are not mirrored. They are protected by device parity protection within the added subsystem.

Warning: When adding 9337-2xx models, make sure the drives are in the desired mode before adding the disk units.

Device Parity Protection and Checksum Protection

9337 Disk Array Subsystems without device parity protection can be added to an ASP with checksum protection. If this is done, the disk units are added to the checksum sets. Subsystems with device parity protection cannot be added to an ASP that has checksum protection.

If the system ASP has checksum protection, you can create a user ASP with subsystems that have device parity protection. If a user ASP has checksum protection, you cannot add subsystems with device parity protection to that user ASP.

Device Parity Protection in an Unprotected ASP

9337 Disk Array Subsystems without device parity protection can be added to an unprotected ASP. If a disk unit failure occurs, the ASP needs to be restored. If you add subsystems with device parity protection to an unprotected ASP, the disk units within the subsystem are protected. However, if another disk unit that does not have device parity protection fails, the system stops and the ASP must be restored.

Device Parity Protection and System Availability

Device parity protection is not a substitute for normal save operations, mirrored protection, or checksum protection. Combining these availability and recovery tools with user ASPs, an uninterruptible power supply, and dual system concepts can provide a high-availability system. Table 10-3 provides an overview of the availability tools that can be used on the AS/400 system to protect against different types of failure.

Table 10-3. Availability Options

What Is Needed?	Checksum Protection	Device Parity Protection	Mirrored Protection	Unprotected User ASPs
Protect from data loss	Yes[1]	Yes	Yes	Yes[1]
Maintain availability	See note[2]	Yes	Yes	No
Aids in disk unit recovery	Yes[1]	Yes	Yes	Yes[1]
System available when disk controller fails	No	No	Yes[3]	No
System available when disk I/O processor fails	No	No	Yes[3]	No
System available when disk I/O bus fails	No	No	Yes[3]	No

Notes:

1. Load source is not protected.
2. Checksum protection improves recoverability. If a disk unit fails, the system stops. When the disk unit is repaired or replaced, the system rebuilds the data.
3. Depends on hardware used, configuration, and level of mirrored protection required.

Planning for Device Parity Protection

The following section provides system protection considerations if you are planning to use device parity protection. It includes configuration examples and performance considerations.

Protected System Using Device Parity Protection

If your goal is to have a system with data loss protection and concurrent maintenance repair, you should plan to use one or more of the following configurations:

- Mirrored protection and device parity protection to protect the system ASP.
- Mirrored protection for the system ASP and device parity protection for user ASPs.
- Mirrored protection and device parity protection to protect the system ASP and user ASPs.

Mirrored Protection and Device Parity Protection to Protect the System ASP

On V2R3 systems and V2R2 systems with Hardware Update Feature 1982 installed, device parity protection can provide data loss protection to the system ASP because the internal disk units (2800 and 2801 disk units) in the system ASP can be protected with mirrored protection. It is possible to combine mirroring the internal disk units (2800 and 2801) with device parity protection in the system ASP.

See the configuration shown in Figure 10-3 on page 10-11. When one of the 9337 disk units with device parity protection fails, the system continues to run. The failed unit can be repaired concurrently. If one of the internal disk units fails, the system continues to run using the operational unit of the mirrored pair.

Mirrored Protection in the System ASP and Device Parity Protection in the User ASPs

Device parity protection should be considered if you have mirrored protection in the system ASP and you are going to create user ASPs. (See Figure 10-4 on page 10-12). The system can tolerate a failure in one of the 9337-110 disk units in a user ASP. The failure can be repaired while the system continues to run.

If a failure occurs in the system ASP, the system continues to run using the operational unit of the mirrored pair.

Mirrored Protection and Device Parity Protection in All ASPs

If you have all ASPs protected with mirrored protection and you want to add units to the existing ASPs, you may consider device parity protection. (See Figure 10-5 on page 10-13). The system can tolerate a failure in one of the disk units with device parity protection. The failed unit can be repaired while the system continues to run.

If a failure occurs on a disk unit that has mirrored protection, the system continues to run using the operational unit of the mirrored pair.

Storage Planning for Device Parity Protection

The space equivalent to one disk unit is used by the disk controller to provide device parity protection.

The minimum number of disk units in a 9337 Disk Array Subsystem with device parity protection is four. The maximum number of disk units in a subsystem with device parity protection is seven or eight, depending on the type.

The parity information is distributed on four units in the subsystem. All disk units in the subsystem are protected by the parity information on the four disk units.

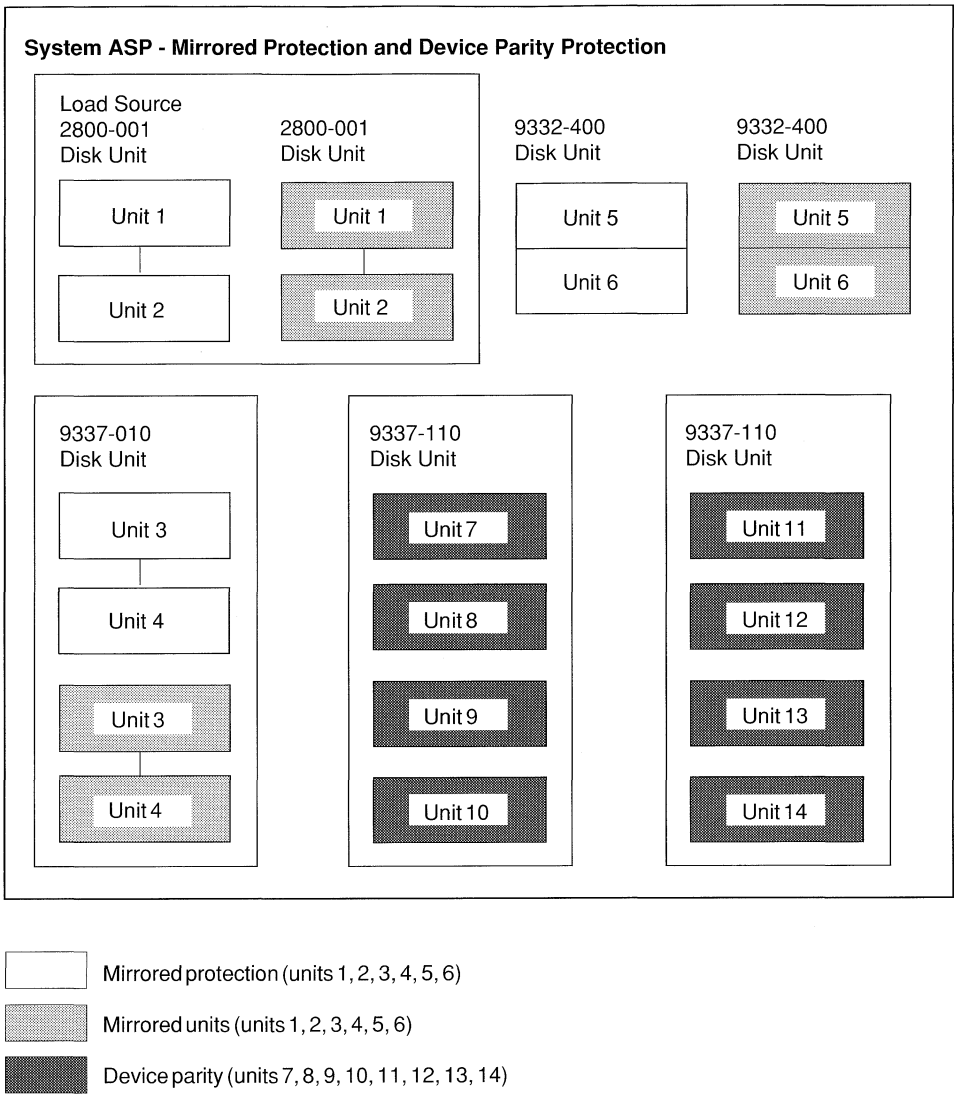


Figure 10-3. Example of Mirrored Protection and Device Parity Protection Used in the System ASP

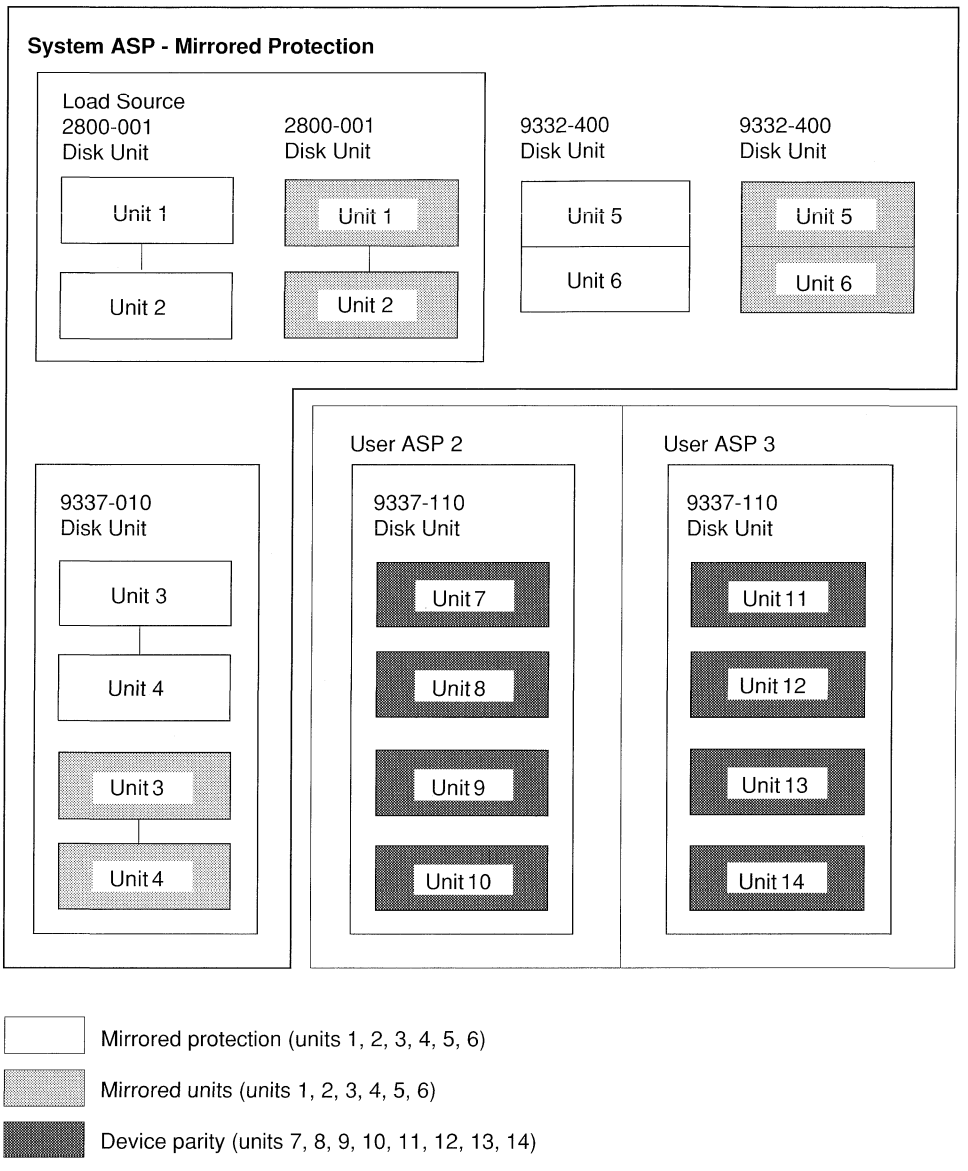


Figure 10-4. Example of a System ASP with Device Parity Protection in the User ASPs

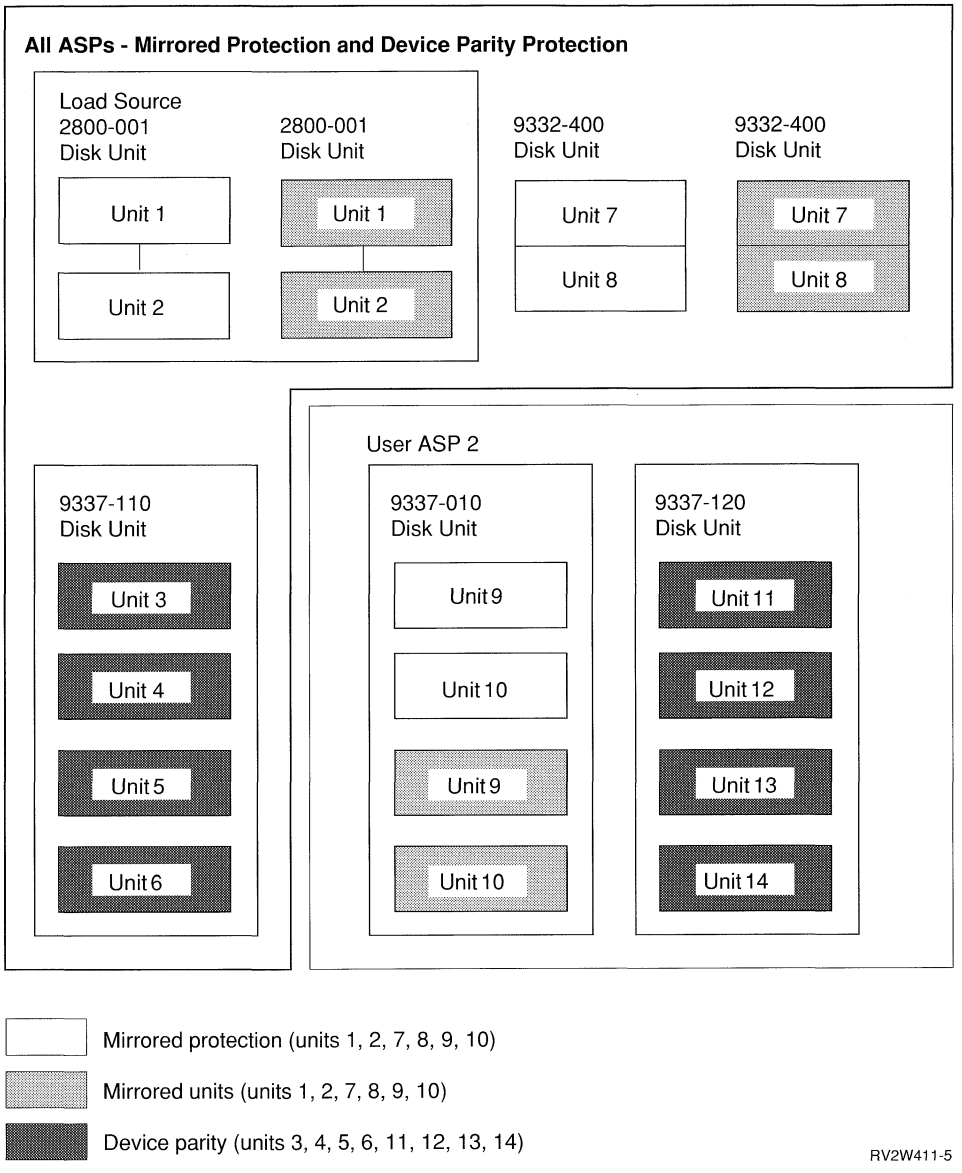


Figure 10-5. Example of Mirrored Protection and Device Parity Protection in All ASPs

Chapter 11. 9337 Disk Array Subsystem Performance

This chapter compares the performance of the 9337 Disk Array Subsystem with other AS/400 9406 disk subsystems. The 9337 Disk Array Subsystem includes the following components:

- The IOP
- The bus that connects the IOP to the disk controller
- The disk controller
- The bus that connects the controller to the disk units
- The disk units

The 9337 Disk Array Subsystem is supported on AS/400 9406 models D, E, and F using the 6500 and 6501 I/O processor. Some of the 9337 Disk Array Subsystems are also supported on the AS/400 9406 model B using the 2611 I/O processor. Feature Code 1982 for Version 2 Release 2 of OS/400 supports mixing 9337 high-availability models in the same ASP that has mirrored devices. In Version 2 Release 3, this support is provided by the base operating system.

The term **standard configuration** refers to a disk configuration that does not use system checksum, the high-availability models of the 9337, or mirrored protection.

9337 Disk Configurations That Are Allowed

Unlike the 9336 disk unit subsystems that allow you to mix 857MB and 471MB disk units in the same 9336 disk unit subsystem, all disk units in one 9337 Disk Array Subsystem must be the same capacity and disk type. This does not restrict you from having different models of the 9337 Disk Array Subsystem on the same system.

The disk units that are used in the 9337 Disk Array Subsystems have improved read-ahead buffers that can provide performance advantages. Each of the disk units has a 256KB buffer or 512KB buffer. (The x40 and xx5 models have 512KB.) The buffer is allocated into multiple segments that are larger than 32KB each. Read-ahead data from recent input/output operations are kept in these buffer segments. Depending on the data access patterns, it is possible that the data is already contained in a buffer segment, and that no physical access to the disk unit is required.

Depending on your data access patterns, these buffers can significantly improve performance. Analysis of several specific customer workloads indicates that 10% to 30% of the disk read operations for interactive transactions are already contained in the read-ahead buffer. For batch jobs, 25% to 45% of the disk read operations are already in the read-ahead buffer.

The Start Performance Monitor (STRPFRMON) command captures additional performance data (buffer accesses, and so on) for the 9337 Disk Array Subsystem. This data is available in the QAPMDISK performance data file and is documented in Appendix A of the *Work Management Guide*.

How Fast Are the Disk Subsystems?

Figure 11-1 on page 11-2 compares the service times for the disk unit subsystem offerings for the AS/400 9406 system unit. The read and write operations being performed are 3.5KB transfer size. Seventy percent are reads, and 30% are writes. Eighty percent of the read and write operations require a seek over 1/3 of the disk surface, while 20% require no seek. Data shown is based on typical interactive disk I/O operation, which is not representative of a specific customer environment. Results in other environments may vary significantly.

Observations

- The 9337-2xx models are up to 40% faster than the equivalent 9337-0xx models. They are faster because of a combination of factors:
  - Faster 6501 IOP that contains a RISC processor
  - Faster 9337 RISC processor in the 9337
  - Write cache in the 9337
  - Faster buses between the IOP and 9337 and between the 9337 and the disk units

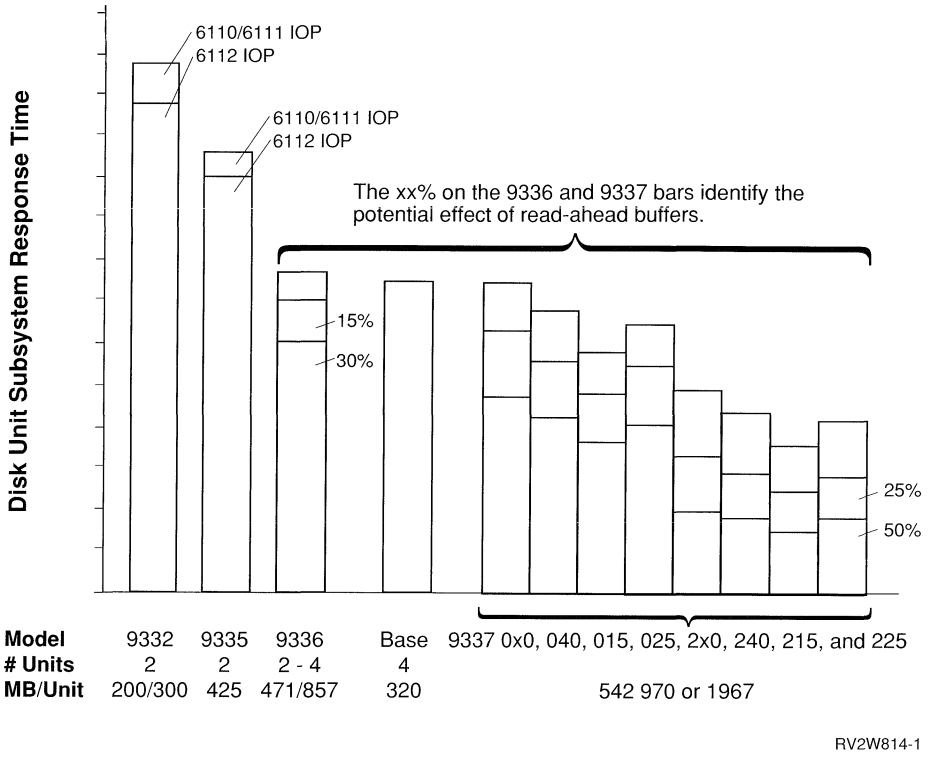


Figure 11-1. 9406 Disk Subsystem Interactive Performance

- The 9337-x25 models are up to 20% faster than those in the 9337-x20 models. The faster disk units is the factor that allows improved performance.
- The 9337-x15 disk units are faster than the 9337-x25 disk units. The x15 disk units have a faster seek time than the x25 disk units.
- Because of faster disk units, the 9337-x40 models are 10% to 15% faster than those in the other 9337-xx0 models.
- The potential effect of read-ahead buffers are shown. For the 9337, the figure shows the effect of having 25% and 50% of the total disk operations already in the read-ahead buffer. For the 9336, it shows the effect of having 15% and 30% of the total disk operations already in the read-ahead buffer. The percentages used for the 9337 are higher because it has a larger and improved read-ahead buffer compared to the 9336. Depending on your data access patterns, the buffers may provide significant performance improvements.
- The 6110 and 6111 are Version 1 IOPs. The 6110 is used for 9332 and 9335; the 6111 is used for 9336. The 6112 is a Version 2 IOP and is used for 9332, 9335, and 9336. The 6500 is a Version 2 IOP used for 9337-0xx and 9337-1xx models. The 2611 is a Version 1 IOP used for 9337-0xx and 9337-1xx models. The 6501 is a Version 2 IOP used for 9337-2xx models. Up to two disk units of 9337-2xx can be connected to a single 6501 IOP.
- "Base" refers to the integrated disk units included with model D and later 9406 models.

Batch Performance

Figure 11-2 compares 9406 disk unit subsystem configurations when running a typical batch job. This scenario assumes that 75% of the duration of a batch job depends on the performance of the disk subsystem. The IO operations are a combination of 2KB, 8KB, and 16KB transfers. Seventy percent are reads, and 30% are writes. Twenty percent of the read and write operations require a seek over 1/3 of the disk surface, while 80% require no seek. Data shown is based on typical interactive disk I/O operation, which is not representative of a specific customer environment. Results in other environments may vary significantly.

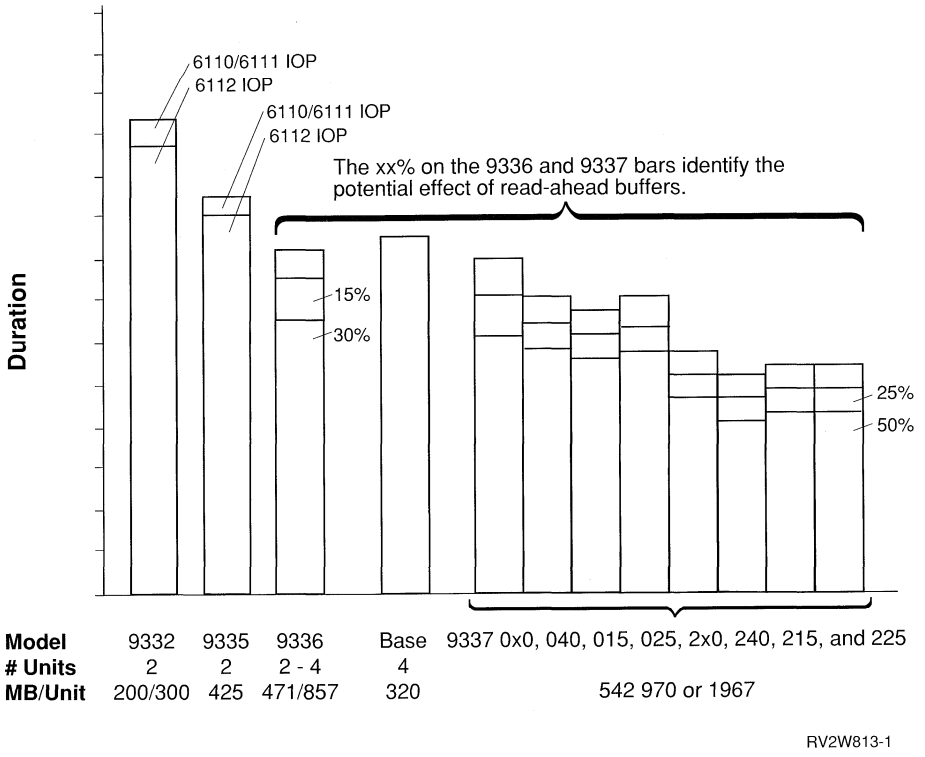


Figure 11-2. 9406 Disk Subsystem Batch Performance

Observations

- The 9337-2xx models provide better performance than the same 9337-0xx configurations.
- The 9337-xx5 models provide better performance than the other 9337-xx0 configurations.
- The 9337-040 model provides 10% to 15% better performance than the other 9337-0x0 configurations.
- The potential effect of read-ahead buffers are shown, assuming that 25% and 50% of the total disk operations for the 9337 are already in the read-ahead buffer. Read-ahead buffer effects of 15% and 30% are shown for the 9336. Depending on your data access patterns, read-ahead buffers may provide significant performance improvements.

Write-Intensive Applications with RAID-5

Like system checksum, RAID-5 can have a significant effect on batch programs that issue many write operations in a short period of time. This effect is caused by the four-times increase in the amount of disk I/O required for each write operation.

The 9337-2xx write cache handles this effect for almost all scenarios except those that write hundreds of records to the disk in a very short period of time. See the row labeled "3480/3490 Large file" in Table 11-1 on page 11-4 for an example of the effect of one worst-case scenario. Even in this worst-case scenario, with only one 9337-2xx configured, the restore operation for a large file took only 40% longer than a restore operation for a standard 9336 or 9337-0x0 configuration.

On 9337-1xx configurations, degradations of two times or more are possible when compared to a standard or mirrored environment. In these types of scenarios, the 9337-1xx combination of memory and write-assist disk unit (WAD) may become saturated and lose its performance advantage. The applications may have to wait for I/O operations on all four disks to complete (as in system checksum) before the applications can continue.

The saturation of the 9337 high-availability (HA) models is more likely when there is only one 9337. If more than one 9337 HA model is installed, the disk I/O operations may be spread across the multiple 9337s. A significantly higher

rate of disk write operations would be required before the 9337 HA configuration would run noticeably slower.

Table 11-1 shows restore performance for several workloads and configurations. In the worst case shown, there is only one 9337-1xx. In the other case shown, there are more GB of disk (six 9337-1xx configurations). The source-members workload is a Restore Object (RSTOBJ) operation of 2MB of data that consists of 200 source members. This workload results in many smaller write operations to disk. The install operation would be similar to this workload. The large-file workload is a RSTOBJ operation of a single 200MB file. This workload results in many large write operations to disk. If you are running multiple batch jobs concurrently, the results could be similar to the 3480/3490 restore operation.

Data can be restored to a 9337-1x0 faster than the rate of a 9332, 9335, 9336, or 9337 that is using system checksum protection. The restore rate to any device using system checksum protection can be significantly slower than a restore operation to a standard or mirrored device.

The save and restore rates are affected by

- Disk configuration
- Tape configuration
- Types of data being saved and restored

Table 11-1 identifies the approximate time differences for restoring data to a device using system checksum protection versus restoring data to a standard or mirrored device. Each line is normalized to a standard configuration of 9336 or 9337-0x0 disk drives.

Table 11-1. Approximate Time Differences for Restoring Data. Restoring data to a device using system checksum protection versus restoring data to a standard or mirrored device.

Tape Drive	Data	9336 or 9337-0x0 Standard	Mirrored Protection	System Checksum Protection	9337-1xx HA, 1 Disk Unit	9337-1xx HA, 6 Disk Units	9337-2xx HA, 1 Disk Unit
3480/3490	Source members	1	1.1	2.8	1.6	1	0.5
3480/3490	Large file	1	1.1	4.4	2.2	1.4	1.4
Medium[1]	Source members	1	1.1	2.8	1.6	1	1
Medium[1]	Large file	1	1.1	3.0	2.2	1.4	1
Slower[2]	Source members	1	1	1	1	1	1
Slower[2]	Large file	1	1	1	1	1	1

1 Medium-speed tape drives = 2440, 9348, and 3422
2 Slower tape drives = 7208, 9346, 9347, 6366, and 3430

Example of a Write-Intensive Application with RAID-5

If a restore operation for a large file takes 20 minutes on a standard 9348 configuration, it would take approximately 22 minutes (20 x 1.1) on a mirrored configuration, approximately 60 minutes (20 x 3) on a system checksum configuration, approximately 44 minutes (20 x 2.2) on a 9337-120 configuration, and about 20 minutes (20 x 1) on a 9337-2xx HA.

The 9337-040 and 9337-0x5 have about a 12% better restore performance than the 9337-020. The 9337-140 and 9337-1x5 have about a 25% better restore performance than the 9337-120.

The 9337 HA models offer significant advantages in availability, reliability, and price. One of the costs of the availability advantage is the increased time to restore data. This increase in time needs to be considered when planning the installation of RAID-5 disk. The time to load data onto the RAID-5 disk units must be included in the overall installation planning. With the increased availability and reliability offered by 9337 HA, the

necessity to reload the data because of a single 9337 HA disk unit failure is virtually eliminated.

What 9337 Configuration Should I Select?

Figure 11-3 on page 11-6 shows the "rule of thumb" for the IO/second/GB of usable space that each of the 9337 models can achieve. The top of each bar is the volume of 3.5K data transfer that each model can achieve when it is 40% busy. Seventy percent are reads, and 30% are writes. Eighty percent of the read and write operations require a seek over 1/3 of the disk surface, while 20% require no seek. For the 9337-2xx models, the write cache is assumed to have an efficiency of 50%. The vertical scale is the volume of physical I/O operations issued from the system. The RAID-5 mode bars are lower because of the additional work that the 9337 must do to maintain the RAID-5 parity stripes. Data shown is based on typical interactive disk I/O operation, which is not representative of a specific customer environment. Results in other environments may vary significantly.

Observations

- The 9337-240 models can now be configured where 9337-020 or 9337-120 models were previously required.
- The 9337-x40 models (1967 MB per disk unit) can provide good performance for many environments. Eight MB is the minimum amount of disk space that can be configured using 9337-x40 models.
- The 9337-x2x models (970 MB per arm) is the appropriate choice for almost all other situations. A configuration of 9337-x2x has approximately twice as many disk arms as a 9337-x40 configuration of equal capacity. The additional disk arms can provide a significant performance benefit at higher numbers of I/O operations per second per GB of capacity.
- The 9337-x1x models (542 MB per arm) are available to support those few AS/400 configurations with extremely high rates of I/O operations per second per GB of capacity. A configuration of 9337-x1x has approximately four times as many disk arms as a 9337-x40 configuration of equal capacity. The additional disk arms can provide a significant performance benefit at extremely high rates of I/O operations per second per GB of capacity.

Summary

- The new 9337-2xx models are significantly faster than previous disk offerings. Compared with equivalent configurations of 9337-0xx and 9337-1xx, a typical interactive transaction response time is up to 20% faster, and a typical batch job duration is up to 30% faster.
- Compared to existing 9337-0x0 or 9336 configurations, the 9337-2xx provides improved performance, even when the 9337-2xx is configured for high-availability mode.
- The new 9337-2xx write cache provides significantly improved performance.
- The new 9337-xx5 disk units provide performance improvements compared to the 9337-xx0. A typical interactive transaction response time is up to 10% faster, and a typical batch job duration is up to 30% faster.
- The 9337 high-availability models offer significantly increased protection from a disk unit failure compared to system checksum, at a significantly reduced price compared to mirroring.
- The 9337 offers significantly improved reliability compared to the 9336, 9335, or 9332.
- The 9337 offers increased GB per rack.

We suggest the use of BEST/1 Capacity Planner (5738-PT1) to help determine the configuration of 9337 that is best for your situation. In addition to many other benefits, BEST/1 allows you to compare the performance of various configurations of disk. Also, the Sales Force has access to additional performance comparison information that they can discuss with you.

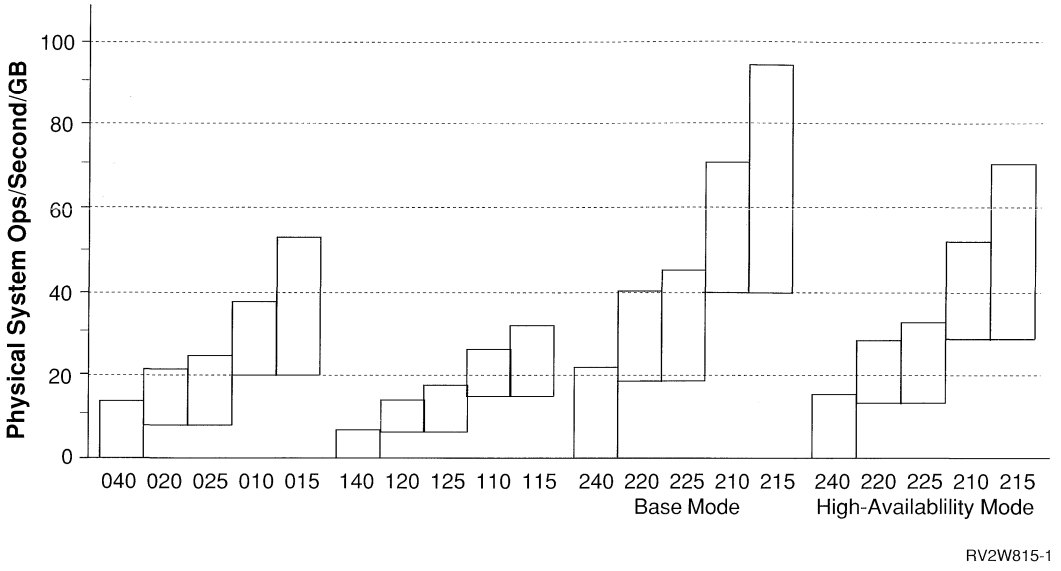


Figure 11-3. 9337 Disk Array Subsystem Operations/Second/GB of Capacity

Chapter 12. Working with Device Parity Protection

Device parity protection is a data redundancy option used by the hardware of some disk unit subsystems. Device parity protection can improve system availability.

To work with device parity protection, select option 6 (Work with device parity protection) on the Work with Disk Units display. This option provides you with functions that allow you to perform the system preparation necessary for starting or stopping device parity protection.

Preparing to Start Device Parity Protection

The preparing to start device parity function works only for the 9337 models 211, 221, and 241 disk unit subsystems. When preparing to start device parity protection, the system does validity checking, moves extents from the required units, and notifies you when you or your service representative can begin the procedure to start device parity protection on the specified disk unit subsystem.

Restrictions for Preparing to Start Device Parity Protection

The following restrictions apply when preparing to start device parity protection:

- The configuration must be complete and no disk units can be missing or suspended.
- The disk units that will become device parity protected can not be in a mirrored or checksummed ASP. Device parity protection must be started before mirrored protection in order to mirror some units in an ASP while giving device parity protection to the other units in the ASP. Checksum protection and device parity protection can not be mixed in the same ASP.
- Starting device parity protection reduces the capacity of some of the disk units in the subsystem. The system must have sufficient storage in each affected ASP to make room for redundant parity data.

Steps for Preparing to Start Device Parity Protection

To prepare to start device parity protection, do the following.

Access DST options:

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                      Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage



Selection
     _

F3=Exit           F12=Cancel
```

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools (DST) menu and press the Enter key. The following display is shown.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display and press the Enter key. The following display is shown.

```
                       Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 6 (Work with device parity protection) on the Work with Disk Configuration display and press the Enter key. The following display is shown.

```
                    Work with Device Parity Protection

Select one of the following:

     1. Display device parity status
     2. Prepare to start device parity protection
     3. Prepare to stop device parity protection
```

4. Select option 2 (Prepare to start device parity protection) on the Work with Device Parity Protection display and press the Enter key. The following display is shown.

```
                    Prepare to Start Device Parity Protection

Select one or more of the following disk subsystems to prepare
for starting device parity protection.

Type choices, press Enter.
  1=Prepare subsystem for starting device parity protection

                              Serial
Option   Type   Model  Number      Address
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
  -      ____   ____   __________  ____________
                                                                More...

F3=Exit          F12=Cancel
```

5. Type a 1 in the *Option* column for the disk unit subsystems you want to prepare to start device parity protection. Press the Enter key.

   The following display may be shown if a partial IPL is required.

```
                              Confirm Continuation

In order to proceed the system must perform internal processing
that may take several minutes during which the system may appear
inactive.  Once you confirm to continue, the system must
perform an IPL when you leave Work with Disk Configuration
functions.

Press Enter to continue.
Press F12=Cancel to return to change your choice.
```

6. Press the Enter key to continue. The following display is shown.

```
          Confirm Preparation for Starting Device Parity Protection

Preparing to start device parity protection will move system data
from parts of some disk units that will contain redundant device
parity data once device parity protection is started.  Moving
data may take several minutes for each subsystem selected.

Configured disk units in the selected storage subsystems will
become missing disk units at this time.

Press Enter to continue.
Press F12=Cancel to return and change your choice.

                          Serial
Option   Type   Model     Number       Address
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
                                                    More...

F12=Cancel
```

7. Press the Enter key to continue. The following display is shown.

```
                    Ready to Start Device Parity Protection

The system has completed its preparation for starting device
parity protection on the selected subsystems.  Consult the
appropriate subsystem documentation for the procedures to complete
starting device parity protection.

Press Enter to continue.

                          Serial
Option   Type   Model     Number       Address
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
                                                    More...

F3=Exit          F12=Cancel
```

8. The system has completed its preparation for starting device parity protection on the selected subsystems. At this time, you or your service representative should consult the *9406 System Installation and Upgrade Guide* for the procedures to complete starting device parity protection.

Preparing to Stop Device Parity Protection

When preparing to stop device parity protection, the system does validity checking to make sure that stopping device parity protection does not leave the system in a configuration that is not supported.

Stopping Restriction

- The configuration must be complete; no disk units can be missing or suspended.
- You cannot stop device parity protection on a subsystem when a unit in that subsystem is in a mirrored ASP. In order to stop device parity protection, mirrored protection must be stopped first.
- You cannot stop device parity protection on a subsystem containing data when one of the units in the subsystem has failed. Stopping device parity protection would destroy the redundant data required to reconstruct data on the failed unit.

Steps for Preparing to Stop Device Parity Protection

To prepare to stop device parity protection, do the following.

1. Access DST options:
   a. Notify the users to sign off the system by sending a break message.
   b. Change the QSYSOPR message queue to break mode:

      ```
      CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
      ```

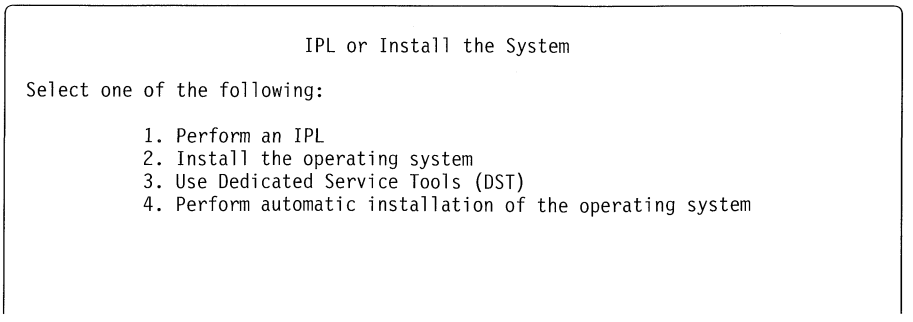
   c. End all subsystems:

      ```
      ENDSBS SBS(*ALL) OPTION(*IMMED)
      ```

      Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
   d. Ensure the key is in the keylock switch on the control panel.
   e. Turn the key until it points to the Manual position.
   f. Power down the system:

      ```
      PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
      ```

   g. When the system has powered down and then powered back up, the IPL or Install the System display appears.



h. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

i. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

The Use Dedicated Service Tools (DST) menu is shown.

```
                        Use Dedicated Service Tools (DST)

Select one of the following:

           1. Perform an IPL
           2. Install the operating system
           3. Work with licensed internal code
           4. Work with disk units
           5. Work with DST environment
           6. Select DST console mode
           7. Start a service tool
           8. Perform automatic installation of the operating system
           9. Work with save storage and restore storage




Selection
     _
F3=Exit            F12=Cancel
```

2. Select option 4 (Work with disk units) on the Use Dedicated Service Tools (DST) menu and press the Enter key. The following display is shown.

```
                              Work with Disk Units

 Select one of the following:

      1. Work with disk configuration
      2. Analyze disk device problem
      3. Work with disk unit recovery
      4. Work with disk unit information
```

3. Select option 1 (Work with disk configuration) on the Work with Disk Units display and press the Enter key. The following display is shown.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

4. Select option 6 (Work with device parity protection) on the Work with Disk Configuration display and press the Enter key. The following display is shown.

```
                  Work with Device Parity Protection

Select one of the following:

     1. Display device parity status
     2. Prepare to start device parity protection
     3. Prepare to stop device parity protection
```

5. Select option 3 (Prepare to stop device parity protection) on the Work with Device Parity Protection display and press the Enter key. The following display is shown.

```
                Prepare to Stop Device Parity Protection

Select one or more of the following disk subsystems to prepare
for stopping device parity protection.

Type choices, press Enter.
  1=Prepare subsystem for stopping device parity protection

                          Serial
Option   Type   Model     Number      Address
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
  _      ____   ____   ____________  ____________
                                                    More...

F3=Exit          F12=Cancel
```

6. Type a 1 in the *Option* column for the disk unit subsystems you want to prepare to stop device parity protection. Press the Enter key.

```
                Prepare to Stop Device Parity Protection

Select one or more of the following disk subsystems to prepare
for stopping device parity protection.

Type choices, press Enter.
  1=Prepare subsystem for stopping device parity protection

                          Serial
Option   Type   Model     Number       Address
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
                                                      More...
F3=Exit          F12=Cancel
```

7. Press the Enter key to continue. The following display is shown.

```
          Confirm Preparation for Stopping Device Parity Protection

Press Enter to continue.
Press F12=Cancel to return and change your choice.

                          Serial
Option   Type   Model     Number       Address
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
  _      ____   ____   ____________   ____________
                                                      More...
F12=Cancel
```

8. Press the Enter key to continue. The following display is shown.

```
                    Ready to Stop Device Parity Protection

The system has completed its preparation for stopping device
parity protection on the selected subsystems.  Consult the
appropriate subsystem documentation for the procedures to
stop device parity protection.

Press Enter to continue.

                          Serial
Option   Type   Model     Number       Address
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
  _      ____   ____   ___________   ____________
                                                        More...

F3=Exit          F12=Cancel
```

9. The system has completed its preparation for stopping device parity protection on the selected subsystems. At this time, you should consult the *9406 System Installation and Upgrade Guide* for the procedures to complete stopping device parity protection.

Displaying Device Parity Status

To display device parity status, do the following:

Access SST options:

1. Type the following and press the Enter key:

   STRSST

2. Select option 1 (Work with disk units).

   The following display is shown.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

Figure 12-1. Work with Disk Units Display for SST Options

3. Select option 1 (Display disk configuration) on the Work with Disk Units display and press the Enter key. The following display is shown.

```
                        Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units
     5. Display device parity status
```

Select option 5 (Display device parity status) on the Display Disk Configuration display and press the Enter key. The following display is shown.

```
                        Display Device Parity Status

                Serial
ASP  Unit       Number       Type  Model  Address        Status
                10-00A7498   9337   100   0010-03FFFFFF
 1     5        10-10A7498   9337   012   0010-0300FFFF  Active
 1     6        10-20A7498   9337   013   0010-0301FFFF  Active
 1     7        10-30A7498   9337   012   0010-0302FFFF  Active
 2     8        10-40A7498   9337   013   0010-0303FFFF  Active
 2     9        10-50A7498   9337   012   0010-0304FFFF  Active
 3    10        10-60A7498   9337   013   0010-0305FFFF  Active
 1    11        10-70A7498   9337   012   0010-0306FFFF  Active
                10-0444742   9337   100   0010-04FFFFFF
 4    12        10-1444742   9337   012   0010-0400FFFF  Unprotected
 4    13        10-2444742   9337   013   0010-0401FFFF  Unprotected
 4    14        10-3444742   9337   012   0010-0402FFFF  Unprotected
 2    18        10-4444742   9337   013   0010-0403FFFF  Failed
 2    19        10-5444742   9337   012   0010-0404FFFF  Unprotected
 3    20        10-6444742   9337   013   0010-0405FFFF  Unprotected
 5    21        10-7444742   9337   012   0010-0406FFFF  Unprotected

Press Enter to continue.

F3=Exit   F5=Refresh    F11=Display disk configuration status  F12=Cancel
```

Use the following information to determine the device parity status.

- *ASP*. Indicates the auxiliary storage pool (ASP) number in the current disk configuration. The valid values are 1 through 16. The system auxiliary storage pool, ASP 1, always exists. ASPs 2 through 16 are available for you to define as user ASPs.
- *Unit*. Indicates a number assigned by the system to identify a specific unit. If an asterisk (*) shows to the right of the unit number, the unit could not be found in the current configuration. However, system records indicate that the unit was present during the last IPL of the system.
- *Serial Number*. Indicates the serial number assigned by the manufacturer to identify a specific disk unit. A disk unit may have more than one unit.

  If the disk unit has more than one unit, then the units in the disk unit may all have the same serial number.
- *Type*. Indicates the number assigned by the manufacturer to identify a specific type of disk unit.
- *Model*. Indicates the set of numbers or letters used to identify the feature level for a product of a specific type.

- *Address. (1234-5678FFFF)*

Position	Specifies
1-4	The location of the storage device controller card. See the *Operator's Guide* for the physical location of the card.
5-6	The disk unit functional controller.
7-8	The disk unit.
FFFF	The device or channel. It has no significance for disk units and controllers.

  Note: Note that positions 7-8 will contain FF for controllers.

- *Status.* Indicates the status of each unit in a disk unit subsystem that has device parity protection.

  The valid status values for the units with device parity protection are:

  Active. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is fully operational.

  Failed. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has failed. If another unit in the disk unit subsystem fails, data could be lost.

  Rebuild. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The data on this unit is being rebuilt from other units in the disk unit subsystem.

  Unprotected. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is operational. However, another unit in the disk unit subsystem has failed or is being rebuilt. If another unit in the disk unit subsystem fails, data could be lost.

  HDW Failure. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A hardware-related failure has occurred. The failure does not affect data or performance. However, an exposure to an outage exists if another failure of a redundant component, such as a power supply, occurs.

  Degraded. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A decrease in performance has occurred because a component that is not critical has failed. The failed component needs to be repaired or replaced.

  Power Loss. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has lost power.

  Not Ready. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The unit is not ready to perform read and write operations.

  Unknown. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The status of this unit is not known to the system.

Part 6. Mirrored Protection

Chapter 13. Introduction to Mirrored Protection

The information in this chapter is intended to be used by you, your IBM marketing representative or re-marketing representative, and your service representative. The procedures in this part require the support of your IBM marketing representative and your service representative.

This chapter provides the following:

- Overview of mirrored protection
- Terminology used with mirrored protection.
- An overview of how mirrored protection works, including information about levels of protection, concurrent maintenance, system performance, and protection limitations.
- A review of how the system identifies disk units.

Overview of Mirrored Protection

Mirrored protection is a function that increases the availability of the AS/400 system in the event of a failure of a disk-related hardware component. It can be used on most models of the AS/400 system and is a part of the licensed internal code. Different levels of mirrored protection are possible, depending on what hardware is duplicated. The system remains available during a failure of a disk-related hardware component, such as a disk unit, a disk controller, a disk I/O processor, or a bus, if the failing hardware component and hardware components attached to it are duplicated. For the 9406 system unit, some failed hardware components can be serviced while the system remains available.

Mirrored protection provides these benefits:

- It keeps the system available when most disk-related hardware failures occur. For a disk unit failure, you do not need to restore your data and wait for a long recovery.
- It reduces the possibility of data loss by keeping the two copies of each unit's data on two different storage units. Changes to the data are written to both storage units of the mirrored pair. Data is read from either storage unit.
- For the 9406, it allows a degree of concurrent maintenance. Disk-related hardware failures can often be repaired while using the system.
- System performance is comparable to a non-mirrored system. System performance can improve, depending on the number of disk-related hardware components that are duplicated and depending on the application environment.
- Starting mirrored protection is fast and easy. You do not need to restore your data after starting mirrored protection.

Mirrored protection should not be used as a replacement for system backup procedures. You should continue to save the system on a regular basis because there are times when mirrored protection cannot be used to recover data. Mirrored protection reduces the times you have to restore from your backup copy. For more information, see "Mirrored Protection Limitations" on page 13-5.

Mirrored Protection Terminology

The following is a list of terms used in referring to mirrored protection.

ASP (auxiliary storage pool): A group of units defined from the disk storage devices. ASPs provide a means of isolating objects on specific units to prevent loss of data because of disk failures of other disk units in different ASPs. Performance may be improved by isolating heavily referenced objects in a user ASP. Every system has a system ASP which is ASP 1. You can create ASPs 2 through 16.

Bus: One or more conductors used for transmitting signals or power. It provides all signal, power, and ground connections to the internal adapters.

Concurrent maintenance: The process of repairing or replacing a failed disk-related hardware component while using the system. Concurrent maintenance is only possible on the 9406 with mirrored protection or the 9406 with disk units with device parity protection.

Controller: A device that coordinates and controls the operations of one or more I/O devices and synchronizes the operations of such devices with the operation of the system as a whole.

Deferred maintenance: The process of waiting to repair or replace a failed disk-related hardware component until the system can be powered down. The system is available, although mirrored protection is reduced by whatever hardware components have failed. Deferred maintenance is only possible with mirrored protection or device parity protection.

Disk Unit: The physical enclosure containing one or more storage units.

Examples:

- A 6105 disk unit has a single storage unit.
- A 9335-B01 disk unit has 2 storage units in its enclosure.

I/O processor: The I/O processor is attached to the bus and controls information between the bus and specific groups of I/O controllers and disk units. The I/O processor is also called a storage controller or a magnetic storage device controller.

Mirrored protection: A function that protects data by duplicating all disk data on one storage unit in an auxiliary storage pool on another storage unit in the same auxiliary storage pool. If a disk failure occurs, the system continues to run using the remaining storage unit of the mirrored pair.

Mirrored pair: Two storage units that contain the same data and are referred to by the system as one unit.

Mirrored unit: The storage unit that is half of a mirrored pair.

Storage Unit: The defined space within a disk unit that is addressed by the system.

Unit: The defined division of single-level storage. This space is the smallest disk location addressable by the user. An ASP is one or more units which are identified by unique unit numbers. A unit in a non-mirrored ASP is one storage unit. A unit in a mirrored ASP is a mirrored pair which is two storage units.

Certain create commands (CRTPF, CRTJRNRCV, etc) can create an object on a specified unit. In the non-mirrored environment this is a single storage unit. In the mirrored environment, the UNIT parameter value means a mirrored *pair*.

How Mirrored Protection Works

Because mirrored protection is configured by auxiliary storage pool (ASP), you can mirror one, some, or all ASPs on the system. By default, every system has a system ASP. It is not necessary to create user ASPs in order to use mirrored protection. Although mirrored protection is configured by ASP, all ASPs must be mirrored to provide for maximum system availability. If a disk unit fails in an ASP that is not mirrored, the system cannot be used until the disk unit is repaired or replaced.

The start mirrored pairing algorithm automatically selects a mirrored configuration that provides the maximum protection at the bus, I/O (input/output) processor, or controller level for the hardware configuration of the system. When storage units of a mirrored pair are on separate buses, they have maximum independence or protection. Because they do not share any resource at the bus, I/O processor, or controller levels, a failure in one of these hardware components allows the other mirrored unit to continue operating. The pairing algorithm does not allow mirroring between storage units in the same disk unit when replacement of a common part could cause data loss for both storage units. For example, the two storage units in a 9332 or a 9335 disk unit are not paired to make a mirrored pair.

Any data written to a unit that is mirrored is written to both storage units of the mirrored pair. When data is read from a unit that is mirrored, the read operation can be from either storage unit of the mirrored pair. It is transparent to the user which mirrored unit the data is being read from. A user is not aware of the existence of two physical copies of the data.

If one storage unit of a mirrored pair fails, the system *suspends* mirrored protection to the failed mirrored unit. The system continues to operate using the remaining mirrored unit. The failing mirrored unit can be physically repaired or replaced.

After the failed mirrored unit is repaired or replaced, the system *synchronizes* the mirrored pair by copying current data from the storage unit that has remained operational to the other storage unit. During synchronization, the mirrored unit to which the information is being copied is in the *resuming* state. If repairing the failed storage unit requires replacement of the other storage unit in the disk unit, (as in a 9335 Model B01), synchronization occurs to both storage units in the repaired disk unit. Synchronization does not require a dedicated system and runs concurrently with other jobs on the system. System performance is affected during synchronization. When synchronization is complete, the mirrored unit becomes *active*.

Level of Protection

The level of mirrored protection determines if the system keeps running when different levels of hardware fail. Mirrored protection always provides disk unit-level protection which keeps the system available for a single disk unit failure. To keep the system available for failures of other disk-related hardware requires higher levels of protection. For example, to keep the system available when an I/O processor fails, all of the disk units attached to the failing I/O processor must have I/O processor level protection. That is, all of the disk units attached to the failing I/O processor must have active mirrored units attached to a different I/O processor.

For the 9406, the level of mirrored protection also determines if concurrent maintenance can be done for different types of failures. Certain types of failures require concurrent maintenance to diagnose hardware levels above the failing hardware component. For example, to diagnose a power failure in a disk unit requires resetting the I/O processor to which the failed disk unit is attached. Therefore, I/O processor-level protection is required. The higher the level of mirrored protection, the more often concurrent maintenance is possible.

The level of protection you get depends upon the hardware you duplicate. If you duplicate disk units, you will have disk unit-level protection. If you duplicate disk unit controllers as well, you have controller-level protection. If you duplicate I/O processors, you have I/O processor-level protection. If you duplicate busses, you have bus-level protection.

Figure 13-1 on page 13-4 shows four hardware configurations of mirrored protection. In each configuration, the two storage units shown make a mirrored pair.

1. The first (upper left) configuration shows disk unit-level protection. With disk unit-level protection, the system continues to operate after a disk unit failure. If the controller or I/O processor fails, the system cannot access data on either of the storage units of the mirrored pair, so the system is unusable.
2. The next (upper right) configuration shows disk controller-level protection. With controller-level protection, the system can continue to operate after a disk controller failure. If the I/O processor fails, the system cannot access data on either of the disk units, and the system is unusable.
3. The next (lower left) configuration shows I/O processor-level protection. With I/O processor-level protection, the system can continue to operate after an I/O processor failure. Only if the bus fails will the system become unusable.
4. The last (lower right) configuration shows bus-level protection. With bus-level protection, the system can continue to operate after a bus failure. However, the system cannot continue to operate if bus 0 fails.

During the start mirrored protection operation and during the add operation to a mirrored ASP, the system pairs the disk units to provide the maximum level of protection for the 20 The hardware configuration is both the hardware and how the hardware is connected.

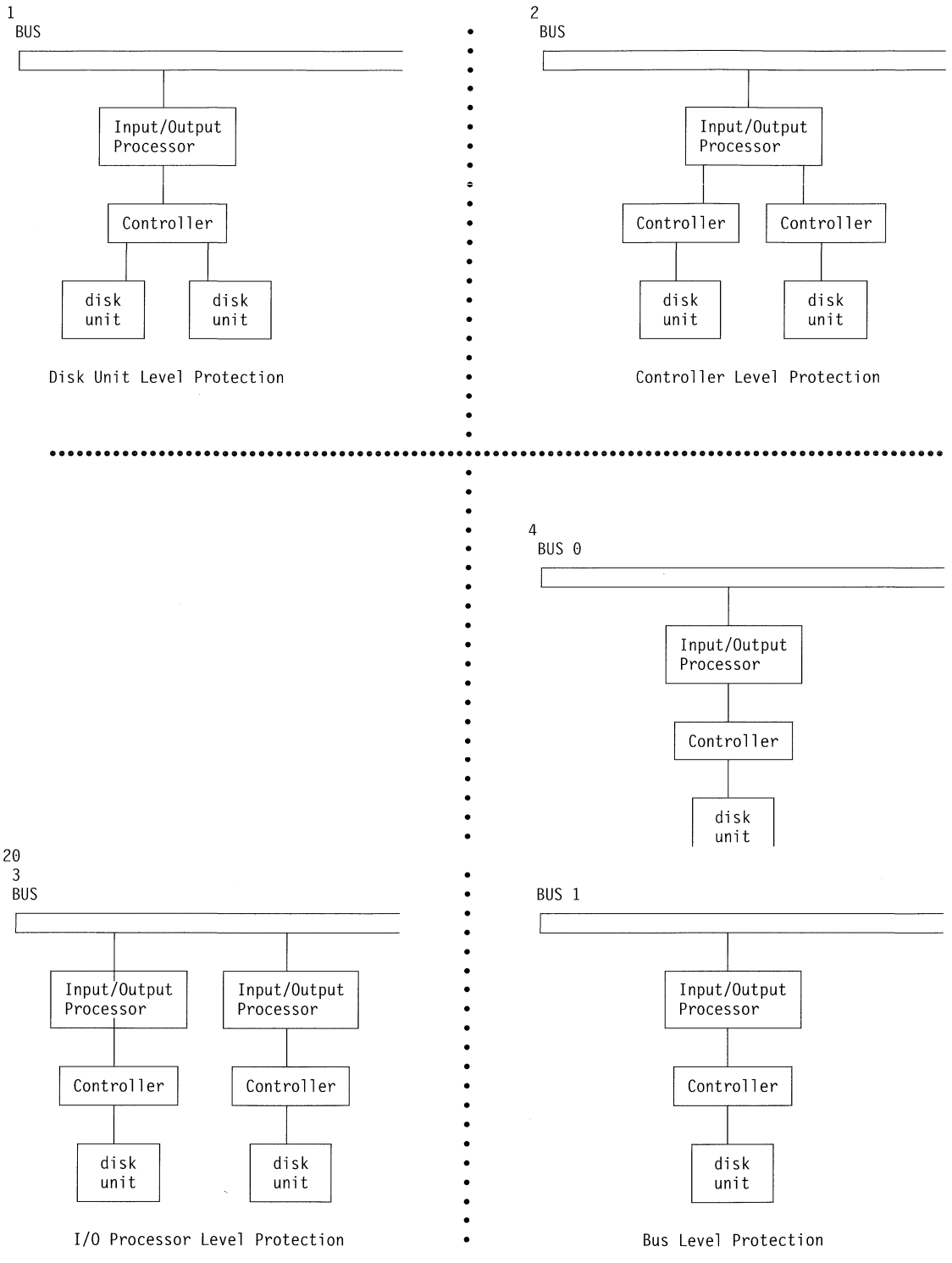


Figure 13-1. Mirrored Protection Configurations

Mirrored Protection Limitations

Although mirrored protection can keep the system available after disk-related hardware failures occur, it is not a replacement for save procedures. There can be multiple types of disk-related hardware failures, or disasters (such as flood or sabotage) that require backup media.

Mirrored protection cannot keep your system available if the remaining storage unit in the mirrored pair fails before the first failing storage unit is repaired and mirrored protection is resumed. If two failed storage units are in different mirrored pairs, the system is still available and normal mirrored protection recovery is done because the mirrored pairs are not dependent on each other for recovery. If a second storage unit of the same mirrored pair fails, the failure may not result in a data loss. If the failure is limited to the disk electronics, or if the service representative can successfully use the Save Disk Unit Data function to recover all of the data, no data is lost.

If both storage units in a mirrored pair fail causing data loss, the entire ASP is lost and all units in the ASP are cleared. You must be prepared to restore your ASP from the backup media and apply any journal changes. (See the *Basic Backup and Recovery Guide* if the system ASP is lost or if a user ASP is lost.)

When starting the mirrored protection operation, objects created on a preferred unit may be moved to another unit. The preferred unit may no longer exist after mirror protection is started.

System Performance When Mirrored Protection is in Effect

Most systems should show little difference in performance. In some cases, mirrored protection can improve performance. Mirrored protection normally requires additional disk units and input/output processors. Since mirrored protection uses some main storage, some systems may require additional main storage to maintain the same level of performance.

Generally, functions that do mostly read operations see equal or better performance with mirrored protection. This is because read operations have a choice of two storage units to read from, and the one with the faster expected response time is selected. Operations that do mostly write operations (such as updating database records) may see slightly reduced performance on a system that has mirrored protection because all changes must be written to both storage units of the mirrored pair. Thus, restore operations are slower.

For details on system performance with mirrored protection, see Chapter 18, "Performance Considerations for Mirrored Protection" on page 18-1 and "Capacity Planning Tools" on page 17-2.

Additional IPL Time after an Abnormal System End: In some cases, if the system ends abnormally, the system cannot determine whether the last updates were written to both storage units of each mirrored pair. If the system is not sure that the last changes were written to both storage units of the mirrored pair, the system synchronizes the mirrored pair by copying the data in question from one storage unit of each mirrored pair to the other storage unit. The synchronization occurs during the IPL following the abnormal system end. If the system can save a copy of main storage before it ends, the synchronization process takes just a few minutes. If not, the synchronization process can take much longer. The extreme case could be close to a complete synchronization.

If you have frequent power outages, you may want to consider adding an uninterruptible power supply to your system. Should main power be lost, the uninterruptible power supply allows the system to continue. A basic uninterruptible power supply allows the system time to save a copy of main storage before ending which avoids long recovery. Both storage units of the load source mirrored pair must be powered by the basic uninterruptible power supply.

Spare Disk Units

The amount of time that the system runs with reduced mirrored protection after a disk-related hardware failure can be minimized by keeping spare storage units. If a disk unit fails, a spare may be used to replace the failed storage unit while the system is running. The new mirrored pair is then synchronized and mirrored protection

is available while the failed storage unit is being serviced.

Overview of Concurrent Maintenance

On systems without mirrored protection, the system is not available when a disk-related hardware failure occurs and remains unavailable until the failed hardware is repaired or replaced. However, with mirrored protection the failing hardware can often be repaired or replaced while the system is being used.

Concurrent maintenance is supported only for a 9406 system unit that has mirrored protection. The best hardware configuration for mirrored protection also provides for the maximum amount of concurrent maintenance.

It is possible for the system to operate successfully through many failures and repair actions. For example, a failure of a disk head assembly will not prevent the system from operating. A replacement of the head assembly and synchronization of the mirrored unit can occur while the system continues to run. The greater your level of protection, the more often concurrent maintenance can be performed.

On some models, the system restricts the level of protection for unit 1 and its mirrored unit to only controller-level protection. See rule 5 on page 15-1 for restrictions.

Under some conditions, diagnosis and repair can require active mirrored units to be suspended. You may prefer to power down the system to minimize the exposure of operating with less mirrored protection. Some repair actions require that the system be powered down. Both of these are *deferred maintenance* situations.

Review of How the System Addresses Storage

Disk units are assigned to an auxiliary storage pool (ASP) on a storage unit basis. The system treats each storage unit within a disk unit as a separate unit of auxiliary storage. When a new disk unit is attached to the system, the system initially treats each storage unit within it as nonconfigured. Through Dedicated Service Tools (DST) options you can add these nonconfigured storage units to either the system ASP or a user ASP of your choosing. When adding nonconfigured storage units, use the serial number information assigned by the manufacturer to ensure you are selecting the correct physical storage unit. Additionally, the individual storage units within the disk unit can be identified through the *Address* field on the DST Display Disk Configuration display. For more information about displaying disk configuration, see Chapter 7, "Working with Auxiliary Storage Pools"

When you add a nonconfigured storage unit to an ASP, the system assigns a unit number to the storage unit. The unit number can be used instead of the serial number and address. The same unit number is used for a specific storage unit even if you connect the disk unit to the system in a different way.

When a unit has mirrored protection, the two storage units of the mirrored pair are assigned the same unit number. The serial number and the address distinguish between the two storage units in a mirrored pair.

To determine which physical disk unit is being identified with each unit number, make note of the unit number assignment to ensure correct identification. If a printer is available, print the DST or SST display of your disk configuration. If you need to verify the unit number assignment, use the DST or SST Display Configuration Status display to show the serial numbers and addresses of each unit.

The address has twelve digits and is identified as follows:

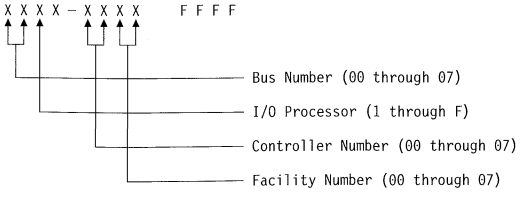


Figure 13-2. Resource Address

The storage unit addressed by the system as unit 1 is always used by the system to store licensed internal code and data areas. The amount of storage used on unit 1 is quite large and varies depending on the configuration of your system.

Unit 1 contains a limited amount of user data. Because unit 1 contains the initial programs and data used during an IPL of the system, it is also known as the *load source unit*.

The system reserves a fixed amount of storage on units other than unit 1. The size of this reserved area is 1.08MB per unit, reducing the space available on each unit by that amount.

Chapter 14. Planning for Mirrored Protection

When considering mirrored protection, contact your IBM marketing representative to guide you through these steps.

You will need to do the following steps before you can start mirrored protection:

1. Decide whether to use mirrored protection.
2. Decide which ASP or ASPs to protect with mirrored protection.
3. Determine disk storage capacity requirements.
4. Determine the level of protection you want for each mirrored ASP.
5. Determine what extra hardware you need for mirrored protection.
6. Determine what extra hardware you need for performance, if any.
7. Order your hardware.
8. Plan the installation of your system and the configuration of new units.
9. Install the new hardware.

Step 1: Deciding Whether to Use Mirrored Protection

If you have a multi-bus system or a large single-bus system, you should consider using mirrored protection. The greater the number of disk units that are attached to a system, the more frequent disk-related hardware failures are, simply because there are more individual pieces of hardware that can fail. Therefore, the possibility of data loss or loss of availability as a result of a disk or other hardware failure becomes more likely. Also, as the amount of disk storage on a system increases, the recovery time after a disk storage subsystem hardware failure increases significantly. Downtime becomes more frequent, more lengthy, and more costly.

Benefits of Mirrored Protection

With the best possible mirrored protection configuration, the system continues to run after a single disk-related hardware failure. On a 9406 system unit, the failed hardware can sometimes be repaired or replaced without having to power down the system. If the failing component is one that cannot be repaired while the system is running, such as a bus or an I/O processor, the system usually continues to run after the failure. Maintenance can be deferred, the system can be shut down normally, and a long recovery time can be avoided.

Even if your system is not a large one, mirrored protection can provide you valuable protection. A disk or disk-related hardware failure on an unprotected system leaves your system unusable for several hours. The actual time depends on the kind of failure, the amount of disk storage, your backup strategy, the speed of your tape unit, and the type and amount of processing the system performs. If you or your business cannot tolerate this loss of availability, you should consider mirrored protection for your system, regardless of your system's size.

Costs of Mirrored Protection

The main cost of using mirrored protection is in additional hardware. To achieve high availability and prevent data loss when a disk unit fails, you need mirrored protection for all the ASPs. This normally requires twice as many disk units. If you want continuous operation and prevention of data loss when a disk unit, controller, or I/O processor fails, you need duplicate disk controllers and I/O processors. A model upgrade can be done to get nearly continuous operation and to prevent data loss when any of these failures occur, as well as the failure of bus 1. If bus 0 fails, the system cannot continue to operate. Because bus failures are rare, and bus-level protection is not significantly greater than I/O processor-level protection, you may not find a model upgrade to be cost-effective for your protection needs.

Mirrored protection has a minimal effect on performance. If the busses, I/O processors, and con-

trollers are no more heavily loaded on a system with mirrored protection than they are on an equivalent system without mirrored protection, then the performance of the two systems should be approximately the same.

In deciding whether or not to use mirrored protection on your system, you must evaluate the cost of potential downtime against the cost of additional hardware, over the life of the system. The additional cost in performance or system complexity is usually negligible. You should also consider other availability and recovery alternatives, such as checksum protection or device parity protection. Mirrored protection normally requires twice as many storage units. For concurrent maintenance and higher availability on systems with mirrored protection, other disk-related hardware may be required.

Step 2: Deciding Which ASPs to Protect with Mirrored Protection

Mirrored protection is configured by auxiliary storage pool, because the ASP is the user's level of control over single-level storage. Mirrored protection can be used to protect one, some, or all ASPs on a system. However, multiple ASPs are not required in order to use mirrored protection. Mirrored protection works well if all disk units on a system are configured into a single ASP (the default on the AS/400). In fact, mirroring reduces the need to partition auxiliary storage into ASPs for data protection and recovery. However, ASPs may still be desirable for performance and other reasons.

To provide the best protection and availability for the entire system, all ASPs in the system should have mirrored protection:

- If the system has a mixture of some ASPs with and some ASPs without mirrored protection, a disk unit failure in an ASP without mirrored protection severely limits the operation of the entire system. Data can be lost in the ASP in which the failure occurred. A long recovery may be required.
- If a disk fails in a mirrored ASP, and the system also contains ASPs that are not mirrored, data is not lost. However, in some cases, concurrent maintenance may not be possible.

The disk units used in user ASPs should be selected carefully. For best protection and performance, an ASP should contain disk units attached to several different I/O processors. The number of disk units in the ASP that are attached to each I/O processor should be the same (that is, balanced).

Warning: If your system contains 224MB or more of main storage, and the system ASP (ASP 1) contains only two 2800-001 storage units, two additional 2800-001 storage units must be added to the system ASP before mirrored protection can be started. See note 3 on page 6-10 for additional information. (The maximum number of 2800-001 storage units allowed on the system is four. You may need to move 2800-001 storage units from a user ASP to the system ASP.)

Mirrored protection and checksum protection should not be used in different ASPs on the same system. A disk unit failure in an ASP with checksum protection leaves the entire system unusable until the failure is repaired and checksum recovery is completed. For a somewhat larger hardware investment, the ASP can be mirrored.

Step 3: Determining Disk Units Needed for Mirrored Protection

A mirrored ASP requires twice as much auxiliary storage as an ASP that is not mirrored, because the system keeps two copies of all the data in the ASP. Also, mirrored protection requires an even number of storage units of each type of disk unit and model so that storage units can be made into mirrored pairs. On an existing system, it should be noted that it is not necessary to add the same types of disk units already attached in order to provide the required additional storage capacity. Any new disk units may be added as long as sufficient total storage capacity and an even number of storage units of each type are present. The system will assign mirrored pairs and automatically move the data as necessary. If an ASP does not contain sufficient storage capacity, or if storage units cannot be paired, mirror protection cannot be started for that ASP.

The process of determining the disk units needed for mirrored protection is similar for existing or new systems. For each ASP that is to be mir-

rored, you and your IBM marketing representative should:

1. Plan how much data the ASP will contain.
2. Plan a target percent of storage used for the ASP (how full the ASP will be).
3. Plan the number and type of disk units needed to provide the storage required. For an existing ASP, you can plan a different type and model of disk unit to provide the required storage. You must ensure an even number of each type of disk unit and model.

After planning for all ASPs is completed, plan for spare units, if desired.

Planning for Storage Capacity

For a new system, your IBM marketing representative or re-marketing representative can help you analyze your system storage requirements. For an existing system, the current amount of data in the ASP being planned is a useful starting point. The DST or SST Display Disk Configuration Capacity option shows the total size (in megabytes) and the percent of storage used for each ASP on the system. Multiply the size of the ASPs by the percent used to calculate the number of megabytes of data currently in the ASP. In planning future storage requirements for an ASP, system growth and performance should also be considered.

The planned amount of data and the planned percent of storage used work together to determine the amount of actual auxiliary storage needed for a mirrored ASP. For example, if an ASP is to contain 1GB (GB equals 1 073 741 824 bytes) of actual data, it requires 2GB of storage for the mirrored copies of the data. If 50% full is planned for that ASP, the ASP needs 4GB of actual storage. If the planned percent of storage used is 66%, 3GB of actual storage are required. One gigabyte of real data (2GB of mirrored data) in a 5GB ASP results in a 40% auxiliary storage utilization.

Calculating Mirrored Capacity

The Calculate mirrored capacity option does the arithmetic to determine the capacity provided by different numbers and types of disk units for an ASP. The ASP can currently be unprotected, checksum protected, or mirror protected. To plan for a new ASP or for an ASP on a different system, you can enter the number of an ASP that does not currently exist. You can change the amount of data in the ASP and the number of storage units of each disk type to calculate the percent of storage used for various storage units and amounts of data. The amount of data is in megabytes and is the amount of actual data, not the amount of mirrored data (which would be twice the actual data). The number of each disk unit type planned refers to individual storage units, not disk units, because mirroring is by storage unit. The Calculate mirrored capacity option is accessed using either DST or SST.

Planning for Spare Storage Units

Spare storage units can reduce the time the system runs without mirrored protection for a mirrored pair after a storage unit failure. If a storage unit fails and a spare unit of the same disk unit type and model is available, that spare unit can be used to replace the failed unit. Using the DST or SST replace option, the user selects the failed storage unit to replace, then selects a spare storage unit to replace it. The system logically replaces the failed unit with the selected spare unit, then synchronizes the new unit with the remaining good unit of the mirrored pair. Mirrored protection for that pair is again active when synchronization completes (in usually less than an hour). However, it might take several hours from the time a service representative is called until the failed unit is repaired and synchronized, and mirrored protection is again active for that pair.

To make full use of spare units, you need at least one spare unit of each type and model of disk unit you have on your system. This provides a spare for any type of disk unit that may fail. A failed unit must be replaced by a spare of the same disk unit type and model. An exception is the 9332 model 200s and model 400s, which can be used as spares for one another.

Total Planned Storage Capacity Needs

After planning for the number and type of storage units needed for each ASP on the system, and for any spare storage units, add up the total number of storage units of each disk unit type and model. Remember that the number planned is the number of storage units of each disk unit type, not the number of disk units. You and your IBM marketing representative will need to convert the planned number of storage units to disk units before ordering hardware.

The preceding procedure helps you plan the total number of disk units needed for your system. If you are planning for a new system, this is the number that needs to be ordered. If you are planning for an existing system, subtract the number of each disk type currently on your system from the number planned. This is the number of new disk units that should be ordered.

Step 4: Determining the Level of Protection

The amount of duplicate disk-related hardware determines your level of protection. The more mirrored pairs that have higher levels of protection, the more often your system will be usable when disk-related hardware fails. You may decide that a lower level of protection is more cost effective for your system than a higher level.

Figure 13-1 on page 13-4 shows the different levels of protection possible. When determining what level of protection is adequate, you should consider the relative advantages of each level of protection with respect to the following:

- The ability to keep the system operational during a disk-related hardware failure.
- For the 9406, the ability to perform maintenance concurrently with system operations. To minimize the time that a mirrored pair is unprotected after a failure, you may want to repair failed hardware while the system is operating.

Disk Unit-Level Protection

Mirrored protection always provides disk unit-level protection because the storage units are duplicated. If your main concern is protection of data and not high availability, then disk unit-level protection may be adequate. The disk unit is the most likely hardware component to fail, and disk unit-level protection keeps your system available after a disk unit failure.

For the 9406, concurrent maintenance is possible for certain types of disk unit failures with disk unit-level protection.

Controller-Level Protection

If your system has 9335 disk units, then determine if you want controller-level protection based on the following:

- To keep your system available when a 9335 A01 controller fails.
- To concurrently repair a failed disk unit or controller. To use problem recovery procedures in preparation for isolating a failing item or to verify a repair action, the controller (9335 A01) must be dedicated to the repair action. If any disk units attached to the controller do not have controller-level protection, then this part of concurrent maintenance is not possible.

To achieve controller-level protection, all 9335 B01 disk units attached to a 9335 A01 controller must have a mirrored unit attached to a different 9335 A01 controller.

I/O Processor-Level Protection

Determine if you want I/O processor-level protection based on the following:

- To keep your system available when an I/O processor fails.
- To keep your system available when the cable attached to the I/O processor fails.
- To concurrently repair certain types of disk unit failures or cable failures. For these failures, concurrent maintenance needs to reset the I/O processor. If any disk units attached to the I/O processor do not have I/O

processor-level protection, then concurrent maintenance is not possible.

To achieve I/O processor-level protection, all disk units attached to an I/O processor must have a mirrored unit attached to a different I/O processor. On many systems, I/O processor-level protection is not possible for the mirrored pair for unit 1. See rule 5 on page 15-1 for restrictions.

Bus-Level Protection

Bus-level protection may allow the system to run when a bus fails. However, bus-level protection is often not cost-effective because of the following:

- If bus 0 fails, the system is not usable.
- If a bus fails, disk I/O operations may continue, but so much other hardware is lost, such as work stations, printers and communication lines, that from a practical standpoint, the system is not usable.
- Bus failures are rare compared with other disk-related hardware failures.
- Concurrent maintenance is not possible for bus failures.

To achieve bus-level protection, all disk units attached to a bus must have a mirrored unit attached to a different bus. On many systems, bus-level protection is not possible for the mirrored pair for unit 1. Rule 5 on page 15-1 has more information on the restrictions for unit 1.

Step 5: Determining the Extra Hardware You Need for Mirroring

In order to communicate with the rest of the system, disk units are attached to controllers, which are attached to I/O processors, which are attached to busses. The number of each of these types of disk-related hardware available on the system directly affects the level of protection that is possible.

To provide the best protection and performance, each level of hardware should be balanced under the next level of hardware. That is, the disk units of each device type and model should be evenly distributed under their controllers. The same number of controllers should be under each I/O processor for that disk type. The I/O processors should be balanced among the available busses.

To plan what disk-related hardware is needed for your mirrored system, you must plan the total number and type of disk units (old and new), that will be needed on the system, as well as the level of protection for the system. It is not always possible to plan for and configure a system so that all mirrored pairs meet the planned level of protection. See 5 on page 15-1 for restrictions. However, it is possible to plan a configuration in which a very large percentage of the disk units on the system achieve the desired level of protection.

When planning for additional disk-related hardware, you should first determine the minimum hardware needed for the planned disk units to function. Plan for one disk unit type at a time. Next, plan the additional hardware needed to provide the desired level of protection for each disk unit type.

Step 5A: Planning the Minimum Hardware Needed to Function.

Various rules and limits exist on how storage hardware can be attached together. The limits may be determined by hardware design, architecture restrictions, performance considerations, or support concerns. Your IBM marketing representative can explain these configuration limits and help you use them in your planning. For a listing of the configuration limits and rules, see *9406 System Installation and Upgrade Guide*.

For each disk unit type, first plan for the controllers needed and then for the I/O processors needed. After planning the number of I/O processors needed for all disk unit types, use the total number of I/O processors to plan for the number of busses needed.

Step 5B: Planning Additional Hardware to Achieve the Level of Protection.

- Bus-level protection

  If you want bus-level protection and already have a multiple-bus system, you need to do nothing. If your system is configured according to standard configuration rules, the mirrored pairing function pairs up storage units

to provide bus-level protection for as many mirrored pairs as possible. If you have a single-bus system, a model upgrade is probably not cost-effective merely to achieve bus-level protection. You should plan instead for I/O processor-level protection.

- I/O processor-level protection

  If you want I/O processor-level protection and do not already have the maximum number of I/O processors on your system, add as many I/O processors as possible, keeping within the defined system limits. Then balance the disk units among them according to the standard system configuration rules. It generally is not cost-effective to add another bus (model upgrade) merely to attach more I/O processors, although a model upgrade may be desirable for performance or other reasons.

- Controller-level protection

  If the planned disk units do not require a separate controller (such as the 9332), you will already have controller-level protection for as many units as possible and you do not need to do anything more. If your planned disk units do require a separate controller (such as the 9335 B01), add as many controllers (9335 A01s) as possible, keeping within the defined system limits. Then balance the disk units among them according to the standard system configuration rules.

- Disk unit-level protection

  If you have planned for disk unit-level protection, you do not need to do anything more. All mirrored ASPs have a minimum of disk unit-level protection if they meet the requirements for starting mirrored protection.

Example of Planning for Additional Hardware

In this example, assume you are upgrading your 9406 Model B40. You plan to use all disk units in a single ASP and mirror that ASP.

- "Step 3: Determining Disk Units Needed for Mirrored Protection" on page 14-2:

  You planned for thirteen 9332 600 disk units and four 9335 B01 disk units.

- "Step 4: Determining the Level of Protection" on page 14-4:

  You decided that you want I/O processor-level protection on as many storage units as possible.

- "Step 5A: Planning the Minimum Hardware Needed to Function." on page 14-5:

  For the 9332 model 600 disk units that contain an internal controller, no extra controllers need be planned. Eight 9332 model 600 disk units can be attached to one I/O processor. Therefore, a minimum of two I/O processors are needed for the 9332 disk units.

  The 9335 model B01 disk units require a 9335 model A01 controller. A maximum of four 9335 model B01 disk units can be attached to a 9335 A01 controller. Therefore, you need to plan for one 9335 model A01 controller. The one 9335 A01 can be attached to a single I/O processor.

  After Step 5A, you know that you need a minimum of three I/O processors and one 9335 model A01 controller to go with your disk units planned.

- "Step 5B: Planning Additional Hardware to Achieve the Level of Protection." on page 14-5:

  You planned for only two I/O processors, but have an uneven number of disk units. This provides I/O processor-level protection on all but one mirrored pair of the 9332 disk units. To achieve I/O processor-level protection on all of the 9332 disk units requires a minimum of three I/O processors. However, adding a third I/O processor for the 9332 disk units requires a second bus and a model upgrade.

  In Step 5B, you planned for only one I/O processor for the 9335 disk units. At least two are needed to provide I/O processor-level protection. Since the 9335 B01 disk unit requires a 9335 A01 controller between the disk unit and the I/O processor, you must also plan for an additional 9335 A01 controller. You can attach both 9335 A01 controllers to the single I/O processor previously planned for and achieve controller-level protection for the 9335 disk units. If you add the second I/O processor, you need a second bus and a model upgrade.

  You have two choices:

- Be satisfied with less than I/O processor-level protection for all units. All but one mirrored pair of the 9332 disk units will have I/O processor protection. The 9335 disk units and the other pair of 9332 disk units will have controller-level protection.
- Upgrade to a model B50 and buy two more I/O processors and one more 9335 A01 controller. This provides I/O processor protection for all mirrored pairs except for unit 1. Even though a model upgrade is often not cost-effective merely to achieve a higher level of protection, a model upgrade may be worth consideration if your general processing and capacity needs may grow in the near future.

Step 6: Determining Extra Hardware for Performance

In addition to the hardware required to provide the level of mirror protection you want, your system may need additional hardware to achieve the level of performance you want.

Processing Unit Requirements

Mirrored protection causes a minor increase in central processing unit usage (approximately 1% to 2%).

Main Storage Requirements

Mirrored protection uses the following amounts of main storage:

- Mirrored protection uses approximately 30KB of main storage from the machine pool for general purposes.
- Mirrored protection uses 4KB from the machine pool for each mirrored pair.

During synchronization, mirrored protection uses an additional 68KB for each mirrored pair that is being synchronized. The system uses the pool with the most storage.

I/O Processor Requirements

To maintain equivalent performance after starting mirrored protection, your system should have the same ratio of disk units to I/O processors as it did before. To add I/O processors, you may also need additional controllers (for example, 9335 A01). To add I/O processors, you may need to upgrade your system for additional busses.

Because of the limit on busses and I/O processors, you may not be able to maintain the same ratio of disk units to I/O processors. In this case, system performance may be less.

Step 7: Ordering Your New Hardware

Your IBM marketing representative will assist you in ordering your new hardware using the usual order process. That ordering process allows for any other hardware that may be needed as part of your upgrade, such as additional racks and cables.

Step 8. Planning Your Installation

You must work with your IBM marketing representative to plan for the installation of mirrored protection on your system. The marketing representative will help you determine whether your system is balanced and meets standard configuration rules, as defined in the *9406 System Installation and Upgrade Guide*. The system must be configured according to the standard rules in order for the mirrored pairing function to pair up storage units to provide the best protection possible from the hardware that is available. Your marketing representative will also help you plan for the new units needed to add for each ASP.

If you are planning to start mirrored protection on a new system, that system is already configured according to standard configuration rules. If you are using an older system, it may not follow the standard rules. However, wait until after attempting to start mirrored protection before reconfiguring any hardware.

Planning What ASPs to Create

Plan the user ASPs that will have mirrored protection and determine what units to add to the ASPs. When selecting units to add to the configuration, consider reviewing "Mirrored Protection Configuration Rules" on page 15-1. Also review "Creating a User ASP and Adding Disk Units to the New User ASP" on page 7-4.

In general, the units in an ASP should be balanced across several I/O processors, rather than all being attached to the same I/O processor. This provides better protection and performance.

Step 9. Installing Your New Hardware

When the hardware arrives, your service representative will install the hardware. After the hardware is installed, continue with Chapter 15, "Setting Up Mirrored Protection" on page 15-1 for information on how to add new units and start mirrored protection.

Chapter 15. Setting Up Mirrored Protection

This chapter provides information about mirrored protection configuration rules and the procedures to start mirrored protection.

Mirrored Protection Configuration Rules

The following rules apply when setting up a mirrored protection configuration:

1. Mirrored protection can be started through dedicated service tools (DST), not through system service tools (SST).
2. Mirrored protection is configured by ASP number. The two units of a mirrored pair are configured by the system within an ASP.
3. Mirrored protection requires an even number of storage units for each type of disk unit in the ASP being mirrored. An uneven number of storage units for any type of disk unit prevents mirrored protection from starting.
4. Two storage units cannot be mirrored if the failure of one unit can cause a data loss on the other unit. For the following current types of disk units, the two units of a mirrored pair are not allowed within the same disk enclosure:
   - 9332 model 400
   - 9332 model 600
   - 9335 model B01

   Additionally, the system attempts to assign the two storage units of a mirrored pair in such a way that if one fails, it can be repaired while the system continues to use the other mirrored unit. For a hardware configuration where this is not possible, the repair of the failing unit must be delayed until the system can be powered down. This is always true on the 9404 and 9402 system units. It may also be true for a failing mirrored unit sharing the same controller or I/O processor as its mirrored unit.
5. The mirrored units for unit 1 of the system ASP can only be at specific input and output addresses on the system. Therefore, the system attempts to assign the mirrored units for unit 1 of the system ASP first. Mirrored protection does not start if valid mirrored units for unit 1 cannot be found.

The following rules apply to Version 1 hardware

For 9406 system units:

1. Both storage units of the mirrored pair for unit 1 of the system ASP must be on Bus 0.
2. Both storage units of the mirrored pair for unit 1 of the system ASP must be under I/O processors in logical card slot 1 or 2.
3. If you have the new service processor card, then unit 1 can be attached to the I/O processor attached to logical card slot 1, 2, or 3.

System Model	Service Processor Card Number	Feature Number
B50	21F8195	2505
B60	21F8195	2505
B70	21F8197	2521

4. If the input/output (I/O) processor in logical card slot 1 is not a 6110, 6111, or 6112; or if the I/O processors in logical card slots 1 and 2 control disk units of different types, then the two storage units of the mirrored pair for unit 1 are under the same I/O processor. They are not protected against an I/O processor failure, and concurrent disk maintenance is not always possible.
5. Both storage units of the mirrored pair for unit 1 must be on controllers at address 0 or 1, under their respective I/O processors.
6. The storage units of the mirrored pair for unit 1 may be at any device address under the appropriate I/O processors and controllers.

For 9402 and 9404 systems units:

1. These systems only use the disk unit at address 1 to perform an IPL. The other storage unit for unit 1 can be anywhere.
2. If unit 1 of the system ASP at address 1 fails, the system continues to run. In order for an IPL of the system to occur, the other mirrored unit of the mirrored pair for unit 1 must be moved to address 1.

The following applies to all system units with Version 2 hardware

Both storage units for unit 1 must be attached to the first I/O processor (service processor). Therefore, I/O processor-level protection is not possible.

Starting Mirrored Protection

Before you start mirrored protection for an ASP, the ASP must exist and must have sufficient storage units. If necessary, first create any new ASPs and add the non-configured storage units. You do not need to save and restore the system to start mirrored protection. It is assumed that your system backup is current. When you have completed these tasks, complete the following tasks:

Any new disk units installed on your system will be nonconfigured. Use the DST Work with ASP Configuration displays to add the selected new storage units to the ASPs to be mirrored. If you planned for spare storage units, leave those storage units in a nonconfigured status.

Task Overview

You will perform the following steps during this task

1. Access DST options
2. Display disk configuration
3. Add new units
4. Start Mirrored protection

Task 1. Access DST Options

To use DST, do the following:

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

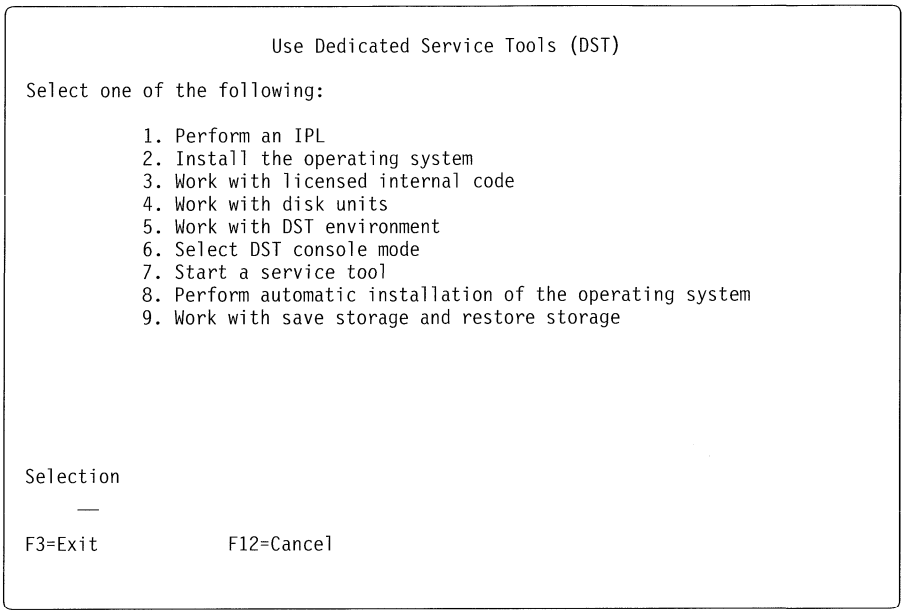
```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                  Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.


Use Dedicated Service Tools (DST)
Select one of the following:
1. Perform an IPL
2. Install the operating system
3. Work with licensed internal code
4. Work with disk units
5. Work with DST environment
6. Select DST console mode
7. Start a service tool
8. Perform automatic installation of the operating system
9. Work with save storage and restore storage
Selection
F3=Exit
F12=Cancel

Task 2. Display the Disk Configuration

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools (DST) menu.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display after the disk is attached.

```
                       Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 1 (Display disk configuration) on the Work with Disk Configuration display.

```
                          Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units
     5. Display device parity status
```

Note: Disk units on the system are either configured or nonconfigured. Options 1, 2, and 3 on the Display Disk Configuration display show information about the configured units. Option 4 shows the nonconfigured units attached to the system. When you physically attach a unit to the system, it becomes part of the nonconfigured set until you place it in an ASP. Option 5 shows the status of disk units that are using device parity protection.

4. Select option 1 (Display disk configuration status) and press the Enter key.

```
                    Display Disk Configuration Status

                Serial
ASP  Unit       Number      Type  Model  Address        Status
 1                                                     Unprotected
       1   10-00A7529  9332   400   0010-0000FFFF  Configured
       2   10-00A7529  9332   400   0010-0001FFFF  Configured
       3   10-00A4936  9332   400   0010-0100FFFF  Configured
       4   10-00A4936  9332   400   0010-0101FFFF  Configured
       5   10-00A7498  9332   400   0010-0300FFFF  Configured
       6   10-00A7498  9332   400   0010-0301FFFF  Configured
       7   10-00A7530  9332   400   0010-0400FFFF  Configured
       8   10-00A7530  9332   400   0010-0401FFFF  Configured




Press Enter to continue.

F3=Exit   F5=Refresh  F11=Display disk configuration capacity  F12=Cancel
```

The following fields appear on the Display Disk Configuration Status display:

- *ASP*. The auxiliary storage pool number.
- *Unit*. The number assigned by the system to identify a specific disk unit.
- *Serial Number*. The number assigned by the manufacturer to identify a specific disk unit.
- *Type*. The number assigned by the manufacturer to identify a type of disk unit.
- *Model*. The numbers or letters used to identify the feature level of a specific product type.
- *Address*. Identifies the following:
  - Location of the storage device controller card (columns 1 through 4)
  - Functional controller for the disk unit (columns 5 and 6)
  - Disk unit itself (columns 7 and 8)
  - FFFF (columns 9 through 12)
- *Status*. The valid values for this field are:
  - For ASPs:
    - *Unprotected*. No software protection exists for this ASP.
    - *Checksummed*. The units in the ASP are protected by checksum protection if the units are a part of a checksum set.
    - *Mirrored*. Some or all the units in the ASP are protected by mirrored protection. Any unit in a disk unit subsystem that has device parity protection does not participate in the mirrored protection provided by the software.

    Note: An ASP that is unprotected or has mirrored protection may contain units with device parity protection. The *Status* field for the unit shows if the ASP contains units with device parity protection. The *Status* field for the ASP shows the level of protection (unprotected or mirrored) for the ASP.

– For units in an unprotected ASP.

  - *Configured*.
  - *Device parity*.

    The valid status values for the storage units with device parity protection are:

    DPY/Active. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is fully operational.

    DPY/Failed. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has failed. If another unit in the disk unit subsystem fails, data could be lost.

    DPY/Rebuild. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The data on this unit is being rebuilt from other units in the disk unit subsystem. If another unit in the disk unit subsystem fails, data could be lost.

    DPY/Unprotected. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit is operational. However, another unit in the disk unit subsystem has failed or is being rebuilt. If another unit in the disk unit subsystem fails, data could be lost.

    DPY/HDW Failure. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A hardware-related failure has occurred. The failure does not affect data or performance. However, an exposure to an outage exists if another failure of a redundant component, such as a power supply, occurs.

    DPY/Degraded. Indicates that this unit is part of a disk unit subsystem that has device parity protection. A decrease in performance has occurred because a component that is not critical has failed. The failed component needs to be repaired or replaced.

    DPY/Power Loss. Indicates that this unit is part of a disk unit subsystem that has device parity protection. This unit has lost power.

    DPY/Not Ready. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The unit is not ready to perform I/O operations.

    DPY/Unknown. Indicates that this unit is part of a disk unit subsystem that has device parity protection. The status of this unit is not known to the system.

– For units in an ASP that has checksum protection.

  - *Checksummed*. Indicates that the unit is part of a checksum set.
  - *Configured*. Indicates that the unit is not part of a checksum set.

– For units in a mirrored ASP.

- *Active*. This unit is capable of having data written to it, or read from it.
- *Suspended*. This unit is not capable of having data written to it, or read from it. The data on this unit is not current. For example, if the disk needs repair action or has been manually suspended, it would be in a *Suspended* state.
- *Resuming*. The current data is being copied (or will be copied) to this unit from the other active unit of the mirrored pair.
- *Unknown*. The system configuration mechanism cannot determine what the valid configuration should be.

Note: Units in an ASP with mirrored protection can have device parity protection. However, these units do not participate in the mirrored protection provided by the system software.

5. Press F11 (Display non-configured unit) three times to display non-configured units.

```
                    Display Non-Configured Units

Serial
Number       Type   Model   Address         Status
10-00A7503   9332    400    0010-0100FFFF   Non-configured
10-00A7503   9332    400    0010-0101FFFF   Non-configured
10-00A3651   9332    400    0010-0400FFFF   Non-configured
10-00A3651   9332    400    0010-0401FFFF   Non-configured
```

6. Write down the serial number and address of the units that you are going to use.
7. Return to the Work with Disk Configuration display by pressing F12 (Cancel) two times.

```
                      Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

Task 3. Add Units to the ASP

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                       Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                          Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 4 (Add units to existing ASP) on the Work with ASP Configuration display.

```
                       Specify ASPs to Add Units to

Specify the ASP to add each unit to.

Specify  Serial
  ASP    Number      Type  Model  Address
   __    10-00A7503  9332    400  0010-0100FFFF
   __    10-00A7503  9332    400  0010-0101FFFF
   __    10-00A3651  9332    400  0010-0400FFFF
   __    10-00A3651  9332    400  0010-0401FFFF
```

5. Type the number of the ASP in the *Specify ASP* column to select the disk units to place in the specified ASP and press the Enter key.

   The Confirm Add Units display shows what the entire system configuration will be when you add the units. Use the serial number of the disk unit to verify that you are selecting the correct disk units to add to the ASP.

```
                              Confirm Add Units

Add will take several minutes for each unit.  The system will
have the displayed protection after the unit(s) are added.

Press F9=Capacity Information to display the resulting capacity.
Press Enter to confirm your choice for 1=Add units.
Press F12=Cancel to return and change your choice.


            Serial
 ASP  Unit  Number      Type  Model  Address         Protection       CSS
  2                                                 Unprotected
        9   10-00A3651  9332   400   0010-0400FFFF   Unprotected
       10   10-00A3651  9332   400   0010-0401FFFF   Unprotected
```

6. If you are satisfied with the configuration, press the Enter key to add the disk units to the ASP.

   Adding units takes several minutes. The time it takes depends on the size of each unit being added and the ability of the system to add more than one unit at the same time.

Task 4. Start Mirrored Protection

1. Return to the Work with Disk Configuration display.

```
                        Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

2. Select option 5 (Work with mirrored protection) on the Work with Disk Configuration display.

```
                    Work with Mirrored Protection

Select one of the following:

     1. Display disk configuration
     2. Start mirrored protection
     3. Stop mirrored protection
     4. Calculated mirrored capacity









Selection
     2

F3=Exit          F12=Cancel
```

3. Select option 2 (Start mirrored protection) on the Work with Mirror Protection display.

```
                  Select ASP to Start Mirrored Protection

Type options, press Enter
  1=Select

Option   ASP   Protection
  _        1   Unprotected
  _        2   Unprotected









F3=Exit          F12=Cancel
```

4. Select the ASP or ASPs to be mirrored on the Select ASP to Start Mirrored Protection display and press the Enter key.

   The system shows a confirmation display of the new mirrored protection configuration, including the levels of protection. Notice that half of the previous unit numbers for the ASP no longer exist. The storage units for those unit numbers have been paired with the storage units for the remaining unit numbers to make mirrored pairs.

```
                    Confirm Start Mirrored Protection

Press Enter to confirm your choice to start mirrored
  protection.  During this process the system performs an IPL.
  You will return to the DST main menu after the IPL is
  complete.  The system will have the displayed protection.
Press F12 to return to change your choice.


           Serial
ASP  Unit  Number      Type  Model  Address        Protection        CSS
 1                                                 Mirrored
      1    10-00A5736  9332   400   0020-0010FFFF  Controller
      1    10-00A5729  9332   400   0010-0000FFFF  Controller
      2    10-00A5736  9332   400   0010-0101FFFF  Controller
      2    10-00A5729  9332   400   0010-0001FFFF  Controller
 2                                                 Unprotected
      3    10-00A5703  9332   400   0010-0201FFFF  Configured
      4    10-00A7498  9332   400   0020-0101FFFF  Configured


                                                                  More...

F12=Cancel
```

Figure 15-1. Confirm Start Mirrored Protection

5. If the configuration is what you had planned and you do not have other configuration changes to make, turn the keylock switch to the **Normal** position and press the Enter key to accept the configuration.

   The system continues the start mirrored protection process in "Start Mirrored Protection Processing" without further operator intervention.

   If the configuration is not what you had planned, for example, the level of protection is less, you have the following options:

   - Verify that the correct ASP was selected. Verify that any new storage units have been added to the correct ASP.
   - Determine if additional hardware is required to achieve the planned level of protection. Review "Step 5: Determining the Extra Hardware You Need for Mirroring" on page 14-5. Contact your IBM marketing representative for assistance.
   - Determine if the existing hardware needs to be connected differently to achieve the planned level of protection. Review "Step 8. Planning Your Installation" on page 14-7 and refer to "Reconfiguring Your System" on page 17-2 for information on how to achieve your planned level of protection. Contact your IBM marketing representative for assistance.
   - Consider continuing the start mirrored protection process which will provide better availability than non-mirrored protection, rather than waiting until additional hardware arrives so that you can achieve your planned level of protection. After you receive and install the additional hardware, do a Stop Mirrored Protection procedure followed by a Start Mirrored Protection procedure to achieve the best level of protection for the new hardware configuration. Even on very large systems, the tasks to stop mirroring, add units, and start mirrored protection can be done in several hours.

6. After the system reaches the Command Entry display, create the QSYSMSG message queue to receive messages. See "Message Handling" on page 16-23 for more information.

Start Mirrored Protection Processing

The following steps are performed by the system when mirrored protection is started.

1. Data is moved off half of the storage units in the selected ASPs. This can take from a few minutes to a few hours, depending on the amount of data that must be moved.

   Objects created on a preferred unit may be moved to another unit. The preferred unit number may no longer exist when mirrored protection is started.

2. New control information is written to disk, describing the new mirrored system configuration.
3. After the data is moved and the control information is written, the system performs an IPL.
4. When the system reaches DST, the previously selected ASPs are mirrored, although the two storage units in the mirrored pairs are not yet synchronized.

   If the keylock switch is in the Manual position, you have the option to perform other configuration changes or perform an IPL. If you do not have configuration changes to make, select the option to perform an IPL and press the Enter key.

   If the keylock switch is in the Normal position, the system automatically continues the IPL.

5. When the system continues the IPL past DST, the mirrored pairs are synchronized during storage management recovery. This can take a few hours, although this long recovery time only occurs when mirrored protection is first started, and not during every IPL on a mirrored system. On very large systems, the entire start mirrored protection process can take up to approximately eight to ten hours.
6. After storage management recovery is completed, the selected ASPs have mirrored protection. If the keylock switch is in the Normal position, the command entry display is shown. If the keylock switch is in the Manual position, the Sign On display is shown.

Mirrored Protection Configuration Errors

Starting mirrored protection requires that the storage management directories are good and there are no missing or suspended mirrored units in the configuration. You must do the following before mirrored protection can start:

- Units with a status of missing, must be powered on, repaired, or replaced.
- Units with a status of suspended in a mirrored ASP need to be repaired or replaced, and resumed before mirrored protection can start in another ASP.
- If you have stopped checksum protection on an ASP, and you want to start mirrored protection on the same ASP, you must IPL the system through storage management recovery. You cannot start mirrored protection on an ASP that has checksum protection.

Start mirrored protection can fail if there is insufficient storage available in the ASP to contain the current data in the ASP. The percentage used in the ASP must be less than half of the ASP threshold.

There must be sufficient storage units in the ASP for the system to create mirrored pairs. If you receive a message indicating that the system cannot pair unit 1 or other units, review "Mirrored Protection Configuration Rules" on page 15-1.

Chapter 16. Managing the Mirrored Environment

This chapter provides information about managing and recovering the mirrored environment on your system. SST and DST are the tools which provide menu options for you to display, change, or recover the mirrored environment. For more information about these options, see "Overview of SST and DST Options" on page 7-1.

Management Options

The following configuration changes are only allowed when all currently configured disk units are present and are not suspended. However, units that are being made nonconfigured may be suspended. All remaining units must be present and not suspended.

- Add units
- Start mirrored protection
- Start checksum protection
- Stop checksum protection
- Move units
- Replace units

The following configuration changes are allowed when the units that are being made nonconfigured units are missing or suspended.

- Stop mirrored protection
- Remove units

Adding Disk Units to a Mirrored ASP

You can add storage units to an ASP while mirrored protection is in effect to increase the storage capacity of that ASP. When mirrored protection is in effect, you must add an even number of each type of storage unit. The number of mirrored pairs added are half the number of storage units added.

Note: If the disk unit subsystems with device parity protection are being added to an ASP with mirrored protection, the disk units in the disk unit subsystems do not participate in the pairing configuration. These disk units are protected by the disk unit subsystem with device parity protection.

It is not necessary to perform a save and restore operation when adding units to a mirrored ASP. You should have a current backup copy of your system.

The following apply to adding units:

- Adding units is only allowed using DST, before an IPL.
- All configured storage units must be operational.
- The new units are automatically paired with one another or with existing units as part of the add operation to provide the best level of protection possible.
- Adding units is not allowed if the unit 1 has been saved by the DST Save Disk Unit Data function. This is because a restore of the saved information would clear the configuration change.

- An even number of storage units of each type are required.

Adding a mirrored unit takes at least twice as long as adding a unit that is not mirrored because each unit is a pair of mirrored units. Later, during the IPL, the added units are also synchronized. This can take up to 30 minutes for each mirrored unit.

Adding a mirrored unit can result in a lower level of protection. The Confirm Add Units display shows the level of protection that results from the add operation.

The following assumes that the service representative has attached the new disk units and they are in nonconfigured status.

Task 1. Calculate Mirrored Capacity Using SST

Use the SST Calculate Mirrored Capacity displays to determine if the new disk units can provide the required capacity.

1. Type the following on the command line and press the Enter key.

   ```
   STRSST
   ```

   The System Service Tools (SST) menu is shown.

2. Select option 3 (Work with disk units) on the System Service Tools (SST) menu and press the Enter key.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

3. Select option 7 (Calculate mirrored capacity) on the Work with Disk Units display.

   The Specify ASP to Calculate Mirrored Capacity display is shown.

```
                Specify ASP to Calculate Mirrored Capacity

Type choices, press Enter.

  ASP . . . . . . . . . . . . . .  _    1-16
```

4. Type the number of the ASP in the *ASP* prompt and press the Enter key.

   The Calculate Mirrored Capacity display is shown.

```
                    Calculate Mirrored Capacity

ASP . . . . . . . . . . . . . . :           2

Type choices, press Enter.

Specify required size . . . . . . . . .   1000       Megabytes

 -------- Number of Units ------
   Planned            Current      Type     Model        Size
       0                  0        6100      015          315
       0                  0        6105      010          320
       8                  0        6105      020          320
       0                  0        6105      030          320
       0                  0        6107      040          320
       0                  0        6107      010          400
       0                  0        6107      020          400
       0                  0        6107      030          400
       0                  0        6107      040          400


F3=Exit   F11=Display configuration capacity    F12=Cancel
```

5. Type the required size of the ASP in the *Specify required size* field if the ASP does not exist.
6. Type the number of planned storage units in the *Planned* prompt in the *Number of Units* column and press the Enter key.

   The Capacity Provided by Mirrored Protection display is shown.

```
                Capacity Provided By Mirrored Protection

ASP . . . . . . . . . . . . . . :          2
Required size . . . . . . . . . :       1000   Megabytes
Actual size   . . . . . . . . . :       1280   Megabytes
Planned Amount used. . . . . . .:         78   %
 Specify required size . . . . . . . . .     _____       Megabytes

  -------- Number of Units ------
    Planned            Current      Type     Model        Size
        0                  0        6100      015
        0                  0        6105      010
        8                  0        6105      020
        0                  0        6105      030
        0                  0        6107      040
        0                  0        6107      010
        0                  0        6107      020
        0                  0        6107      030
        0                  0        6107      040


F3=Exit   F11=Display configuration capacity    F12=Cancel
```

7. Exit SST by pressing F3 (Exit) three times.
8. On the Exit System Service Tools display, press the Enter key to end SST.

Task 2. Access DST Options

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                  Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .   _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit           F12=Cancel
```

Task 3. Adding New Units

After the disk units are attached, they are shown as nonconfigured units on the DST Display Nonconfigured Units display.

When selecting disk units to place in an ASP that has mirrored protection, you need to select an even number of units of each type.

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units display.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 4 (Add units to existing ASP) on the Work with ASP Configuration display.

```
                      Specify ASPs to Add Units to

Specify the ASP to add each unit to.

Specify  Serial
  ASP    Number       Type  Model  Address
   __    10-00A7503   9332    400    0010-0100FFFF
   __    10-00A7503   9332    400    0010-0101FFFF
   __    10-00A3651   9332    400    0010-0400FFFF
   __    10-00A3651   9332    400    0010-0401FFFF
```

5. Type the number of the ASP in the *Specify ASP* column to select the disk units to place in the specified ASP and press the Enter key.

   The Confirm Add Units display shows what the entire system configuration will be when you add the units. Use the serial number of the disk unit to verify that you are selecting the correct disk units to add to the ASP.

```
                              Confirm Add Units

Add will take several minutes for each unit.  The system will
have the displayed protection after the unit(s) are added.

Press F9=Capacity Information to display the resulting capacity.
Press Enter to confirm your choice for 1=Add units.
Press F12=Cancel to return and change your choice.


            Serial
 ASP  Unit  Number      Type  Model  Address         Protection       CSS
  2                                                  Unprotected
        9   10-00A3651  9332   400   0010-0400FFFF   Unprotected
       10   10-00A3651  9332   400   0010-0401FFFF   Unprotected
```

6. If you are satisfied with the configuration, press the Enter key to add the disk units to the ASP.

   Adding units takes several minutes. The time it takes depends on the size of each unit being added and the ability of the system to add more than one unit at the same time.

Moving a Disk Unit from a Mirrored ASP to Another ASP

Units cannot be moved to, or from, a mirrored ASP using the DST move function. When there are mirrored ASPs, units must first be removed from one ASP and then added to another ASP, using the remove function and then the add function.

Removing Units from an ASP that Has Mirrored Protection

When a unit is removed from an ASP that has mirrored protection, both mirrored units of that mirrored pair are removed. If you select only one unit in a mirrored pair, the system automatically selects the other unit in the mirrored pair.

Task Overview

You will perform the following steps during this procedure:

1. Access DST
2. Remove the unit from the system ASP
3. Perform an IPL

Warning: A sufficient number of 2800 or 2801 storage units must be configured to the system ASP (ASP 1) to allow for main storage dump space. (See note 3 on page 6-10.)

Before you begin...

You cannot remove unit 1 from the system ASP. (Unit 1 is reserved for the system licensed internal code.)

In the following example, a unit currently in the an ASP is being removed.

Do the following tasks to remove a disk unit:

Task 1. Access DST Options

Sign on to DST using the following steps:

1. Notify the users to sign off the system by sending a break message.
2. Change the QSYSOPR message queue to break mode:

   ```
   CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```

3. End all subsystems:

   ```
   ENDSBS SBS(*ALL) OPTION(*IMMED)
   ```

   Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.
4. Ensure the key is in the keylock switch on the control panel.
5. Turn the key until it points to the Manual position.
6. Power down the system:

   ```
   PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
   ```

7. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

8. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                    Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

9. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

   The Use Dedicated Service Tools (DST) menu is shown.

```
                         Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
    —

F3=Exit            F12=Cancel
```

Task 2. Remove the Unit the ASP

1. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu.

```
                              Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

2. Select option 1 (Work with disk configuration) on the Work with Disk Units Display and press the Enter key.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

3. Select option 3 (Work with ASP configuration) on the Work with Disk Configuration display.

```
                         Work with ASP Configuration

Select one of the following:

     1. Display disk configuration capacity
     2. Create user ASP
     3. Delete user ASP
     4. Add units to existing ASP
     5. Delete ASP data
     6. Change ASP storage threshold
     7. Move units from one ASP to another
     8. Remove units from configuration
```

4. Select option 8 (Remove units from configuration) on the Work with ASP Configuration display.

   Note: If any user ASP has overflowed into the system ASP, reset the ASP before removing the units. For more information on how to reset the ASP, see "Considerations for Recovering an Overflowed User ASP" on page 7-75.

```
                        Remove Units from Configuration

Type options, press Enter.
   4=Remove unit from the configuration

                Serial
OPT  Unit  ASP  Number  Type  Model  Address        Status
       1    1          9332   400   0010-0000FFFF  Mirrored
       1    1          9332   400   0010-0100FFFF  Mirrored
       2    1          9332   400   0010-0300FFFF  Mirrored
       2    1          9332   400   0010-0001FFFF  Mirrored
       3    1          9332   400   0010-0600FFFF  Mirrored
       3    1          9332   400   0010-0101FFFF  Mirrored
       7    2          9332   200   0010-0201FFFF  Configured
       8    3          9332   200   0020-0701FFFF  Configured
       9    4          9332   200   0010-0700FFFF  Configured

                                                              More...
Press Enter to continue.

F3=Exit   F5=Refresh    F11=Display non-configured units    F12=Cancel
```

5. Type a 4 (Remove unit from configuration) in the *OPT* column for each unit you want to remove and press the Enter key.

   Note: More than one unit can be removed at the same time.

   The Confirm Continuation display may be shown before Confirm Remove of Unit display if the ASP has mirrored protection or the storage management directories are not usable.

```
                              Confirm Continuation

In order to proceed the system must perform internal processing
that may take several minutes during which the system may appear
inactive.  Once the processing starts only the Work with Disk
Configuration operation will be available until an IPL of
the system to DST is performed.

Press Enter to continue.
Press F12=Cancel to return and change your choice.
```

```
                           Confirm Remove of Units

Removing units will take several minutes.

Press Enter to confirm remove of disk units.
Press F9=Resulting capacity to display the capacity information.
Press F12=Cancel to return to change your choice.

                 Serial
 Opt  Unit  ASP  Number  Type  Model  Address        Status
        3    1           9332   400   0010-0101FFFF  Mirrored
        3    1           9332   400   0010-0600FFFF  Mirrored
```

6. Press the Enter key.
7. When the remove operation is complete, you return to the Work with ASP Configuration display.
8. Press F3 (Exit). The Confirm IPL display is shown.
9. Press the Enter key to start the IPL.

Stopping Mirrored Protection

When mirrored protection is ended, mirrored units are unpaired and half of the storage units become nonconfigured units. All units in the ASPs that do not have mirrored protection ended must be present and not suspended before mirrored protection can be ended. To control which mirrored unit of each pair becomes nonconfigured, you may suspend the storage units that you want to become nonconfigured. For units that are not suspended, the selection is automatic. An attempt is made to keep entire disk enclosures either configured or nonconfigured. Mirrored protection can be ended using the DST option before the IPL to the OS/400 licensed program.

To end mirrored protection, do the following:

1. Access DST options.
   a. Notify the users to sign off the system by sending a break message.

b. Change the QSYSOPR message queue to break mode:

```
CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
```

c. End all subsystems:

```
ENDSBS SBS(*ALL) OPTION(*IMMED)
```

Wait until a message is sent to the QSYSOPR message queue indicating that all subsystems have ended and the system is in a restricted state.

d. Ensure the key is in the keylock switch on the control panel.

e. Turn the key until it points to the Manual position.

f. Power down the system:

```
PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
```

g. When the system has powered down and then powered back up, the IPL or Install the System display appears.

```
                        IPL or Install the System

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Use Dedicated Service Tools (DST)
          4. Perform automatic installation of the operating system
```

h. Select option 3 (Use Dedicated Service Tools (DST)) on the IPL or Install the System menu and press the Enter key. The Dedicated Service Tools (DST) Sign On display is shown.

```
                  Dedicated Service Tools (DST) Sign On

Type choice, press Enter.

  DST password . . . . . .  _____
```

i. Sign on DST with the DST security-level or full-level password. The *Security Reference* has more information about DST passwords.

The Use Dedicated Service Tools (DST) menu is shown.

```
                    Use Dedicated Service Tools (DST)

Select one of the following:

          1. Perform an IPL
          2. Install the operating system
          3. Work with licensed internal code
          4. Work with disk units
          5. Work with DST environment
          6. Select DST console mode
          7. Start a service tool
          8. Perform automatic installation of the operating system
          9. Work with save storage and restore storage




Selection
     —

F3=Exit           F12=Cancel
```

2. Select option 4 (Work with disk units) on the Use Dedicated Service Tools menu and press the Enter key.

```
                          Work with Disk Units

Select one of the following:

     1. Work with disk configuration
     2. Analyze disk device problem
     3. Work with disk unit recovery
     4. Work with disk unit information
```

3. Select option 1 (Work with disk configuration) on the Work with Disk Units display and press the Entry key.

```
                    Work with Disk Configuration

Select one of the following:

     1. Display disk configuration
     2. Work with ASP threshold
     3. Work with ASP configuration
     4. Work with checksum protection
     5. Work with mirrored protection
     6. Work with device parity protection
```

4. Select option 5 (Work with mirrored protection) on the Work with Disk Configuration display and press the Enter key.

```
                    Work with Mirrored Protection

Select one of the following:

     1. Display disk configuration
     2. Start mirrored protection
     3. Stop mirrored protection
     4. Calculated mirrored capacity
```

5. Select option 3 (Stop mirrored protection) on the Work with Mirror Protection display.

```
                    Select ASP to Stop Mirrored Protection

Type options, press Enter
  1=Select

Option   ASP   Protection
           1   Mirrored
           2   Mirrored
```

6. Select the ASP or ASPs for which mirrored protection is to be stopped on the Select ASP to Stop Mirrored Protection display and press the Enter key.
7. The Confirm Stop Mirrored Protection display is shown. Press the Enter key to confirm your choice.

```
                    Confirm Stop Mirrored Protection

Press Enter to confirm your choice to stop mirrored
  protection.  During this process the system will be IPLed.
  You will return to the DST main menu after the IPL is
  complete.  The system will have the displayed protection.
Press F12 to return to change your choice.


            Serial
ASP  Unit   Number      Type  Model  Address         Protection        CSS
 1                                                  Unprotected
       1    57-1253342  9335   B01   0010-0101FFFF   Unprotected
       3    57-1225682  9335   B01   0020-0201FFFF   Unprotected
       4    57-1107344  9335   B01   0010-0401FFFF   Unprotected
 2                                                  Mirrored
       5    10-00A5703  9332   400   0010-0101FFFF   Active
       5    10-00A4293  9332   400   0020-0201FFFF   Active


                                                                  More...

F12=Cancel
```

Figure 16-1. Confirm Stop Mirrored Protection

8. If you have no other DST functions to perform, return to the Use Dedicated Service Tools menu.

9. Select option 1 (Perform an IPL) on the Use Dedicated Service Tools menu and press the Enter key. The IPL or Install the System menu is shown.
10. Select option 1 (Perform an IPL) and press the Enter.

Mirrored Protection Recovery Actions

In considering aspects of recovery, you need to distinguish between *errors* and *failures* in the disk subsystem.

A disk *error* refers to an unexpected event during an I/O operation which can cause the loss or corruption of the data being transferred. Most disk errors are hardware related, caused by a failure in some part of the component chain from the I/O processor to the disk surface. Environmental effects such as power abnormalities or severe electrostatic discharges can also cause disk errors. Included in the definition of disk errors is a failure of the licensed internal code that controls the disk subsystem.

When the system detects an error, generally the occurrence is logged and the operation is attempted again. Temporary errors are those from which the system can recover and complete the I/O operation successfully. When the error is so severe that the I/O operation cannot succeed, it is a permanent error.

When the system detects a permanent error, it classifies it as a *failure* in that hardware subsystem. In an ASP that does not have mirrored protection, a failure causes the system to become unusable.

On a system with mirrored protection, errors and failures have different effects.

In "Level of Protection" on page 13-3, the concept of level of protection was introduced. When a failure occurs on a system with mirrored protection, the recovery procedure is affected by the level of protection configured. Because of the hardware differences between the 9406 and the other system units, they will be discussed separately. There are variations between the 9402 or 9404 system unit and the 9406 system unit in the handling of the IPL process.

9402 and 9404 System Units

The following considerations apply to 9402 and 9404 system units:

- Disk units cannot be repaired when the system is powered on. If spare units are available, the SST Replace function may be used to replace a failed disk unit with a spare while the system is running.
- There is a special consideration for unit 1 if it fails. Rule 5 on page 15-1 has more information about restrictions for unit 1.

  Note: This restriction does not apply for the type 2615 service processor on the 9404 system unit.

9406 System Units

The following considerations apply to 9406 system units:

- Bus-level protection is possible only on AS/400 models with two or three busses. Because system hardware requires that unit 1 be on bus 0, the two storage units of the mirrored pair for unit 1 must be on the same bus. Therefore, bus-level protection is not possible for unit 1. A failure of bus 0 always results in a system failure.
- When a bus fails, a disk unit that is mirrored on another bus continues to operate. However, it must be understood that a bus failure can cause other subsystems to fail. A failure of active work stations or communications lines limits the availability provided by mirrored protection. The system must be powered down to repair a bus.
- When an I/O processor fails, there are two possibilities. If there are any configured units attached to the I/O processor that have mirrored protection under the same I/O processor or that are not mirrored, then the system becomes unusable. If the remaining half of each mirrored pair is active, the system suspends all mirrored units under the failed I/O processor and continues operation. A failed I/O processor can be replaced only when the system power is off. After the failed I/O processor is replaced, the system synchronizes the suspended mirrored units during the next IPL.
- It is possible that disk unit temporary errors will be logged with increasing frequency well before the time of a failure. When a disk unit shows these symptoms, you may decide to repair it. If the disk unit contains multiple storage units, all storage units in the disk unit must be suspended before the disk unit can be repaired.
- The 9332 model 400, 9332 model 600, and the 9335 model B01 disk units on the 9406 system unit have two storage units in a single disk unit. It is possible for only one of the storage units in a disk unit to fail. It is recommended that the other storage unit be suspended before the disk unit is powered down.
- During some parts of the IPL process, the system cannot use the mirrored unit for unit 1. If unit 1 fails during this time, the IPL is tried again from the mirrored unit for unit 1. If this is unsuccessful, one of the units in the mirrored pair may be manually powered down or physically removed, and an IPL may be tried again to circumvent the problem.

  Note: The 2800 disk units do not have power off switches. You may be able to switch the book adapter cards.

Suspending or Resuming Mirrored Units

If you have to suspend or resume a mirrored unit, you can do so using the Suspend/Resume Mirrored Protection option on the Work With Disk Unit Recovery display using SST or DST.

There can be more than one unit with the status of *suspended* or *resuming status* on the system at the same time. However, only one unit at a time can have its status changed. If you suspended both units in a 9332 model 400 disk unit, both storage units must be resumed before full protection is started. For a unit to be resumed, it must be selected individually. However, you do not need to wait until the status of the unit is *active* before selecting the next unit.

1. Type the following to suspend or resume mirrored protection using SST options:

   STRSST

2. Select option 3 (Work with disk units) on the System Service Tools (SST) menu and press the Enter key.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

3. Select option 5 (Work with disk unit recovery) on the Work with Disk Units display and press the Enter key.

```
                  Work with Disk Unit Recovery

Select one of the following:

     1. Replace configured unit
     2. Disk problem recovery procedures
     3. Suspend/resume mirrored protection
     4. Delete disk unit data
```

4. Select option 3 (Suspend/resume mirrored protection) on the Work with Disk Unit Recovery display and press the Enter key.

```
                    Suspend/Resume Mirrored Protection

Type option, press Enter.
  1=Suspend Protection        2=Resume Protection

                    Serial
Option  Unit  ASP   Number      Type  Model  Address        Status
  _       1   1     57-00B57BE  9335   B01   0020-0001FFFF  Active
  _       1   1     57-00B5CF7  9335   B01   0020-0100FFFF  Active
  _       2   1     57-00B8BB7  9335   B01   0120-0002FFFF  Active
  _       2   1     57-00B6F1D  9335   B01   0110-0100FFFF  Active
```

5. Type a 1 (Suspend Protection) or a *2* (Resume Protection) in the *Option* column for each unit that you want to suspend or resume mirrored protection. You can suspend protection only on units that have both units in an *Active* or *Resuming* status. If one of the units is in *Resuming* status, then it is the only unit that can be selected to suspend. You can select only a unit in *Suspending* status to resume.

Replacing a Mirrored Unit

A unit selected to replace the failed mirrored unit must satisfy all of the mirrored protection configuration rules and restrictions when it is paired with the remaining unit in the mirrored pair. (See "Mirrored Protection Configuration Rules" on page 15-1.)

You can replace mirrored units using the Replace Disk Unit option in either DST or SST. To do this, you need to have a spare storage unit that can be paired with the mirrored unit of the storage unit being replaced. The storage unit being replaced can have a status of active or suspended. However, one of the storage units in the pair must be suspended. The results of the replace operation is different for each status. Replacing a suspended storage unit causes that storage unit to go to a status of resuming after the replace operation. Replacing an active unit causes the ASP to be cleared. The storage unit being replaced can also be missing or not missing. To replace a unit with a status of resuming, you must suspend it. If the status of unit 1 is unknown, replace operations are not allowed until the status of the mirrored units for unit 1 is known. The unit selected to replace another mirrored unit must satisfy all of the mirrored protection configuration rules and restrictions when it is paired with the remaining unit in the mirrored pair. (See "Mirrored Protection Configuration Rules" on page 15-1.)

If a storage unit fails, and if the same storage unit that failed has been repaired, it is not necessary to replace it. The failed disk should have a status of suspended.

If the storage unit being replaced is active, it can only be replaced at DST before the IPL to the OS/400 licensed program. It should never be necessary to replace an active unit unless both units of the mirrored pair have failed. If this situation does occur, the service representative should first try to recover the data from the failed units using the Save Disk Unit Data option on the Work with Disk Unit Recovery display. When an active unit is replaced, the last good copy of the data has been lost and a warning display is shown. If the replacement continues, the ASP is cleared as if the replace option was used on an unprotected ASP.

Replacing unit 1 requires special handling. If the system ASP has mirrored protection, one of the units in the mirrored pair for unit 1 is selected as the IPL device. It is the only unit that is used until the system performs an IPL to the OS/400 licensed program. Until then, it cannot be replaced or even suspended. However, its mirrored unit can be both suspended and replaced. After the IPL to the OS/400 licensed program, the IPL device can be suspended and then replaced.

Replacing a unit may cause the level of protection for a mirrored pair to change. If a lower level of protection results from a replacement operation, a warning screen is displayed. At certain times, especially when missing units are involved in the replace operation, the system may not be able to accurately calculate the level of protection and the same warning display is shown.

To replace a disk unit using SST, do the following:

1. Type the following to access SST:

   ```
   STRSST
   ```

2. Select option 3 (Work with disk units) on the System Service Tools (SST) menu and press the Enter key.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

3. Select option 5 (Work with disk unit recovery) on the Work with Disk Units display and press the Enter key.

```
                  Work with Disk Unit Recovery

 Select one of the following:

      1. Replace configured unit
      2. Disk problem recovery procedures
      3. Suspend/resume mirrored protection
      4. Delete disk unit data
```

4. Select option 1 (Replace configured unit) on the Work with Disk Unit Recovery display and press the Enter key.

   The Select Configured Unit to Replace display is shown.

```
                       Select Configured Unit to Replace

 Type option, press Enter.
   1=Select

                    Serial
 Option  Unit  ASP  Number      Type  Model  Address           Status
   1       3    2  10-0122875  6100   015   0110-0300FFFF     Active
           3    2  10-0123972  6100   015   0110-0200FFFF     Suspended
```

5. Type a 1 in the *Option* column on the Select Configured Unit to Replace display and press the Enter key.

```
                       Select Configured Unit to Replace

            Serial
 Unit  ASP  Number      Type  Model  Address           Status
  3     2   10-0122875  6100   015   0110-0300FFFF     Suspended
 Type option, press Enter.
   1=Select

                    Serial
 Option  Unit  ASP  Number      Type  Model  Address           Status
   1    10-0124597  6100   015   0110-0301FFFF     Non-configured
        10-0126894  6100   015   0110-0303FFFF     Non-configured
```

6. Type a 1 in the *Option* column on the Select Configured Unit to Replace display and press the Enter key.

```
                    Confirm Replace of Configured Unit

Warning: Replacing a unit will destroy all the data from the
ASP that contained the unit being replaced.

ASP that will have its data destroyed . . . . . .        2

Press F10 to confirm your choice for 1-Replace
Press F12 to return and change your choice.

                    Serial
 OPT  Unit  ASP  Number      Type  Model  Address            Status
   1   3     2   10-0124597  6100   015   0110-0301FFFF      Active
```

7. Press F10 (Confirm) or F12 (Cancel).
8. The replacement function runs for several minutes. Wait until the replacement function completes.

Using Spare Nonconfigured Units for Replacement

If mirrored units become suspended as a result of a hardware failure, the system continues to run. However, one or more storage units will be suspended and therefore unprotected until your service representative can repair or replace the failed hardware. You may be able to resume mirrored protection before the repair actions are done if you have spare nonconfigured units.

Call your service representative. You will be directed to run problem analysis on those units that are suspended. You can determine what units are suspended by using the Display Disk Configuration Status option using SST or the Work with Disk Status (WRKDSKSTS) command. If all disk units under an I/O processor are suspended, the I/O processor probably has failed. If you have enough spare units of the right type and model, and if the spare units are not on the I/O processor that has failed, you may be able to use the spare nonconfigured units to resume mirrored protection.

After your service representative repairs a failed storage unit, you may want to use it instead of the spare to restore the previous level of protection. To use the repaired unit, do the following:

1. Suspend the active storage unit that was previously used as the spare by typing the following on a command line and pressing the Enter key.

   ```
   STRSST
   ```

   The System Service Tools (SST) menu is shown.
2. Select option 3 (Work with disk units) on the System Service Tools menu and press the Enter key.

```
                          Work with Disk Units

Select one of the following:

    1.  Display disk configuration
    2.  Display checksum configuration
    3.  Calculate checksum configuration
    4.  Work with ASP storage threshold
    5.  Work with disk unit recovery
    6.  Work with disk information
    7.  Calculate mirrored capacity
```

3. Select option 5 (Work with disk unit recovery).

```
                  Work with Disk Unit Recovery

 Select one of the following:

      1. Replace configured unit
      2. Disk problem recovery procedures
      3. Suspend/resume mirrored protection
      4. Delete disk unit data
```

4. Select the option 3 (Suspend/resume mirrored protection).

```
                      Suspend/Resume Mirrored Protection

 Type option, press Enter.
   1=Suspend Protection        2=Resume Protection

                      Serial
 Option  Unit  ASP    Number      Type  Model  Address        Status
   _        1   1     57-00B57BE  9335   B01   0020-0001FFFF  Active
   _        1   1     57-00B5CF7  9335   B01   0020-0100FFFF  Active
   _        2   1     57-00B8BB7  9335   B01   0120-0002FFFF  Active
   _        2   1     57-00B6F1D  9335   B01   0110-0100FFFF  Active
   _
```

5. Type a 1 (Suspend Protection) in the *Option* column. The original spare unit is the same disk type and model as the repaired disk unit.
6. Return to the Work with Disk Unit Recovery display by pressing F12 (Cancel)

```
                  Work with Disk Unit Recovery

 Select one of the following:

      1. Replace configured unit
      2. Disk problem recovery procedures
      3. Suspend/resume mirrored protection
      4. Delete disk unit data
```

7. Select option 1 (Replace configured unit).

```
                    Select Configured Unit to Replace

Type option, press Enter.
  1=Select

                  Serial
Option  Unit  ASP Number      Type  Model  Address           Status
  1      3     2  10-0122875 6100   015   0110-0300FFFF     Active
         3     2  10-0123972 6100   015   0110-0200FFFF     Suspended



                    Select Configured Unit to Replace

Type options, press Enter.
  1=Select
```

8. Type a 1 in the *Option* column on the Select Configured Unit to Replace display and press the Enter key.

```
                    Select Configured Unit to Replace

          Serial
Unit  ASP Number      Type  Model  Address           Status
 3     2  10-0122875 6100   015   0110-0300FFFF     Suspended
 Type option, press Enter.
   1=Select

                 Serial
Option  Unit  ASP Number      Type  Model  Address           Status
  1    10-0124597 6100   015   0110-0301FFFF     Non-configured
       10-0126894 6100   015   0110-0303FFFF     Non-configured
```

9. The replacement function runs for several minutes. Wait until the replacement function completes.

Mirrored Protection Recovery Actions Performed by the Service Representative

The procedures described here are overviews of the steps and considerations involved in disk unit repair in the mirrored environment. Although these steps are performed by your service representative, they are included here for your information.

The problem analysis procedure suspends mirrored protection on any storage units required to perform service on the reported failure. After the repair action, the problem analysis procedure resumes the suspended storage units.

9402 and 9404 System Units:

1. The system must be powered down to repair the failing storage unit.
2. If unit 1 at the IPL address has failed, see 5 on page 15-1 for restrictions that apply.
3. An attended IPL of the system is performed to bring up the DST menu.
4. If the Replace configured unit option is necessary, the new storage unit is formatted and initialized.
5. If the disk configuration status is displayed, the failed disk unit is in suspended status.

6. The IPL completes to normal operations.

9406 System Units:

1. If there is a working storage unit in the disk unit to be repaired, it is suspended by the PAR procedure.
2. Power down the disk unit.
3. Repair the failing unit.

Other Recovery Considerations for Mirrored Protection

Message Handling: When a system with mirrored protection experiences a disk failure, the only external indication of the failure is a message sent to the system operator message queue (QSYSOPR). If there is a message queue named QSYSMSG in the QSYS library, the messages are sent there also.

When suspended units exist, the system sends a message every hour to the QSYSOPR message queue as a reminder.

You should have a method of bringing these messages to the attention of the system administrator. If the interactive job at the console allocates the QSYSMSG message queue and places it in break mode, you are notified of any problems. For more information on QSYSMSG, see the *CL Programmer's Guide*.

Synchronization: When the system is synchronizing (resuming) a disk unit, the system response time becomes longer.

When mirrored protection is resumed on a suspended disk unit at DST, the synchronization is done during the IPL to the OS/400 licensed program.

Mirrored Protection Disk-Error Handling

Mirrored protection handles disk errors as follows:

Unrecoverable device error:

1. The system suspends the failing storage unit and mirrored protection is suspended for the mirrored pair.
2. The system continues operations using the operating storage unit of the mirrored pair.
3. A message is sent to the QSYSOPR message queue identifying the failing storage unit and informing you that mirrored protection is suspended for the mirrored pair.

Permanent read error:

1. The system reads from the other storage unit of the mirrored pair. If the permanent read error occurs on the other storage unit as well, the original read request completes with a permanent read error.
2. If the read operation from the other storage unit is successful, the data is written back to the first unit of the mirrored pair, assigning an alternate sector. Only then does the system signal that the original read request is complete.

Not operational storage unit:

1. The system attempts recovery. If it is successful, normal system operations continue with mirrored protection and without suspending or synchronizing the unit. The 0244 attention SRC is displayed in the control panel.
2. If recovery has not succeeded within a time limit, the unit is considered to have an unrecoverable device error, which is processed as described previously.

Time-out

1. The system attempts to recover from the timeout. If it is successful, normal system operations continue with mirrored protection and without suspending or synchronizing this unit.
2. If recovery is unsuccessful, the storage unit is considered to have an unrecoverable device error, which is processed as described previously.

I/O processor or bus failure

1. The system suspends each disk unit attached to the failing I/O processor or bus in the same way it is done for an unrecoverable error.
2. The system saves a copy of the failing I/O processor's storage so the problem can be diagnosed. The system continues without the failing I/O processor.

Disk-related failure of unit 1 before the IPL to the Operating System/400: See 5 on page 15-1 for restrictions that apply.

Missing Disk Units

If a disk unit, a controller, or an I/O processor fails during an IPL, the system detects the failure and does one of the following:

- Displays an SRC on the control panel if the keylock switch is not in the Manual position.
- Displays the Missing Disk Unit display on the console if the keylock switch is in the Manual position.

If the failing unit has mirrored protection and its mirrored unit is active, the following display is shown.

```
                          Suspend Missing Disk Units

The following disk units are missing from the system disk
configuration:

                          Serial
ASP   Unit  Type  Model   Number      Address         Reference Code
 2     3    9332  400     10-234233   0020-0201FFFF      1713
```

You can suspend mirrored protection on the affected units and continue the IPL. An entry is written in the problem log. You can run on-line problem analysis on the failing unit at a later time. The *type* and *reference code* fields can be used with the unit reference code guide to determine the cause of the problem. If the keylock

switch is not in the Manual position, a system reference code is displayed on the control panel. The system automatically suspends mirrored protection on the affected units and continues the IPL.

Saving a Unit

The system allows you to save data from storage units using the DST Save Disk Unit Data option.

The following rules apply to saving units on a system with mirrored protection:

- Only configured units can be saved.
- The save operation is not allowed when both mirrored units of a mirrored pair are active. Only one of the mirrored units can be saved. Therefore, one of them must be suspended.
- Only the active unit of a mirrored pair can be saved because the active unit contains the current data.
- If multiple failures cause the state of unit 1 to be unknown, saving of any storage unit is not allowed.

Restoring a Unit

In the mirrored environment, the system allows you to restore data to storage units.

The following rules apply to restoring units on a system with mirrored protection:

- The restore is possible only for an active device.
- This option can restore either a configured or nonconfigured disk unit.
- The restore operation requires that the unit restored to is the same type as the unit saved.
- The restore operation is not permitted if the state of a unit is unknown. You can restore unit 1 only to the IPL device.
- After restoring unit 1, the system performs an IPL to DST.
- The unit being restored must satisfy all mirrored protection configuration rules and restrictions.

Mirrored Protection for Unit 1 On the 9404 and 9402 System Units

If unit 1 on the 9402 or 9404 System Unit fails during the IPL, and the service processor is not type 2615, the system does not recover automatically. If a system reference code indicates that unit 1 at address 0010-0001FFFF has failed, the service representative may exchange the disk units in the mirrored pair for unit 1. No data recovery actions are required if the exchange is successful.

Note: For systems without service processor card 2615, you should not suspend mirrored protection on unit 1 at location 1 (address 0010-0001FFFF) for system units 9404 and 9402. The system will not be able to perform an IPL past DST. If unit 1 at location 1 is still on, the system will IPL from it and will give a *Back level load source* error. The service representative must then exchange the units in the mirrored pair for unit 1.

Active Mirrored Load Source Failure

If unit 1 has mirrored protection and one of the mirrored units fails, the system determines if the remaining mirrored unit has the correct system records and licensed internal code. If the remaining unit has the correct information, the failed unit is suspended and the system continues to run.

If the remaining mirrored unit does not contain the correct information, one of the following displays is shown if the keylock switch is in the Manual position.

```
                        Display Load Source Failure

The system could not use the load source disk unit that
contains correct data.  The following disk unit contains the
correct data:

Disk unit:
  Type . . . . . . . . . . . . . . . . . . :   9335
  Model  . . . . . . . . . . . . . . . . . :   B01
  Serial number  . . . . . . . . . . . . . :   57-1020449
  Address  . . . . . . . . . . . . . . . . :   0010-0000FFFF

Press Enter to use Dedicated Service Tools (DST).









F11=Display reference codes
```

The failing unit must be repaired before the system can be used.

If the system can locate a nonconfigured disk unit that matches the missing disk unit, the following display is shown.

```
                        Assign Missing Load Source Unit

The system could not locate one disk unit of the load source
mirrored pair.  The following disk unit is the nonconfigured
disk unit located at that address:

Disk unit:
    Type . . . . . . . . . . . . . . . . . . :   9335
    Model  . . . . . . . . . . . . . . . . . :   B01
    Serial number  . . . . . . . . . . . . . :   57-1020449
    Address  . . . . . . . . . . . . . . . . :   0010-0000FFFF

Press Enter to continue the IPL and configure the displayed
  disk unit as the mirrored pair of the load source disk unit.




F6=Use Dedicated Service Tools (DST)
F11=Display reference codes
```

You can accept the nonconfigured unit as the missing unit and continue the IPL.

If the keylock switch is not in the Manual position, a system reference code is displayed on the control panel. The failing unit must be repaired before the system can be used.

Unknown Unit 1 Status

If both the service processor and one unit of the mirrored pair for unit 1 have failed, the following display is shown.

```
                        Display Unknown Mirrored Load Source Status

The system can not determine which disk unit of the load
source mirrored pair contains the correct level of data.
The following disk unit is not available:

Disk unit:
    Type . . . . . . . . . . . . . . . . . . :   9335
    Model  . . . . . . . . . . . . . . . . . :   B01
    Serial number  . . . . . . . . . . . . . :   57-1020449
    Address  . . . . . . . . . . . . . . . . :   0010-0000FFFF

Press Enter to use Dedicated Service Tools (DST).









F11=Display reference codes
```

If the keylock switch is not in the Manual position, a system reference code is displayed on the control panel. The missing unit must be repaired. If the missing unit cannot be repaired, you must install the licensed internal code and restore the entire system.

If the system can locate a nonconfigured disk unit that matches the missing disk unit, the following display is shown.

```
                        Assign Missing Load Source Unit

The system could not locate one disk unit of the load source
mirrored pair.  The following disk unit is the nonconfigured
disk unit located at that address:

Disk unit:
  Type . . . . . . . . . . . . . . . . :  9335
  Model  . . . . . . . . . . . . . . . :  B01_
  Serial Number  . . . . . . . . . . . :  57-1020449
  Address  . . . . . . . . . . . . . . :  0010-0100FFFF

Press Enter to continue the IPL and configure the displayed
  disk unit as the mirrored pair of the load source disk unit.




F6=Use Dedicated Service Tools (DST)
F11=Display reference codes
```

You can accept the nonconfigured unit as the missing unit and continue the IPL.

Display Incorrect Licensed Internal Code Install

When the licensed internal code is restored on a mirrored unit for unit 1, one of the mirrored units may have the incorrect level of data stored on it. If this condition occurs, and the disk unit containing the correct data is not available, the licensed internal code is restored to the disk unit with the incorrect data. When an IPL is performed from disk and the correct disk unit is available, the following display is shown. If the keylock switch is not in the Manual position, system reference code (SRC) is shown on the control panel.

```
                Display Incorrect Licensed Internal Code Install

Licensed internal code has been installed on the incorrect
disk unit of the load source mirrored pair.
If you continue the IPL, the previously installed licensed internal
code installed on the incorrect disk unit of the mirrored
load source pair will be deleted.  The licensed internal code
will be replaced by the licensed internal code from the correct
disk unit.  The following disk unit is the correct disk unit.

Disk unit:
  Type . . . . . . . . . . . . . . . . . :     ____
  Model  . . . . . . . . . . . . . . . . . :     ____
  Serial number. . . . . . . . . . . . . . :     __-_______
  Address  . . . . . . . . . . . . . . . . :     ____-_______



Press Enter to continue the IPL and replace the licensed internal code.

F6=Use Dedicated Service Tools (DST)
```

Chapter 17. Mirrored Protection Considerations

This chapter provides information about the following:

- Abbreviations
- Using DST and SST
- Capacity planning tools
- Reconfiguring your system
- Examples of mirrored protection configuration

Abbreviations

ASP	Auxiliary storage pool
DST	Dedicated service tools
GSP400	General system predictor
I/O	Input/output
IOP	Input/output processor
IPL	Initial program load
KB	Kilobytes
MB	Megabyte
SST	System service tools
2800	Disk unit for the AS/400 9406 system unit. It has two actuators and a storage capacity of 640MB.
6100	Disk unit for the AS/400 9404 system unit. It has one actuator and a storage capacity of 315MB.
6102	Disk unit for the AS/400 9402 system unit. It has one actuator and a storage capacity of 320MB.
6103 G102	Disk unit for the AS/400 9402 system unit. It has one actuator and a storage capacity of 400MB.
6105	Disk unit for the AS/400 9404 system unit. It has one actuator and a storage capacity of 320MB.
6107	Disk unit for the AS/400 9404 system unit. It has one or two actuators and a storage capacity of 400MB or 800MB.
6110	Magnetic storage device controller IOP
6111	Magnetic storage device controller IOP
6112	Magnetic storage device controller IOP
9332-200	Disk unit for the System/3X systems. It has one actuator and a storage capacity of 200MB. It can be migrated to the AS/400 9406 system unit.
9332-400	Disk unit for the AS/400 9406 system unit. It has two actuators and a storage capacity of 400MB.
9335-A01	Device function controller for the 9335-B01 disk units
9335-B01	Disk unit for the AS/400 9406 system unit. It has two actuators and a storage capacity of 856MB.
9336	Disk unit for the AS/400 9406 system unit. It has two to four actuators and a storage capacity between 942 and 3428MB.

Using DST and SST for Mirrored Protection Management

Mirrored protection management is done by means of System Service Tools (SST) and Dedicated Service Tools (DST). The SST is used while the system is up and running; the DST implies a dedicated system.

Table 17-1. Disk Management

Operation	DST	SST
Display disk configuration	X	X
Change disk configuration	X	
Display ASP configuration	X	X
Configure ASP	X	
Display mirrored protection	X	X

Table 17-2. Mirrored Protection Management		
Operation	**DST**	**SST**
Calculate mirrored configuration	X	X
Starting mirrored protection	X	
Stopping mirrored protection	X	
Suspend mirrored protection	X	X
Resume mirrored protection	X	X
Add disk units	X	
Move disk units	X	
Replace disk units	X	X
Assign disk units	X	
Save disk units	X	
Restore disk units	X	

Capacity Planning Tools

Given normal growth within a business and the ever-increasing application demands being put on the data processing area of a business, users need to monitor changing system work loads and plan for future system requirements. The AS/400 Performance Tools licensed program provides a set of easy-to-use tools that can assist with these tasks.

A capacity planning function is available to help determine future system requirements. The capacity planner incorporates a performance model, an evaluator, and a high-level configurator that assists the user in configuring a balanced system that meets resource use guidelines as well as optional response time and throughput objectives. Users may override the evaluator recommendations if they desire to analyze any system configuration. The ability to base projections on measured data is another feature of the planner.

The capacity planning function is integrated into the data collection and reporting support. Electronic input of the data needed by the capacity planner allows the user to more easily do capacity planning and also allows for increased accuracy by projecting future requirements based on actual measured data. The capacity planner also includes a set of predefined work loads which the user can use to represent applications which are not currently running on the system. The measured data can be combined with the predefined work loads to show the effect of new applications on the current workload.

Reconfiguring Your System

To get your desired level of mirrored protection as determined in "Step 4: Determining the Level of Protection" on page 14-4, you may need to reconfigure your existing hardware or install additional hardware. Your IBM marketing representative will help you plan for any hardware configuration changes necessary to configure your system according to standard system configuration rules.

Determining Your Current Hardware Configuration

For an existing system, you can use the Work With Hardware Resources command to determine your current hardware configuration. From the displays, you can see the following:

- What disk units are attached to each controller (disk storage controller).
- What I/O processor (storage controller) each disk unit and controller is attached to.
- What bus each storage controller is attached to.
- Where each device or card is located - rack ID, EIA location, and card slot. (This information should have been entered into the system when the hardware was installed or changed.)

You need to consider separately each ASP that is to be mirrored.

Look at the Display Disk Configuration Status display using SST to determine what disk units are assigned to each ASP.

1. Enter the STRSST command to start the system service tools.
2. Select the Work with disk units option.
3. Select the Display disk configuration option.
4. Select the Display disk configuration status option. Notice which disk units are assigned to each ASP. If a printer is available, it is helpful to print the display of your disk configuration.
5. Exit SST.

6. Enter WRKHDWRSC TYPE(*STG) on the command line and press the Enter key. A list of storage resources is displayed. Beside each resource of type 6110 or 6111, enter a 9 (Work with resource).
7. A list of storage controller resources is displayed. Look at the entries that have a description of Disk Storage Control. Use F11 (Display resource addresses/statuses) to display the resource address of each resource.
8. Look at the *resource address* field of each disk unit and disk storage controller and determine its bus number, storage controller number, and disk storage controller number. Refer to Figure 13-2 on page 13-6.
9. It may be helpful for you to draw a diagram to represent the data collected from the Disk Configuration Status display and the storage controller resources list, and to show how each disk unit and controller is attached. (See Figure 17-1 on page 17-3.)

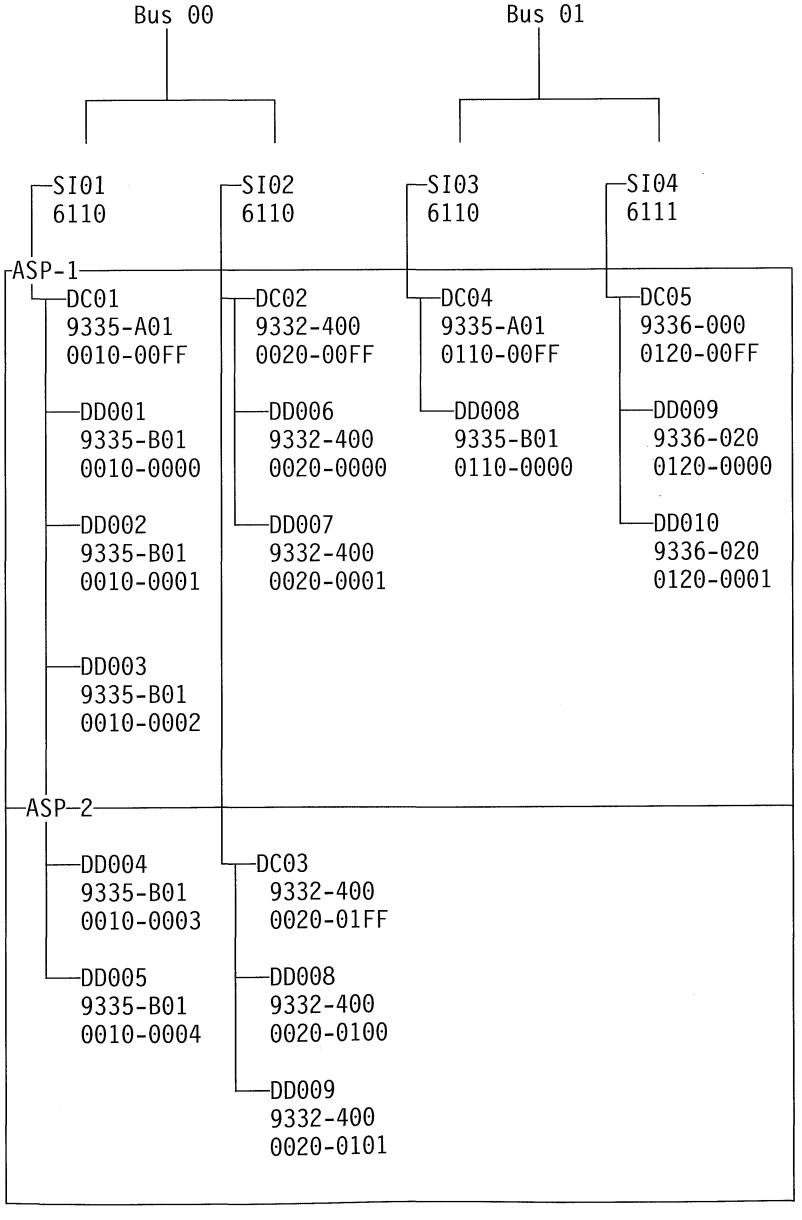


Figure 17-1. Hardware Configuration Diagram

Balancing Your Configuration

With the help of your IBM marketing representative, follow the standard configuration rules in the *9406 System Installation and Upgrade Guide*, and plan how to reconfigure your system. If the system is configured according to the standard rules, it is balanced to provide the best possible mirrored protection with the hardware available. You may find it helpful to use diagrams to note the new configuration.

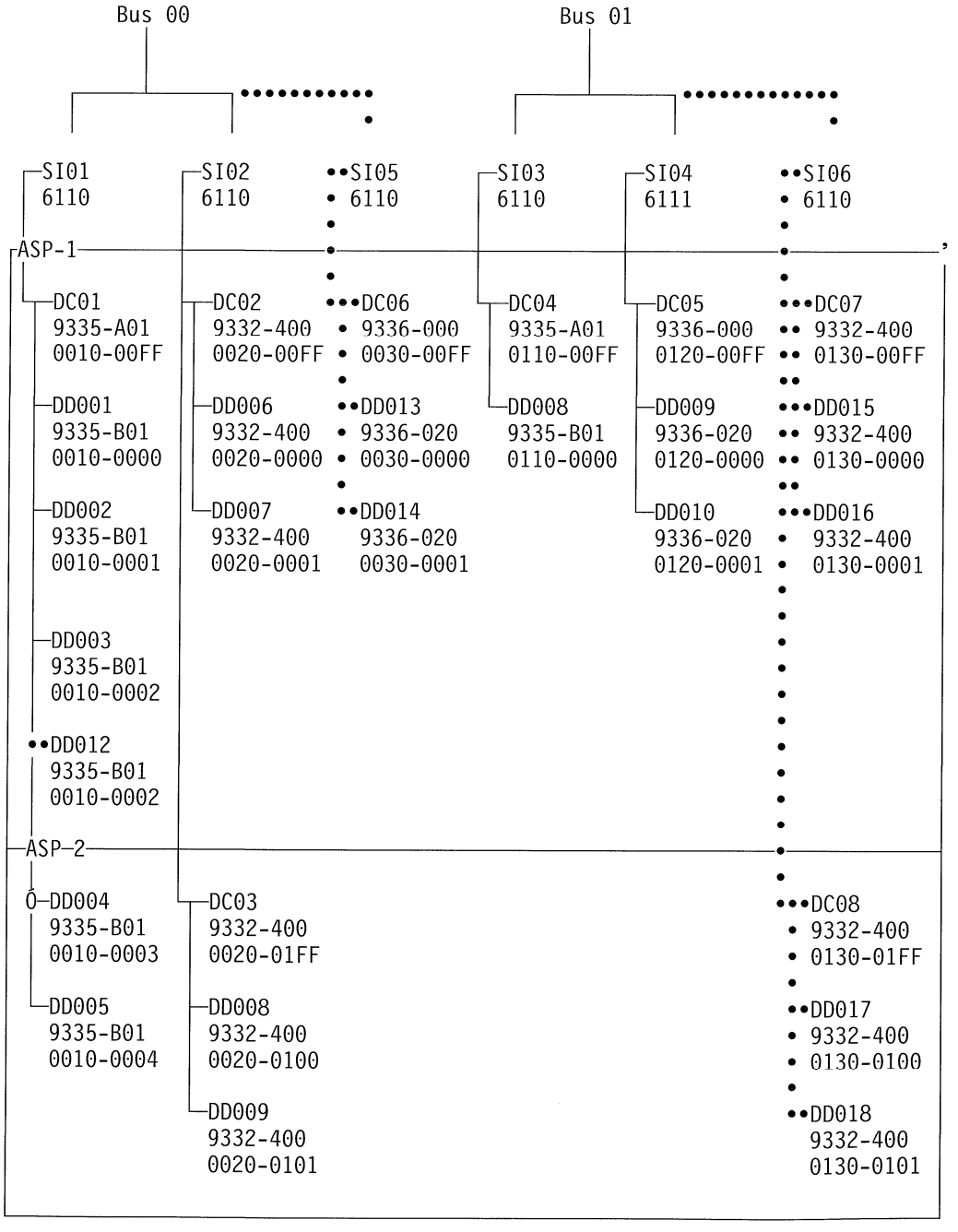


Figure 17-2. Augmented Hardware Configuration

Figure 17-2 shows how the configuration in Figure 17-1 on page 17-3 looks after you add hardware to allow for mirrored pairs and maintain the same capacity for growth. Notice that all disk units have bus-level protection except for unit 1, which has controller-level protection.

Examples of Mirrored Protection Configurations

Example of a Mirrored Configuration with Disk Unit-Level Protection

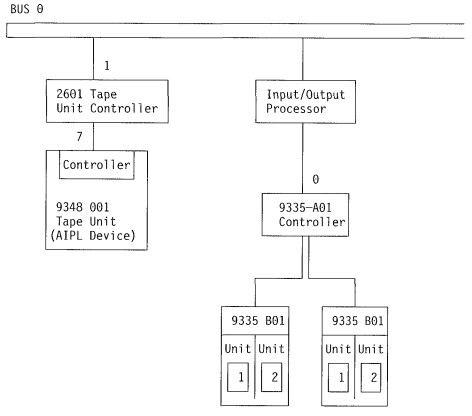


Figure 17-3. Example of a Mirrored Configuration with Disk Unit-Level Protection

Figure 17-3 shows a small system with disk units of type 9335, one controller, and one I/O processor. The system has disk unit-level protection. This configuration is not the best; a failure of the bus, the I/O processor, or the 9335-A01 controller causes all disk units to be unusable, and the system to end abnormally.

Concurrent maintenance may not be possible because of the requirement of the test procedure to have the controller dedicated to the repair action.

To increase the effectiveness of mirrored protection for this set of disk units, an additional 9335-A01 disk storage controller is required. If the controller is added, the failure of any one of the disk units or a 9335-A01 can be tolerated. Because they must be at I/O processor address 1 or 2, there is no way for unit 1 to be under separate I/O processors.

Concurrent maintenance is not possible if the I/O processor must be reset as part of the problem analysis procedures.

Example of a Mirrored Configuration with Controller-Level Protection

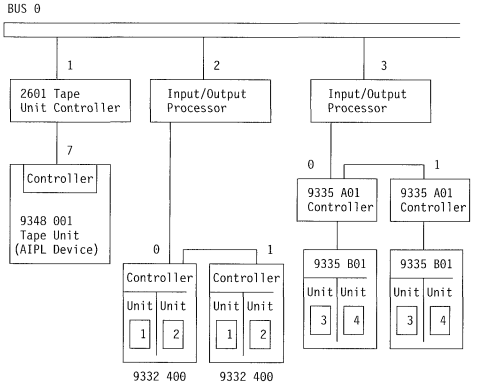


Figure 17-4. Example of Mirrored Configuration with Controller-Level Protection

Figure 17-4 shows a small-to-medium sized system with two different disk unit types (9332s and 9335s), two I/O processors, and controller-level protection for all disk units. I/O processor 2 controls the 9332 disk units, while I/O processor 3 controls the 9335 disk units. All disk units have controller-level protection because each disk unit and its mirrored unit are under separate controllers. This is the best protection possible with this hardware configuration.

Any disk unit or controller can be powered off while the system is running. However, if the problem analysis procedures need to reset the I/O processor controlling the disk unit or controller that has a failure, then concurrent maintenance is not possible.

Example of a Mirrored Configuration with I/O Processor-Level Protection on Units Other Than Unit 1

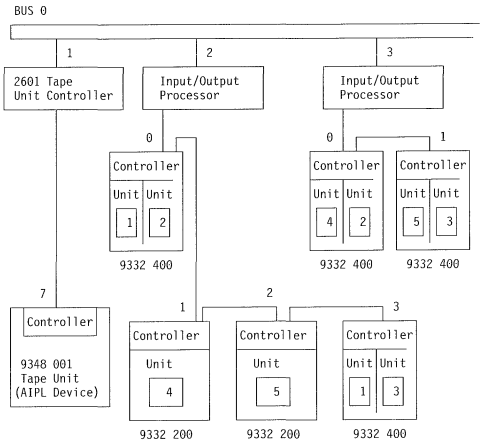


Figure 17-5. Example of a Mirrored Configuration With I/O Processor-Level Protection on Units Other Than the Unit 1

Figure 17-5 shows a system with all disk units of the same type (9332) and two disk I/O processors. All units other than unit 1 have I/O processor-level protection. This is the maximum protection possible on a single-bus system.

If I/O processor 2 has to be reset during the problem analysis procedures, then concurrent maintenance is not possible because both unit 1 and its mirrored unit are under this I/O processor. For units under I/O processor 3, concurrent maintenance is usually possible.

Example of a Mirrored Protection Configuration with Bus-Level Protection on Units Other Than the Unit 1

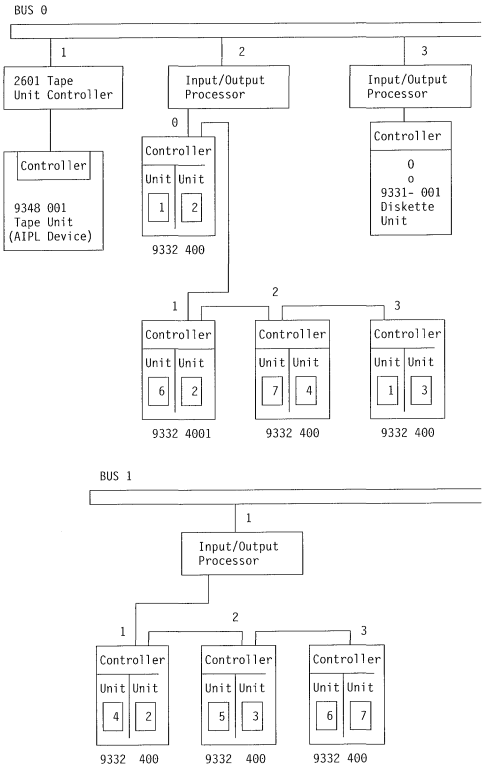


Figure 17-6. Example of a Mirrored Configuration with Bus-Level Protection

Figure 17-6 shows a two-bus system with 9332 disk unit types and bus-level protection for all disk units except unit 1. This means that bus 1 can fail while the system continues to run. Concurrent maintenance is usually possible for all units. Unit 1 may not be serviced during system operation if I/O processor 2 has to be reset as part of the problem analysis procedures.

If I/O processor 1 on bus 1 has to be reset as part of the problem analysis procedure, then all disk units in the system except unit 1 and its mirrored units are suspended and therefore unprotected for the duration of the maintenance.

To improve protection on this set of disk units, additional I/O processors on bus 1 are required. Storage units 3 and 5 under I/O processor 1 can be moved to an additional I/O processor. This reduces the number of storage units that are unprotected during the maintenance of any disk units on bus 1.

Chapter 18. Performance Considerations for Mirrored Protection

This chapter discusses the effects of mirrored protection on system performance. The normal run-time performance is looked at, as well as the effect of synchronization on system operation. We also look at some ways to improve performance in the mirrored environment.

Run-Time Performance for Mirrored Protection

Following is a description of a general performance expectation for normal operation on a system with mirroring.

Processing unit: Mirrored protection places a minimal demand on the processing unit. Mirrored protection uses the processing unit mainly for flagging and look-up operations, and some calculations, such as sector address compares, stripe computations, and page allocation. Processing unit use can be expected to increase by about 1%.

Disk:

- Read performance

  When storage units are busy, the ability to read from either mirrored unit leads to a performance improvement in read-intensive jobs. If storage units are less than 20% busy, then there is no benefit from the balancing effect of mirrored-read operations.

- Write performance

  When storage units are busy, write performance decreases in the mirrored environment because every write operation causes two mirrored-write operations. Jobs that characteristically perform large numbers of database I/O operations experience this decrease in performance. Jobs in this category are:

  - Batch update jobs
  - Jobs using journaling and commitment control
  - Large restore operations
  - Save operations to save files
  - Rebuild of large access paths

Synchronization Effects

When a mirrored pair is synchronized, the synchronization process affects other jobs running on the system. Synchronization is a Licensed Internal Code task that runs in the machine pool. Some of the major factors that affect job performance during synchronization are:

- Use of the disk subsystem being synchronized
- I/O activity to the disk being synchronized that is mainly write operations
- Available main storage
- The number of disks being synchronized simultaneously

Some suggestions to minimize the effect of synchronization are:

- If you choose to resume mirrored protection using DST, the synchronization process occurs during the IPL to the OS/400 licensed program. The system must complete the IPL before you can use it. Mirroring can be resumed while the system is operational.
- You can resume multiple disk synchronization two ways. You can resume mirrored protection on all mirrored pairs at the same time, or you can resume mirrored protection to one mirrored pair at a time.

  The first approach returns the system to a fully protected state faster, but can significantly impact other jobs while the synchronization takes place. Using much of the disk space also slows the synchronization process.

  The second approach affects other jobs on the system to a lesser extent, while extending the exposure window (the time during which mirrored pairs on the system are unprotected).

  It is not possible to recommend one method over the other, as priorities vary from system to system. Being aware of these factors can help you make a decision.

- The length of time a mirrored unit is suspended has no impact to the length of the synchronization process. The moment a

storage unit is suspended (either manually or by the system) in response to a disk failure, it must be fully synchronized.

- An uninterruptible power supply enables the system to copy main storage during a power failure, and reduces the chance of a long recovery on the following IPL. If there is an incomplete copy of main storage, the system synchronizes parts of the mirrored pairs. This makes the recovery process longer.

Additional IPL Time after an Abnormal System End

In some cases, if the system ends abnormally, the system cannot be sure that the last changes were written to both units of each mirrored pair. In this case, the system synchronizes parts of the mirrored pairs. The synchronization occurs during the IPL following an abnormal system end. If the system can save a copy of main storage before it ends, this process takes just a few minutes. If not, the synchronization process can take much longer.

Part 7. Uninterruptible Power Supply

Chapter 19. Description of Power Loss Recovery

One way to help prevent the system from ending abnormally because of a power failure is to provide an uninterruptible power supply. An uninterruptible power supply provides auxiliary power to the system during a power failure.

Power Supply Support

There are four approaches to power supply support, depending on the type and model of the AS/400 system:

- Optional battery feature on the 9402 and 9404 System Units
- Standard internal battery feature on the 9406 Model D, E, F and G System Units
- Limited support with vendor-supplied uninterruptible power supply
- Complete support with vendor-supplied uninterruptible power supply

Optional Battery Feature on the 9402 and 9404 System Units.

An optional battery feature exists for the 9402 and 9404 System Units.

If you do not plan to have a vendor-supplied uninterruptible power supply, the optional battery feature is the normal approach. However, you can use a vendor-supplied uninterruptible power supply with or without the battery feature.

You should use the Battery feature with the system defaults for the 9402 and 9404 System Unit. When you use the Battery feature, the system remains active if utility power is lost, although the work station jobs may end abnormally if you do not have a user-written program to handle this situation. You need to restart the work station jobs (sign on again) if utility power is restored while the system is operating on the battery. If utility power is not restored shortly, either the delay timer or the weak-battery signal causes the system to save main storage and power down.

If the system is powered down while on an uninterruptible power supply, the QPWRRSTIPL system value determines if an IPL will occur when utility power is restored. The default is to perform an IPL for the system programs.

Standard Internal Battery Feature on the 9406 Model D, E, F and G System Uni

For Version 2 hardware, the 9406 model D, E, F and G System Units come with an internal battery that allows the system time to copy main storage in the event a power failure occurs. If you have frequent power outages, consider using an uninterruptible power supply. Figure 19-1 shows a logical view of the standard internal battery feature with an uninterruptible power supply.

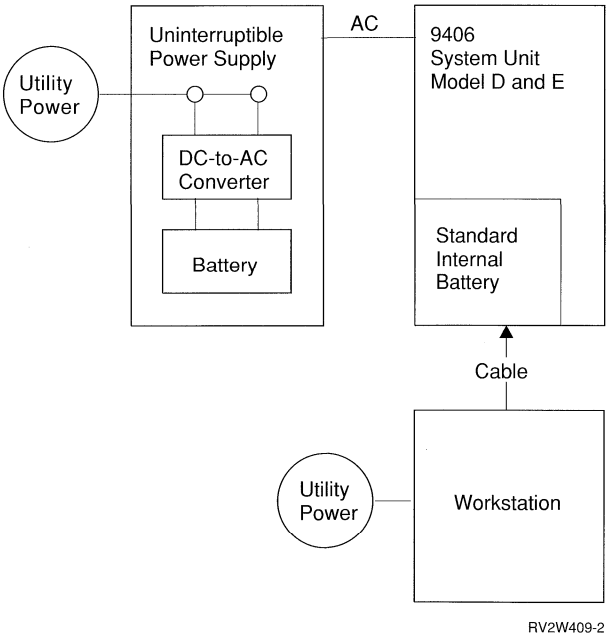


Figure 19-1. Logical View of the 9406 Model D and E with Attached Power Supply

For Version 1 hardware, the 9406 System Unit requires a vendor-supplied uninterruptible power supply.

Limited Power Supply Support

Some vendor-supplied uninterruptible power supplies provide *limited* support by supplying power to a limited number of devices: the processing unit, unit 1[1], and all storage device controllers. If the utility power fails, the system continues to run for up to 30 seconds. If power does not return in 30 seconds, the system saves main storage and a power down of the system occurs.

Unit 1 contains the Licensed Internal Code and resides on the disk unit from which the system performs an IPL. Storage device controllers are cards in the card enclosure of the system that disk units attach to.

A system with one system unit and an expansion unit that contains disk units would not benefit from the use of an uninterruptible power supply that provides limited support because both system units would need power. In addition, a single 9404 System Unit would not benefit from the limited support power supply because the entire system would be powered.

Complete Power Supply Support

The most frequently used type of power supply provides *complete* support by supplying power to the processing unit and all disk units long enough to last through a temporary power outage or to power the system down normally.

Uninterruptible power supplies vary, but Figure 19-2 shows a logical view of a typical uninterruptible power supply.

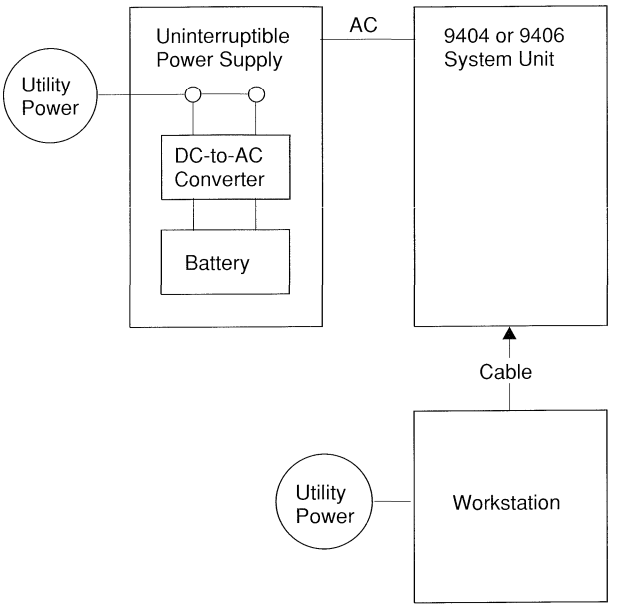


Figure 19-2. Logical View of a Typical Uninterruptible Power Supply

Uninterruptible Power Supply Attachment

The uninterruptible power supply attachment allows you to attach a vendor-supplied uninterruptible power supply. The uninterruptible power supply attachment can be used on all models. The vendor-supplied uninterruptible power supply must be able to supply power to the processing unit and all disk units. The vendor-supplied uninterruptible power supply optionally supplies power to the other devices (work stations).

A vendor-supplied uninterruptible power supply that provides limited support need only provide power to the processing unit, unit 1, and the storage device controllers.

You control if a power down of the system occurs or operations continue when the utility power is interrupted. See Figure 19-3 on page 19-3 for an example of the attachment of an uninterruptible power supply supplied by a vendor.

[1] If the system ASP has mirrored protection, the mirrored unit for unit 1 is also supplied power.

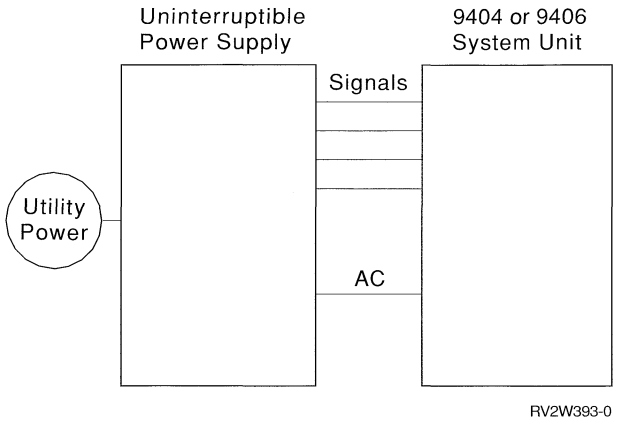


Figure 19-3. Attachment of a Vendor-Supplied Uninterruptible Power Supply

Description of System Values for Uninterruptible Power Supply

This topic discusses the system values that affect the uninterruptible power supply. Three system values define the action the system takes in response to a change in one of the power supply signals, as shown in Figure 19-4 on page 19-5.

QUPSMSGQ - Uninterruptible Power Supply Message Queue

This value controls which message queues the power supply messages are sent to.

Messages sent by the system about the power supply are sent to the system operator (QSYSOPR) message queue regardless of the value specified in the system value. If you specify a different message queue, that message queue also receives the same power supply messages. Specify a different message queue if you have one of the following:

- Another message queue you want to receive the power supply messages (for example, the data processing manager's message queue).
- A program that handles events that are related to the uninterruptible power supply. For more information about writing a program to handle the power, see the topic "Using a Program to Handle Uninterruptible Power Supply Conditions" on page 19-13.

See the topic "Uninterruptible Power Supply Messages" on page 19-12 for information on power supply messages and a description of the messages and message IDs.

QUPSDLYTIM - Uninterruptible Power Supply Delay Time

This value controls the amount of time the system waits before saving main storage and powering down the system. The default value, *CALC, causes the system to calculate the wait time in seconds for the Battery Power Unit.

To perform a normal power down of the system, you must end all other work on the system. Then, the Power Down System (PWRDWNSYS) command is used. The PWRDWNSYS command can be run with OPTION(*IMMED) or with a time delay, which causes any active jobs to be abnormally ended. In either case, successfully completing the PWRDWNSYS command causes the next IPL of the system programs to be considered normal by both the Licensed Internal Code and the OS/400 licensed program.

If an uninterruptible power supply or Battery Power Unit is being used and the system senses that it is on back-up power, a timer is set based on the QUPSDLYTIM system value. If utility power is restored before the time ends, the system ends the timer. If the timer ends first, the system saves main storage and powers down.

You can also:

- Specify *BASIC if the uninterruptible power supply attached to the system provides limited support. This value must be specified if the uninterruptible power supply that is attached provides power for only the processing unit, unit 1, and the storage device controllers.

  The system sets a timer based on the system configuration. If utility power is restored before the time ends, the system ends the timer and normal operations continue. If the timer ends first, the system saves main storage and powers down the system.
- Specify a value of 0 to ask the system to immediately save main storage and then power down. Normally, you only specify this value when you are controlling the uninterruptible power supply conditions with a program and suspect that the battery is not properly recharged.

- Specify a different value based on the duration of your battery's charge and the time it normally takes the system to save main storage and power down. See the topic "Description of System Values for Uninterruptible Power Supply."
- Specify a value of *NOMAX to tell the system that you do not want this function to save main storage and then power down. Normally, specify *NOMAX only when you are controlling the uninterruptible power supply conditions with a program. See "Using a Program to Handle Uninterruptible Power Supply Conditions" on page 19-13.

QPWRRSTIPL - Power Restore IPL Option Following a Power Down

This value controls what happens if the system is ended when utility power is off and then power is later restored. The default is 0 (Not allowed), which causes the system to not perform an IPL when utility power is restored.

Normally, you would only set this value to 0 if:

- You prefer to manually start the system again.
- You have a power handling program that determines whether the batteries have not been recharged enough to allow another IPL.

Optional Battery Feature for the 9402 and 9404 System Units

The following applies if you are using the battery feature on the 9402 or 9494 System Units without an uninterruptible power supply attached.

Setting the System Values

QUPSDLYTIM System Value: With the Battery Power Unit on the 9402 or 9404 System Unit, you would normally use the default value of *CALC. This should allow sufficient time for the system to save main storage and power down, based on the battery rating and the maximum size of main storage.

QPWRRSTIPL System Value: Set this value to 1 (Allowed) if you have a Battery Power Unit and you use the Power Down System (PWRDWNSYS) command while utility power is off and the battery feature is providing power.

The system will save main storage and power down after the PWRDWNSYS command is started. When the utility power is restored, the system checks the values for the QPWRRSTIPL system value. If it set to 1, the system performs an IPL automatically.

The keylock switch must be in the Normal or Automatic position for the automatic IPL function to work.

The QPWRRSTIPL system value has no effect if the system is powered down normally while running on utility power. If utility power goes off after the system is powered down normally, the system does not automatically do an IPL when utility power is restored.

The system may override a 1 value to try to protect the battery from being completely discharged. There are two cases:

- During an IPL, if utility power fails and a weak battery is signaled before the OS/400 licensed program is given control, the system saves main storage, powers down, and overrides the power restore option to 0 (Not allowed).
- When performing an IPL due to the power restore IPL value and a utility power failure occurs, if the OS/400 licensed program is not given control before the power supply delay time ends, the system saves main storages, powers down, and overrides the power restore option to 0 (Not allowed).

Power Down when Using a Power Supply or 9402 or 9404 Battery Feature

The power-down function when using power supply support operates from the Licensed Internal Code and does not involve the OS/400 licensed program. It cannot be started from a command (the PWRDWNSYS command is not run). There are multiple phases and considerations for the power-down function associated with power supply support:

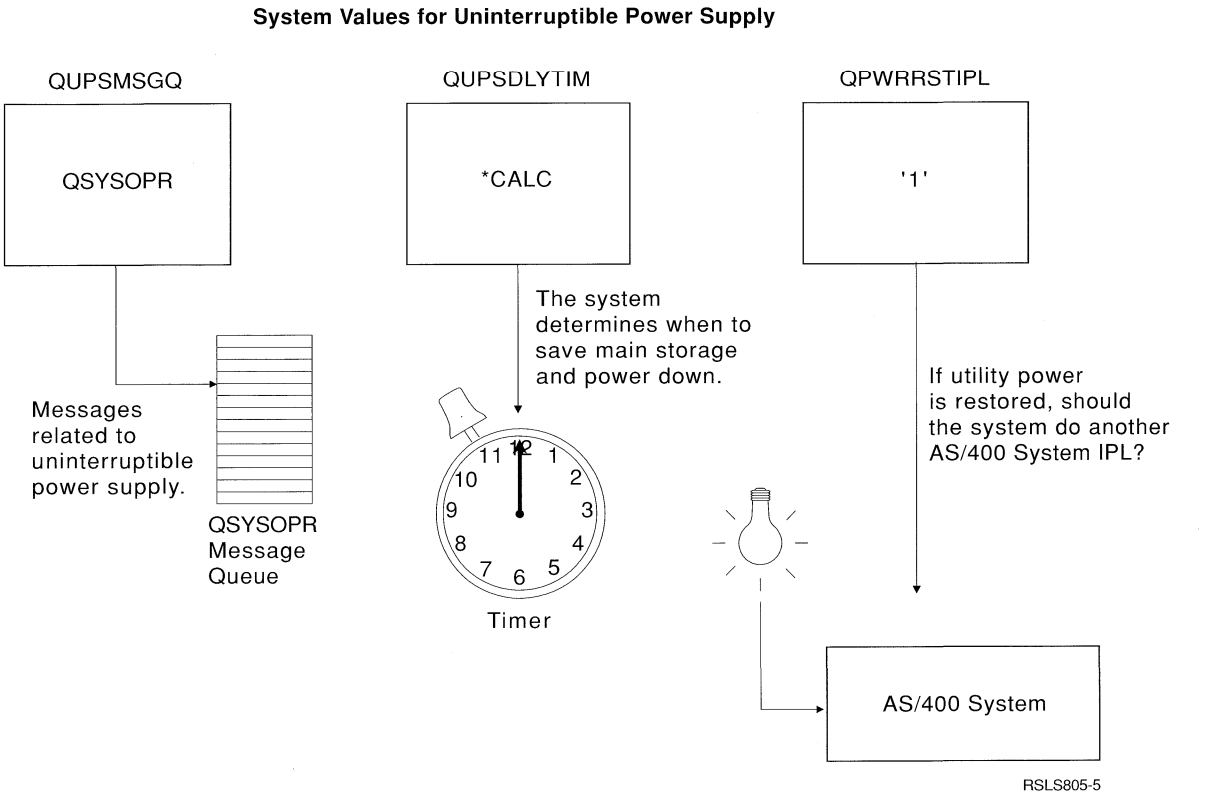


Figure 19-4. System Values That Define Uninterruptible Power Supply Action

- The Licensed Internal Code signals each job to end at the next internal instruction boundary. (This same function, known as forced Licensed Internal Code completion, can be started from the control panel. For more information about Licensed Internal Code, refer to the topic "Licensed Internal Code Completion" on page B-9.) Each job attempts to come to a point where an orderly end can occur.

  Usually a job is at the next internal instruction boundary or will be shortly. However, some internal instructions are long running, such as those that build access paths or create programs. A job performing one of these long-running instructions may not be able to reach an orderly end in the time allowed.

  This first phase is shorter than that performed by the forced Licensed Internal Code completion function. The time allowed for completion varies, depending on the main storage size of the system.

- If all jobs have reached a point where they can properly end, the system performs additional operations and then powers down. For example, the system must write all of the changed pages from main storage to auxiliary storage. The next IPL is normal from a Licensed Internal Code viewpoint, but is considered abnormal from the operating system viewpoint. There is no directory recovery or access path recovery required.

- The Licensed Internal Code ends jobs that cannot reach a point (in a reasonable amount of time) where they can end properly, and continues to power down. The next IPL does not involve directory recovery, but may require some access path recovery, depending on what was happening to the job when it ended.

- In some cases the power-down function cannot complete successfully. The next IPL will be abnormal from a Licensed Internal Code viewpoint.

The power-down sequence sets the system to do an IPL automatically when utility power is restored if the system value QPWRRSTIPL is set to 1.

The system completes the power down if utility power is restored while the system is in the power-down sequence. If the QPWRRSTIPL

system value is set to 1, the system automatically performs an IPL after the power down is complete.

The power-down function for power supply support always causes an abnormal IPL from an OS/400 point of view. This allows certain cleanup functions to occur, but normally will not significantly affect the IPL time.

Weak Battery Conditions

During a power failure, any indication of a weak battery causes the system to save main storage and power down, no matter what your program is designed to do. If the system value QUPSDLYTIM is met, the system also automatically starts to save main storage and powers down. You can prevent this system action by not connecting the lead for the weak-battery condition.

If you do not have a weak-battery indication, you must consider the results if the utility power fluctuates. Normally, the battery does not fully recharge unless the system has been on utility power for some time. Account for this in your program by adjusting the values for QUPSDLYTIM and QPWRRSTIPL.

For example, if power has gone off, comes back on, and then goes off again, you may want to adjust the QUPSDLYTIM to account for the fact that the battery is not fully recharged. If power has gone off and on several times in a 30-minute period, you may want to set QPWRRSTIPL to 0 (No) and issue the PWRDWNSYS command. This prevents the system from doing another IPL. You can either wait for a manual IPL or set the QIPLTIME system value to cause a timed IPL at a later point. In general, it is desirable to prevent the battery from being totally used up.

Standard Internal Battery for the 9406 Model D, E, F,

and G System Units

The following information applies to the 9406 Model D and E System Units without an uninterruptible power supply attached.

Setting the System Values

The internal batteries in 9406 Model D, E, F, and G systems provide power to the system unit, the 5040 extension unit, and the 5040 expansion unit for a minimum of 5 minutes. These batteries provides the system with limited (*BASIC) support by default.

The internal battery provides power for at least as long as the time in Table 19-1. This time includes the shutdown time in addition to a 30-second delay time used when waiting for utility power to return.

Table 19-1 shows the time it takes for the 9406 Models D and E to power down.

Table 19-1. Basic Uninterruptible Power Supply. Save Main Storage Times for the 9406 System Unit Models D and E (Internal Battery)

Main Storage Size	Time (seconds)	Time (minutes)
8	35	0 minutes 35 seconds
16	40	0 minutes 40 seconds
32	60	1 minute 0 seconds
64	75	1 minute 15 seconds
96	95	1 minute 35 seconds
128	115	1 minute 55 seconds
160	135	2 minutes 15 seconds
192	160	2 minutes 40 seconds
224	180	3 minutes 0 seconds
256	200	3 minutes 20 seconds
288	220	3 minutes 40 seconds
320	240	4 minutes 0 seconds
352	265	4 minutes 25 seconds
384	280	4 minutes 40 seconds

Power Down when Using the Battery Feature on the 9406 Models D, E, F, and G

The internal battery in 9406 Model D, E, F, or G systems provides power to the system for 5 minutes. The internal batteries supply the system with limited (*BASIC) support by default.

The power-down function for the limited (*BASIC) support operates from the Licensed Internal Code and does not involve the OS/400 licensed program. It cannot be started from a command, but is selected by specifying the value *BASIC for

the QUPSDLYTIM system value for the 9406 model D and E system units. The following is performed by the limited power supply support:

- Contents of main storage are copied (saved) to the main storage dump area in unit 1.
- The system is powered down.
- The next IPL is normal from the viewpoint of the Licensed Internal Code.

  The system must write all of the changed pages from the saved copy of main storage to auxiliary storage. The system performs a directory recovery that is shortened and you may need to recover certain access paths. Typically, the recovery time should be significantly reduced.

The power-down sequence using limited power supply support sets the system to do an automatic IPL when utility power is restored if the system value QPWRRSTIPL is set to 1.

The system completes the power-down if utility power is restored while the system is in the limited power supply support power-down sequence. If the QPWRRSTIPL system value is 1, the system automatically does an IPL after the power down is complete.

The power-down function using *BASIC always causes an abnormal IPL from an OS/400 point of view. This allows certain cleanup functions to occur, but normally does not significantly affect the IPL time.

Weak Battery Conditions

During a utility power failure, any indication of a weak battery causes the system to save main storage and power down, no matter what your program is coded to do. If the system value QUPSDLYTIM is met, the system also automatically starts to save main storage and powers down.

If you are using the limited uninterruptible power supply, the weak battery signal causes the power-down sequence for limited power supply support (see the topic "Power Down for Limited (*BASIC) Power Supply Support" on page 19-8).

If you do not have a weak-battery indication, you must consider the results if the utility power fluctuates. Normally, the battery does not fully recharge unless the system has been on utility power for some time. Account for this in your program by adjusting the values for QUPSDLYTIM and QPWRRSTIPL.

For example, if power has gone off, comes back on, and then goes off again, you may want to adjust the QUPSDLYTIM to account for the fact that the battery is not fully recharged. If power has gone off and on several times in a 30-minute period, you may want to set QPWRRSTIPL to 0 (No) and issue the PWRDWNSYS command. This prevents the system from doing another IPL. You could either wait for a manual IPL or set the QIPLTIME system value to cause a timed IPL at a later point. In general, it is desirable to prevent the battery from being totally used up.

Uninterruptible Power Supply with Limited Support

The following information applies to the 9402, 9404, and 9406 Model B System Units without the Battery Feature.

Setting the System Values

With the uninterruptible power supply attachment, do not use the value *CALC. Consider how long your battery stays charged and how long the system takes to save main storage and then power down. Specify the difference between these two as the time for the uninterruptible power supply delay time (QUPSDLYTIM), that is, how long the system should wait before it saves main storage and powers down. The time line of the function is shown in Figure 19-5 on page 19-10.

With the uninterruptible power supply attachment, the value *BASIC must be specified if an uninterruptible power supply with limited power support is providing power only to the processing unit, unit 1, and the storage device controllers (this means that not all system units are powered by the uninterruptible power supply). This approach normally requires a less expensive uninterruptible power supply but provides less function. Table 19-1 on page 19-6 shows the time it takes for and uninterruptible power supply with limited support to power down.

Table 19-2. Uninterruptible Power Supply with Limited Support. Save Main Storage Times for the 9402, 9404, and 9406 System Units Model B.

Main Storage Size	Time (seconds)	Time (minutes)
8	45	0 minutes 45 seconds
16	60	1 minute 0 seconds
32	95	1 minute 35 seconds
64	160	2 minutes 40 seconds
96	225	3 minutes 45 seconds
128	300	5 minutes 0 seconds
160	350	5 minutes 50 seconds
192	420	7 minutes 0 seconds

Power Down for Limited (*BASIC) Power Supply Support

The power-down function for the limited (*BASIC) power supply support operates from the Licensed Internal Code and does not involve the OS/400 licensed program. It cannot be started from a command, but is selected by specifying the value *BASIC for the QUPSDLYTIM system value. The following is performed by the limited power supply support:

- Contents of main storage are copied (saved) to the main storage dump area in unit 1.
- The system is powered down.
- The next IPL is normal from the viewpoint of the Licensed Internal Code.

  The system must write all of the changed pages from the saved copy of main storage to auxiliary storage. The system performs a directory recovery that is shortened and you may need to recover certain access paths. Typically, the recovery time should be significantly reduced.

The power-down sequence using limited power supply support sets the system to do an automatic IPL when utility power is restored if the system value QPWRRSTIPL is set to 1.

The system completes the power-down if utility power is restored while the system is in the limited power supply support power-down sequence. If the QPWRRSTIPL system value is 1, the system automatically does an IPL after the power down is complete.

The power-down function using *BASIC always causes an abnormal IPL from an OS/400 point of view. This allows certain cleanup functions to occur, but normally does not significantly affect the IPL time.

Weak Battery Conditions

Any indication of a weak battery while on the uninterruptible power supply causes the system to save main storage and power down, no matter what your program is designed to do. If the system value QUPSDLYTIM is met, the system also automatically starts to save main storage and powers down. You can prevent this system action by not connecting the lead for the weak-battery condition.

If you are using the limited uninterruptible power supply, the weak battery signal causes the power-down sequence for limited power supply support.

If you do not have a weak-battery indication, you must consider the results if the utility power fluctuates. Normally, the uninterruptible power supply battery does not fully recharge unless the system has been on utility power for some time. Account for this in your program by adjusting the values for QUPSDLYTIM and QPWRRSTIPL.

For example, if power has gone off, comes back on, and then goes off again, you may want to adjust the QUPSDLYTIM to account for the fact that the battery is not fully recharged. If power has gone off and on several times in a 30-minute period, you may want to set QPWRRSTIPL to 0 (No) and issue the PWRDWNSYS command. This prevents the system from doing another IPL. You could either wait for a manual IPL or set the QIPLTIME system value to cause a timed IPL at a later point. In general, it is desirable to prevent the battery from being totally used up.

Uninterruptible Power Supply with Complete Support

The following information applies to the uninterruptible power supply with complete support.

Setting the System Value

QUPSDLYTIM System Value: If you have the uninterruptible power supply attachment, do not use *CALC. The value *CALC is designed for the 9402, 9404, and 9406 Models D and E Battery Power Unit feature. Instead, specify a value based on the following.

When the system senses that it is on back up power (either the uninterruptible power supply or the Battery Power Unit), a timer is set based on the QUPSDLYTIM system value. If utility power is restored before the time ends, the system ends the timer. If the timer ends first, then the system saves main storage and powers down. (See Figure 19-5 on page 19-10.)

You can also:

- Specify a value of 0 to ask the system to immediately save main storage and then power down. Normally, you only specify this value when you are controlling the uninterruptible power supply conditions with a program and suspect that the battery is not properly recharged.
- Specify a value of *NOMAX to tell the system that you do not want this function to save main storage and then power down. Normally, specify *NOMAX only when you are controlling the uninterruptible power supply conditions with a user power-handling program.
- Specify a different value based on how long your battery stays charged and the time it normally takes the system to save main storage and power down.

  If you have a main storage size of 16MB or less, consider a 5-minute value for the length of time to power down the system (PWRDWNSYS command). For a larger system, add 1 minute for each 16MB. For example, for a 64MB system you would use a value of 8 minutes (5 + 3).

  This minutes-per-megabyte number is not an exact number. It is an estimate based on the number of jobs that normally are run in different main storage sizes and the average amount of time it takes for the system to end them. For more information, see the topic "Power Down when Using Complete Power Supply Support" on page 19-10 for the function your system must perform. Add a buffer of time and make a conservative estimate to account for unusual circumstances. Also consider that if power is fluctuating, your battery may be discharged to the point where it may not have enough time to save main storage and power down.

  When you know how long your battery stays charged and the time it takes to save main storage and power down, subtract to determine the value to specify for QUPSDLYTIM. For example, if you have a 32MB system and your battery stays charged 15 minutes, calculate as follows:

```
Battery rating            15  Minutes
Save-main-storage and   - 6          (5 Standard + 1)
power-down time           ----
                            9
Buffer                  - 2
                          ----
                            7  Minutes (420 seconds)
```

  To change the system QUPSDLYTIM value according to the above calculation, specify:

```
CHGSYSVAL SYSVAL(QUPSDLYTIM) VALUE('420')
```

  If you change the main storage size, consider changing the QUPSDLYTIM value.

QPWRRSTIPL System Value: Specify the value 1 (Allowed) if:

- You have a Battery Power Unit or an uninterruptible power supply attachment, and you use either the Power Down System (PWRDWNSYS) command or the power down function for the limited power support while utility power is off and the uninterruptible power supply is providing power.

  The system is powered down when either the PWRDWNSYS command or the limited support power-down sequence starts. If utility power is restored during the sequence, the system is powered down and an IPL occurs if the system value QPWRRSTIPL is set to 1. If utility power is restored after the system is powered down, the system uses the value to see if an automatic IPL should occur.

The keylock switch must be in the Normal or Automatic position for the automatic IPL function to work.

The QPWRRSTIPL system value has no effect if the system is powered down normally while running on utility power. If utility power goes off after the system is powered down normally, the system does not automatically do an IPL when utility power is restored.

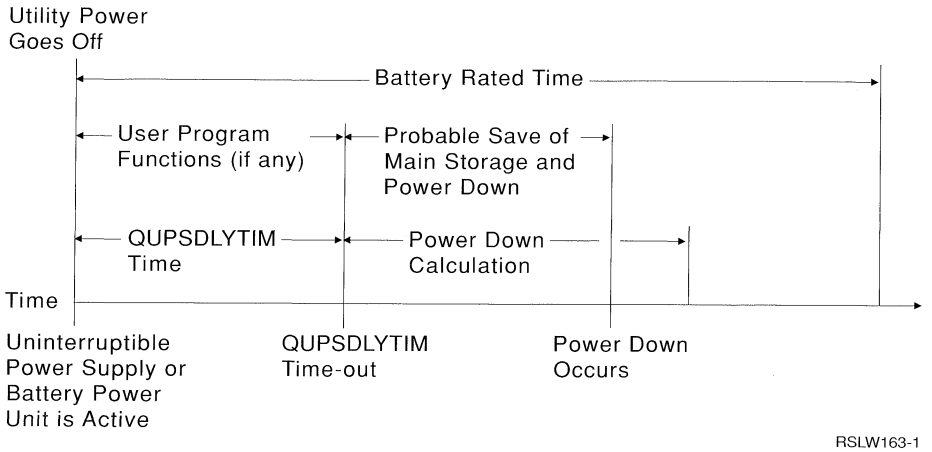


Figure 19-5. Time Line of QUPSDLYTIM Function

Normally, you would only set this value to 0 if:

- You prefer to manually start the system again.
- You have a power handling program that determines whether the batteries have not been recharged enough to allow another IPL.

The system may override a 1 value to try to protect the battery from being completely discharged. There are two cases:

- During an IPL, if utility power fails and a weak battery is signaled before the OS/400 licensed program is given control, the system saves main storage, powers down, and overrides the power restore option to 0 (Not allowed).
- When performing an IPL due to the power restore IPL value and a utility power failure occurs, if the OS/400 licensed program is not given control before the power supply delay time ends, the system saves main storage, powers down, and overrides the power restore option to 0 (Not allowed).

Power Down when Using Complete Power Supply Support

The power-down function when using power supply support operates from the Licensed Internal Code and does not involve the OS/400 licensed program. It cannot be started from a command (the PWRDWNSYS command is not run). There are multiple phases and considerations for the power-down function associated with power supply support:

- The Licensed Internal Code signals each job to end at the next internal instruction boundary. (This same function, known as forced Licensed Internal Code completion, can be started from the control panel. For more information about Licensed Internal Code, refer to the topic "Licensed Internal Code Completion" on page B-9.) Each job attempts to come to a point where an orderly end can occur.

  Usually a job is at the next internal instruction boundary or will be shortly. However, some internal instructions are long running, such as those that build access paths or create programs. A job performing one of these long-running instructions may not be able to reach an orderly end in the time allowed.

  This first phase is shorter than that performed by the forced Licensed Internal Code completion function. The time allowed for completion varies, depending on the main storage size of the system.
- If all jobs have reached a point where they can properly end, the system performs additional operations and then powers down. For example, the system must write all of the changed pages from main storage to auxiliary storage. The next IPL is normal from a Licensed Internal Code viewpoint, but is considered abnormal from the operating system viewpoint. There is no directory recovery or access path recovery required.
- The Licensed Internal Code ends jobs that cannot reach a point (in a reasonable amount of time) where they can end properly, and continues to power down. The next IPL does not involve directory recovery, but may require

some access path recovery, depending on what was happening to the job when it ended.

- In some cases the power-down function cannot complete successfully. The next IPL will be abnormal from a Licensed Internal Code viewpoint.

The power-down sequence sets the system to do an IPL automatically when utility power is restored if the system value QPWRRSTIPL is set to 1.

The system completes the power down if utility power is restored while the system is in the power-down sequence. If the QPWRRSTIPL system value is set to 1, the system automatically performs an IPL after the power down is complete.

The power-down function for power supply support always causes an abnormal IPL from an OS/400 point of view. This allows certain cleanup functions to occur, but normally will not significantly affect the IPL time.

Weak Battery Conditions

Any indication of a weak battery while on the uninterruptible power supply causes the system to save main storage and power down, no matter what your program is coded to do. If the system value QUPSDLYTIM is met, the system also automatically starts to save main storage and powers down. You can prevent this system action by not connecting the lead for the weak-battery condition.

If you do not have a weak-battery indication, you must consider the results if the utility power fluctuates. Normally, the uninterruptible power supply battery does not fully recharge unless the system has been on utility power for some time. Account for this in your program by adjusting the values for QUPSDLYTIM and QPWRRSTIPL.

For example, if power has gone off, comes back on, and then goes off again, you may want to adjust the QUPSDLYTIM to account for the fact that the battery is not fully recharged. If power has gone off and on several times in a 30-minute period, you may want to set QPWRRSTIPL to 0 (No) and issue the PWRDWNSYS command. This prevents the system from doing another IPL. You could either wait for a manual IPL or set the QIPLTIME system value to cause a timed IPL at a later point. In general, it is desirable to prevent the battery from being totally used up.

Weak-Battery Signal

If you make the connection for a weak-battery signal and get a signal that the battery is weak, the system responds by saving main storage and powering down, regardless of the setting of the system values.

If a system is still running on utility power and a weak-battery signal is received, message CPI0964 is sent. The system saves main storage and performs power-down functions only if it is already running on an uninterruptible power supply or the Battery feature.

This ensures the shortest possible IPL time in the event of a weak-battery condition. This does not assure that sufficient time remains on the battery for the system to perform its power-down sequence.

On the Battery feature, the connection always exists. You cannot override the system action, which saves main storage and then powers down.

For the uninterruptible power supply attachment, this is an optional connection that can be made from certain uninterruptible power supply attachments. This connection allows the system to automatically save main storage and then power down. For most uninterruptible power supply, it is better not to completely use up the battery. The weak battery signal helps prevent this. If you do not want the weak-battery signal, do not make the connection.

Some vendor-supplied uninterruptible power supply attachments may have an adjustable setting to allow you to set the time for a weak battery condition. Other uninterruptible power supply attachments may be adjusted at the factory. Some may have a fixed value and others may not have any.

Do not assume that the time remaining on the battery is sufficient for the system to save main storage and power down if the uninterruptible power supply has a weak-battery condition. Calculate the amount of time necessary to save main storage and power down. For more information

on setting the QUPSDLYTIM system value, see "Description of System Values for Uninterruptible Power Supply" on page 19-3.

Plan for an increase in the main storage size, the number of units in the existing racks, and the number of racks. The planning requirement is minimized by the QUPSDLYTIM value. That is, whichever occurs first (weak battery or QUPSDLYTIM) signals the system to save main storage and power down. However, the QUPSDLYTIM is a constant value and cannot change to respond to unusual occurrences involving the uninterruptible power supply. The weak battery signal can account for these with more reliability.

Uninterruptible Power Supply Messages

The uninterruptible power supply messages are always sent to QSYSOPR and then to QHST. Optionally, the same messages can be sent to a user-specified message queue.

The following describes the uninterruptible power supply messages that are sent to QSYSOPR and to the named message queue specified in the QUPSMSGQ system value (if it differs from QSYSOPR):

CPF1816: System utility power failed at &1.

CPF1817: System power restored at &1.

The system power switched to the utility source.

CPI0961: Uninterruptible power supply (UPS) no longer attached.

CPI0962: The uninterruptible power supply (UPS) is now attached.

CPI0963: System on auxiliary power.

System is currently running on auxiliary power.

CPI0964: Weak-battery condition exists.

The external uninterruptible power supply (UPS) or the internal battery indicates a weak-battery condition.

If utility power fails during this condition, the system may begin an immediate power down. See the Advance Backup and Recovery Guide, SC41-8079, and your uninterruptible power supply manual for more information.

CPI0965: Failure of battery backup feature in system unit.

There may be a failure of the battery or the battery charger for the battery backup feature in the system unit. Contact your service representative.

CPI0966: Failure of battery backup feature in expansion unit.

There may be a failure of the battery or the battery charger for the battery backup feature in the expansion unit. Contact your service representative.

CPI0973: Weak battery condition no longer exists.

The weak-battery condition for the external uninterruptible power supply or the internal battery no longer exists. See your UPS manual.

CPI0974: UPS has been bypassed.

If a utility power failure occurs, the uninterruptible power supply cannot supply system power. The system will end abnormally.

CPI0975: UPS no longer bypassed.

The uninterruptible power supply (UPS) is no longer bypassed.

CPI0976: Notification of message &1 failed.

Unable to send &1 message to message queue. &2 in library &3 specified in QUPSMSGQ system value.

CPI0981: Automatic IPL disabled.

Automatic IPL after utility power restored, specified by system value QPWRRSTIPL, was disabled for one of the following reasons:

- Utility power failed and the battery weak condition was detected during the IPL.

- Utility power failed during the IPL and the uninterruptible power supply delay time specified in system value QUPSDLYTIM was exceeded before the IPL complete.

CPI0994: System power is restored.

The system power switched to the utility source at &1. The utility power failed for &2 seconds. During this time the system was not doing any application processing.

If the utility power continues to fail, power down the system (PWRDWNSYS command).

IPL Considerations

When the system performs an IPL, the Licensed Internal Code verifies various internal switches to see if the system was correctly powered down. Only the successful completion of the Power Down System (PWRDWNSYS) command causes the AS/400 system to be correctly powered down. The Licensed Internal Code considers the IPL to be normal if the system saves main storage and completed the power-down sequence successfully; however, the OS/400 program considers the next IPL to be abnormal. If neither power-down technique completes normally, the Licensed Internal Code runs various recovery functions on the next IPL.

When an abnormal IPL occurs, the OS/400 program performs additional recovery functions. In an attended IPL, you can control some of these functions. In an unattended IPL (caused by the QPWRRSTIPL system value or a timed IPL), only the defaults can be taken.

If the Power Down System (PWRDWNSYS) command is run while power is being supplied by the Battery feature or uninterruptible power supply, the system delays writing any job logs until the next IPL. The system handles this type of PWRDWNSYS so that the amount of processing is minimized. The system will not do an IPL while operating on the 9404 System Unit Battery feature.

You can perform an IPL on the system if utility power is off and the system is operating on an uninterruptible power supply. This does not apply for a timed or remote IPL. Only a manual IPL is allowed when utility power is interrupted.

Uninterruptible Power Supply Signal Flowchart

Figure 19-6 is the logical flowchart of how the AS/400 system handles the signals from the uninterruptible power supply.

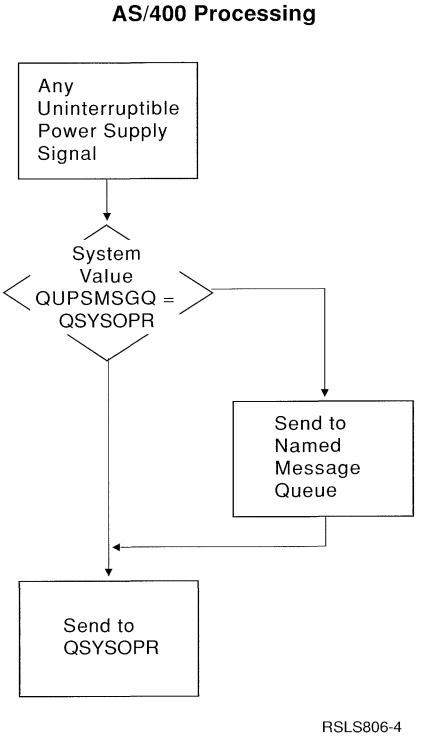


Figure 19-6. How the AS/400 System Handles Uninterruptible Power Supply Signals

Figure 19-7 on page 19-14 is the logical flowchart of how the Licensed Internal Code handles the signals from the uninterruptible power supply.

Using a Program to Handle Uninterruptible Power Supply Conditions

System software support is essentially the same for both the Battery feature and the uninterruptible power supply attachment. You control the type of processing you want to perform when utility power is interrupted through system values and an optional user-written program.

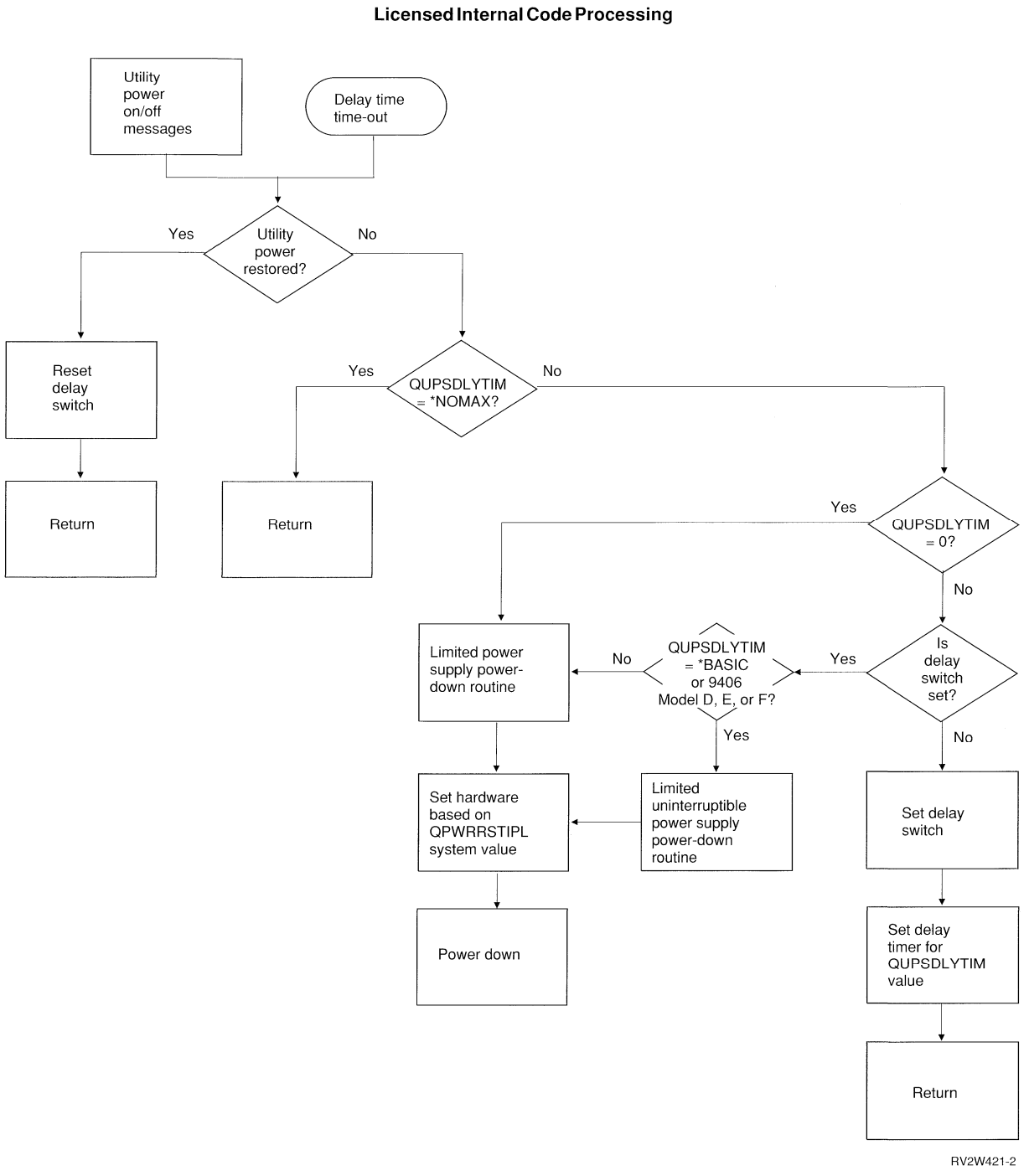


Figure 19-7. Uninterruptible Power Supply Signals Handled by Licensed Internal Code

When No User Power-Handling Program Exists

You can specify that you do not have a power handling program by using the default for the QUPSMSGQ system value, which is QSYSOPR. This causes all power-related messages to be sent to QSYSOPR. The QUPSDLYTIM system value should be set to something other than *NOMAX.

Normally, you supply power to the racks and not to all of the work station devices. When utility power is interrupted, the system remains active, but the work station jobs usually end abnormally. If utility power is restored while operating on the uninterruptible power supply, the system remains active and the work station jobs can be restarted.

If the QUPSDLYTIM timer ends or the weak-battery signal occurs, the system saves main storage and powers down. Change the QUPSDLYTIM value according to your uninterruptible power supply and your system size. This is particularly important if you do not have a weak-battery connection or if the time is fixed by the uninterruptible power supply vendor and is not appropriate for your AS/400 configuration. For more information, see "Weak-Battery Signal" on page 19-11 and "Description of System Values for Uninterruptible Power Supply" on page 19-3.

If the system is powered down while on uninterruptible power supply, the QPWRRSTIPL system value determines whether an IPL is performed when utility power is restored. The default is to not perform the IPL.

When a Power-Handling Program Does Exist

In some environments you may want to perform different actions when you begin operating on uninterruptible power supply or when power is fluctuating. A user program can handle these situations by sending specific messages to interactive users, ending batch jobs in preparation for powering down, or dynamically changing the system values that control uninterruptible power supply processing.

To specify that you have power handling programs, change the QUPSMSGQ system value to the name of a queue you have created. The same messages will be sent to both QSYSOPR and the queue you specified. Change the QUPSDLYTIM system value to *NOMAX.

The program you use to handle the message queue must be active and must allocate the queue (normally done by using the ALCOBJ command). If a program has not allocated the queue specified in QUPSMSGQ, the system will assume no power handling program exists.

Writing the Program

A power handling program should be activated at each IPL and remain active at all times. It should be accounted for in the activity level available in work management subsystem specifications.

The message queue specified in QUPSMSGQ is used for uninterruptible power supply message processing. The program normally allocates the queue by specifying the command:

```
ALCOBJ OBJ(xxx/yyy *MSGQ *EXCL)
```

When a message arrives, the critical messages to process are:

- **CPF1816: System utility power failed at &1.** (this message applies to the battery feature and full power supply)
- **CPF1817: System power restored at &1.** (this message applies to the battery feature and full power supply)
- **CPI0994: System power is restored** (this message applies to the limited uninterruptible power supply)

You can choose to ignore the other messages.

Your program can handle a brief power interruption without doing any unique processing. For example, when the CPF1816 message arrives, you can set a switch in your program indicating that the message occurred. The program could then perform a RCVMSG with WAIT(10) to cause a time-out in 10 seconds. If the CPF1817 message is received before the time-out occurs, you can reset the switch and perform no other action.

Your program can prepare for a normal power down if power is not restored after a brief time period. For example, if you have remote work stations that are still active, you may want to send them a message requesting they sign off quickly. You may want to issue ENDSBS OPTION(*CNTRLD) to prevent new work stations from signing on or new batch work from beginning. If you have batch jobs running, you may want to end them with:

```
ENDJOB   OPTION(*CNTRLD)
```

This sets an indicator to end the job. Some higher level languages and the control language allow you to test within a program to see if a controlled ENDJOB was specified. If the program does not end itself, the default on ENDJOB (30 seconds) is used.

You can set a second timer in your program, such as RCVMSG WAIT(120). If utility power has not been restored, you can issue the PWRDWNSYS

OPTION(*IMMED) command. The wait time should be specified based on your battery time and the time required for a power-down.

If you name a message queue (other than QSYSOPR) for the QUPSMSGQ system value and *NOMAX for QUPSDLYTIM, this message queue must be allocated by a program when the CPF1816 message occurs, or be in a break or notify mode if it is a work station message queue. If not, the system assumes no power handling program exists and the system will be powered down.

Note: When the system has been placed in a restricted state (for example, ENDSBS *ALL), your uninterruptible power supply handling program will no longer be active. For this reason, it is necessary to prepare an alternate method of dealing with your uninterruptible power supply and any possible power interruptions that may occur while your system is in a restricted state.

For example, when performing a SAVSYS (Save System) or RCLSTG (Reclaim Storage), your uninterruptible power supply program will no longer be active once all subsystems have been terminated since only a single workstation job will be active. As an alternative:

1. Once all subsystems have been ended, from the command line change the message queue specified in system value QUPSMSGQ to *BREAK. This will cause all uninterruptible power supply messages to be sent as break messages to the user signed on to that work station. With this method the user will manually decide what to do should a power failure occur.
2. Change the system value QUPSDLYTIM to some value other than *NOMAX (for example, the number of minutes you wish the uninterruptible power supply to ride out the power failure). This method will prevent the system from performing an immediate quick power down; however, if a power failure occurs, a quick power down will be performed should the power failure last longer than the value specified for the system value QUPSDLYTIM.
3. Modify your existing uninterruptible power supply handling program for use as a BREAK HANDLING program which may be used while the system is in a restricted state (all subsystems ended). This can be done by creating a second version of your uninterruptible power supply program that does not allocate the message queue specified in system value QUPSMSGQ (i.e., no ALCOBJ command). To utilize this program while in a restricted state, prior to starting a dedicated function such as SAVSYS, enter the command:

   ```
   CHGMSGQ MSGQ(LIB/MSGQ) DLVRY(*BREAK)
           PGM(LIB/PGM)
   ```

   where (LIB/MSGQ) is the name the message queue specified in system value QUPSMSGQ, and (PGM/LIB) is the name of your modified uninterruptible power supply handling program. Now, should a power failure occur, the power failure message will be handled by the break handling program, even while a function such as SAVSYS is running. To deactivate the break handling program either have the user sign off or enter:

   ```
   CHGMSGQ MSGQ(LIB/MSGQ) DLVRY(*HOLD)
           PGM(*DSPMSG)
   ```

   Once you have deactivated the break handling program you should immediately start your subsystems and your normal uninterruptible power supply handling program.

Running the Program

Run the power handling program with a priority higher than your normal jobs. The system normally is very busy attempting to end and clean up interactive jobs that have lost power to the work stations. You may want to hold any batch work, even during a brief power interruption.

Power-Handling Program for Full Uninterruptible Power Supply

The following is an example of implementing a power-handling program on an AS/400 system when a full Uninterruptible power supply is attached. This example assumes that QCTL is the controlling subsystem.

1. Because of the critical nature of a power-handling program, it is recommended that you isolate the objects used by the power-handling program in their own library and secure them from other users:

   ```
   CRTLIB LIB(UPSLIB) AUT(*EXCLUDE) CRTAUT(*EXCLUDE)
   ```

2. A power-handling program requires exclusive use of a message queue. For this reason, it is recommended that you create a unique message and exclude its use from all other users and general system use.

   ```
   CRTMSGQ MSGQ(UPSLIB/UPSMSGQ) AUT(*EXCLUDE)
   ```

3. Create the CL power-handling program and exclude its use from all other users:

   ```
   CRTCLPGM PGM(UPSLIB/UPSPGM) AUT(*EXCLUDE)
   ```

   Note: See "Power-Handling CL Program Flowchart" on page 19-18 and "Power-Handling CL Program Example" on page 19-18 for the power-handling program flowchart and program example.

4. Create the job description for the power-handling program you want started automatically whenever the controlling subsystem is started.

   ```
   CRTJOBD JOBD(UPSLIB/UPSJOBD) JOBQ(QSYS/QCTL2) JOBPTY(1) +
        RQSDTA('CALL UPSLIB/UPSPGM') AUT(*EXCLUDE)
   ```

   Note: If you are using a subsystem other than QCTL as the controlling subsystem, verify that JOBQ(QSYS/QCTL) is allocated by that subsystem.

5. Because the controlling subsystem is always active when the system is running, you cannot change the current controlling subsystem description. Therefore, create an alternative controlling subsystem description by making a copy of the current controlling subsystem description:

   ```
   CRTDUPOBJ OBJ(QCTL) FROMLIB(QSYS)
        OBJTYPE(*SBS) TOLIB(QSYS)  NEWOBJ(QCTL2)
   ```

6. Add the autostart job entry to the alternative controlling subsystem description.

   ```
   ADDAJE SBS(QSYS/QCTL2) JOBQ(QSYS/QCTL2)
          JOBD(UPSLIB/UPSJOBD)
   ```

7. Change the controlling subsystem system value to use the alternative controlling subsystem description.

   ```
   CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTL2')
   ```

8. Change the system values to allow the program to handle a power outage:

   ```
   CHGSYSVAL SYSVAL(QUSPMSGQ) VALUE('UPSMSGQ UPSLIB')

   CHGSYSVAL SYSVAL(QUPSDLYTIM) VALUE(*NOMAX)
   ```

9. Perform an IPL of the system:

   a. Ensure the key is in the keylock switch on the control panel.

   b. Turn the key in the keylock switch until it points to the Manual position.

   c. Power down the system:

      ```
      PWRDWNSYS OPTION(*IMMED) RESTART(*YES) IPLSRC(B)
      ```

10. If you want the system to return to the original controlling subsystem at the next IPL, add the autostart job entry to the original controlling subsystem description.

    ```
    ADDAJE SBS(QSYS/QCTL) JOBQ(QSYS/QCTL)
           JOBD(UPSLIB/UPSJOBD)
    ```

11. Change the controlling subsystem system value to use the new subsystem description.

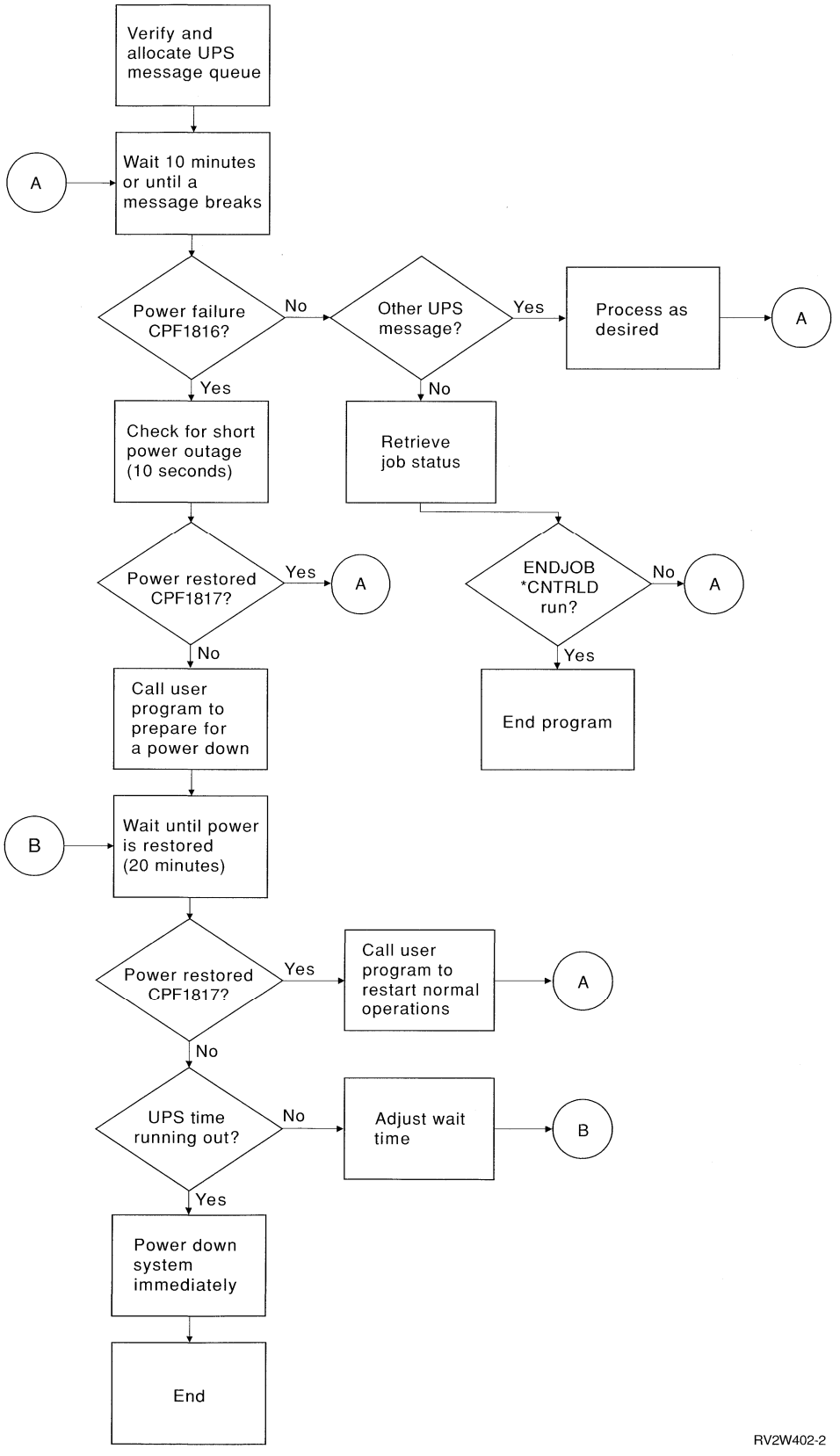
    ```
    CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTL')
    ```

    At the next IPL, the system will change the controlling subsystem back to QCTL.

Power-Handling CL Program Flowchart

Power-Handling CL Program Example

```
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...
 1.00        PGM
 2.00        DCL       VAR(&UPSMSGQ)  TYPE(*CHAR)  LEN(20)
 3.00        DCL       VAR(&LIB)      TYPE(*CHAR)  LEN(20)
 4.00        DCL       VAR(&MSGQ)     TYPE(*CHAR)  LEN(20)
 5.00        DCL       VAR(&MSGID)    TYPE(*CHAR)  LEN(7)
 6.00        DCL       VAR(&ENDSTS)   TYPE(*CHAR)  LEN(1)
 7.00        DCL       VAR(&WAIT)     TYPE(*DEC)   LEN(6)
 8.00        DCL       VAR(&HOUR)     TYPE(*DEC)   LEN(6)
 9.00        DCL       VAR(&MIN)      TYPE(*DEC)   LEN(6)
 0.00        DCL       VAR(&SEC)      TYPE(*DEC)   LEN(6)
11.00        DCL       VAR(&TIME)     TYPE(*CHAR)  LEN(6)
12.00        DCL       VAR(&START)    TYPE(*DEC)   LEN(6)
13.00        DCL       VAR(&END)      TYPE(*DEC)   LEN(6)
14.00        DCL       VAR(&RESULT)   TYPE(*DEC)   LEN(6)
15.00
16.00        RTVSYSVAL  SYSVAL(QUPSMSGQ) RTNVAR(&UPSMSGQ)
17.00        CHGVAR     VAR(&MSGQ) VALUE(%SST(&UPSMSGQ 1 10))
18.00        CHGVAR     VAR(&LIB)  VALUE(%SST(&UPSMSGQ 11 10))
19.00        DLTMSGQ    MSGQ(&LIB/&MSGQ)
20.00        MONMSG     MSGID(CPF2105) /* Message queue not found. */
21.00        CRTMSGQ    MSGQ(&LIB/&MSGQ) TEXT('UPS Power handling +
22.00                          program message queue') AUT(*EXCLUDE)
23.00        ALCOBJ     OBJ((&LIB/&MSGQ *MSGQ *EXCL))
24.00
25.00  A:    RCVMSG     MSGQ(&LIB/&MSGQ) WAIT(600) RMV(*YES) +
26.00                          MSGID(&MSGID)
27.00        IF         COND(&MSGID *NE CPF1816) THEN(DO)
28.00        RTVJOBA    ENDSTS(&ENDSTS)
29.00        IF         COND(&ENDSTS *EQ '1') THEN(GOTO CMDLBL(ENDPGM))
30.00        GOTO       CMDLBL(A)
31.00        ENDDO
32.00
33.00       /* Check to see if this is a short power outage. */
34.00       IF         COND(&MSGID *EQ CPF1816) THEN(DO)
35.00       RCVMSG     MSGQ(&LIB/&MSGQ) WAIT(10) RMV(*YES) +
36.00                     MSGID(&MSGID) /* Wait ten seconds)
37.00       IF         COND(&MSGID *EQ CPF1817) THEN(GOTO CMDLBL(A))
38.00       ENDDO
39.00
40.00       /* Power outage was longer than 10 seconds.  */
41.00       CALL       PGM(LIB/PGM) /* User program that prepares +
42.00                         system for possible shutdown. */
43.00
44.00      /* Check to see if this is a long power outage. */
45.00       CHGVAR     VAR(&WAIT) VALUE(01200) /* 20 minutes. */
```


Verify and allocate UPS message queue
A
Wait 10 minutes or until a message breaks
Power failure CPF1816?
No
Other UPS message?
Yes
Process as desired
A
Yes
No
Check for short power outage (10 seconds)
Retrieve job status
Power restored CPF1817?
Yes
A
ENDJOB *CNTRLD run?
No
A
No
Yes
Call user program to prepare for a power down
End program
B
Wait until power is restored (20 minutes)
Power restored CPF1817?
Yes
Call user program to restart normal operations
A
No
UPS time running out?
No
Adjust wait time
B
Yes
Power down system immediately
End

RV2W402-2

```
46.00  B:   RTVSYSVAL  SYSVAL(QTIME) RTNVAR(&TIME)
47.00       CHGVAR     VAR(&HOUR)  VALUE(%SST(&TIME 1 2))
48.00       CHGVAR     VAR(&MIN)   VALUE(%SST(&TIME 3 2))
49.00       CHGVAR     VAR(&SEC)   VALUE(%SST(&TIME 5 2))
50.00       CHGVAR     VAR(&START) VALUE((&SEC) + (&MIN * 60) + +
51.00                     (&HOUR * 3600))
52.00       RCVMSG     MSGQ(&LIB/&MSGQ) WAIT(&WAIT) RMV(*YES) +
53.00                           MSGID(&MSGID)
54.00       IF         COND(&MSGID *EQ CPF1817) THEN(DO)
55.00       CALL       PGM(LIB/PGM) /* User program that restarts +
56.00                     system operations. */
57.00       GOTO       CMDLBL(A)
58.00       ENDDO
59.00
60.00       IF         COND(&MSGID *NE CPF1817) THEN(DO)
61.00       RTVSYSVAL  SYSVAL(QTIME) RTNVAR(&TIME)
62.00       CHGVAR     VAR(&HOUR)  VALUE(%SST(&TIME 1 2))
63.00       CHGVAR     VAR(&MIN)   VALUE(%SST(&TIME 3 2))
64.00       CHGVAR     VAR(&SEC)   VALUE(%SST(&TIME 5 2))
65.00       CHGVAR     VAR(&END)   VALUE((&SEC) + (&MIN * 60) + +
66.00                   (&HOUR * 3600))
67.00       CHGVAR     VAR(&RESULT) VALUE(&END - &START)
68.00       IF         COND(&RESULT < 0) THEN(CHGVAR VAR(&RESULT) +
69.00                     VALUE(86400 + &RESULT)) /* Check for +
70.00                           change of day.  86400 = 24 hours. */
71.00       IF         COND(&RESULT *GE &WAIT) THEN(PWRDWNSYS +
72.00                    OPTION(*IMMED) /* UPS battery reserve has +
74.00                     expired. */
75.00       CHGVAR     VAR(&WAIT) VALUE(&WAIT - &RESULT) /* UPS +
76.00                           battery reserve has not expired. */
77.00       GOTO       CMDLBL(B)
78.00       ENDDO
79.00
80.00  ENDPGM:    DLCOBJ     OBJ((&LIB/&MSGQ *MSGQ *EXCL))
81.00             ENDPGM
```

Figure 19-8. Sample Power-Handling Program

Power-Handling Sample CL Program Notes

The sample program "Power-Handling CL Program Example" on page 19-18 is based on the power-handling CL program flowchart in "Power-Handling CL Program Flowchart" on page 19-18 and can be used with a full uninterruptible power supply (UPS). Although this sample CL program is fully functional, it should be tailored to your specific system requirements. For example additional recovery should be added to the program by monitoring for error conditions specific to your system. You will also need to supply a user written program that performs the steps necessary to prepare for a normal shutdown of the system (such as holding job queues, sending messages, ending subsystems) that restarts normal operations should the power outage end before the system is powered down.

The program performs the following:

1. The power-handling program retrieves the system value QUPSMSGQ into the variables &LIB and &MSGQ. Although, this is not absolutely necessary, it does help to ensure that the correct message queue is allocated each time the program is started. The program then deletes the message queue (if it already exists) and then creates it again.

   This step helps eliminate clearing the message queue or any problems that might occurred if the message queue is damaged.

2. After the message queue is created, the program must allocate (ALCOBJ command) the message queue exclusively.

Note: When the system value QUPSDLYTIM is set to *NOMAX, the message queue specified for system value QUPSMSGQ must be allocated either by a user using `CHGMSGQ MSGQ(QUPSLIB/QUPSMSGQ) MODE(*BREAK)`, or by using the ALCOBJ command within the power-handling program, but not by both.

If the message queue has not been allocated by a user or a program, and a power outage occurs, then the system performs an immediate quick power down.

3. At label A, in the example power handling program, the Receive Message (RCVMSG) command is used to determine what message has been sent to the message queue and the amount of wait time (WAIT parameter) throughout the program.

   On line 27.00 of the example power-handling program, the RCVMSG command with a value of 600 seconds for the WAIT parameter causes the program to wait for ten minutes. After ten minutes, the program checks to see if a controlled end to the job has been run (using the ENDSBS or ENDJOB command). This prevents the never-ending program from delaying the ENDJOB or ENDSBS command.

   If you use ENDSBS *IMMED or ENDJOB *IMMED, then this part of the program can be removed and the value for the WAIT parameter on the RCVMSG command can be changed to *MAX. Regardless of the value specified for the WAIT parameter, the RCVMSG command runs immediately if a message is sent to the message queue specified on the RCVMSG command.

4. If the message received by the RCVMSG command is CPF1816 (system utility power failed), then the program checks to see if this is simply a short power failure. The programs runs a second RCVMSG command with a value of ten seconds for the WAIT parameter (you must decide how many seconds is adequate for your site).

   If the message received by the RCVMSG command within the specified ten seconds is CPF1817 (system utility power restored), then power was restored and the program returns to label A and starts the cycle again.

   If the ten seconds is reached and no message is received, then the power failure is longer than ten seconds and additional steps are necessary. At this point, you can call a user-written program that holds certain long running batch jobs (HLDJOBQ), notifies unaffected remote users, or begins ending jobs and subsystems in an orderly manner.

5. At label B, in the example power-handling program, the program attempts to wait-out the power failure. The program retrieves the present time (used later to determine how much uninterruptible power supply time is left) and places this information into a CL variable named &START.

   A third RCVMSG command is run and the value for the WAIT parameter is determined by a CL variable named &WAIT (changed earlier in the program). The CL variable &WAIT is the amount of reserve power the uninterruptible power supply can provide. The value for the &WAIT variable at label A: should be adjusted to the number of seconds of reserve power that the uninterruptible power supply can provide.

   In the example program, the value of &WAIT variable is set to 1200 seconds (20 minutes). If message CPF1817 (System utility power restored) is received during this 1200 second wait, then power *has* been restored and another program can be called to restart normal system operations. The program then returns to label A: and starts the cycle again. If message CPF1817 is not sent after 1200 seconds, then RCVMSG returns a blank message ID (not equal to CPF1817) indicating that power has *not* been restored and a immediate system power down is started. If a message other than CPF1817 is received during this 1200 second wait, the program:

   a. Retrieves the present time, calculates how much of the 1200 seconds has been used

   b. Subtracts the difference, changes the CL variable &WAIT to reflect that amount

   c. Returns to label B: to use the remaining power provided by the uninterruptible power supply

   Notice that this part of the program checks to see if a change of date occurs, which is necessary should the power outage occur on a different date.

Testing a Power-Handling CL Program

Once a power-handling program has been created, it can be tested by creating a simple CL program that uses the SNDPGMMSG (Send Program Message) command and the Delay Job (DLYJOB) command. Simply set the DLY parameter value on the DLYJOB command to meet your testing needs. (See Figure 19-9.)

Note: When testing an uninterruptible power supply program, commands, such as PWRDWNSYS, ENDJOB, and ENDSBS should be replaced with the SNDMSG command to indicate that the command has run.

```
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7....
 1.00              PGM
 2.00              DLYJOB     DLY(120) /* Wait for 2 minutes. */
 3.00              SNDPGMMSG  MSGID(CPF1816) MSGF(QCPFMSG) +
 4.00                           TOMSGQ(UPSLIB/QUPSMSGQ) /* Power failure
 5.00                           message. */
 6.00              DLYJOB     DLY(5) /* Wait for 5 seconds. */
 7.00              SNDPGMMSG  MSGID(CPF1817) MSGF(QCPFMSG) +
 8.00                           TOMSGQ(QGPL/QUPSMSGQ) /* Power restored
 9.00                           message. */
10.00              ENDPGM
```

Figure 19-9. Testing a Power-Handling Program Example

Part 8. Appendixes

Appendix A. Licensed Internal Code SRCs That Require User Input (A6xx xxxx)

The following topics describe Licensed Internal Code SRCs that require user input (A6xx xxxx).

Function 11, Data Code A6xx 6001:

This topic describes the user input required by the Licensed Internal Code SRC A6xx 6001.

Description: Select the utility (Licensed Internal Code Install or Licensed Internal Code Restore you want to run.

Reply: Using the Select key on the control panel, select the function code for the utility you want to run, then press the Enter key:

Function Code	*Utility Selected*
23	Stand-Alone Licensed Internal Code Restore The stand-alone Licensed Internal Code restore utility copies all system Licensed Internal Code from the tape and writes over the Licensed Internal Code found on the disk. Select this utility to exchange or update an existing system's Licensed Internal Code without losing customer data already on the system.
24	Stand-Alone Licensed Internal Code Install The stand-alone Licensed Internal Code install utility deletes all information found on unit 1 (including customer data) and copies all system Licensed Internal Code from tape to disk. The data on all the remainder of the disk units will not be erased but may not be accessible (because of the way data is spread over multiple units on the system). Select this utility when initially starting up a new system (which contains no customer data and no Licensed Internal Code) or in cases where the primary disk was exchanged.
29	Load Model-Unique Licensed Internal Code The load model-unique Licensed Internal Code utility copies only the Licensed Internal Code from tape, and writes over any existing Licensed Internal Code on disk. Select this utility when a hardware model upgrade is performed.
32	Download disk Licensed Internal Code The download disk Licensed Internal Code utility downloads the Licensed Internal Code from tape to disk. Use this function when the disk Licensed Internal Code should be changed.
(None)	Cancel this request. If you do not want to select any of these options, turn the system control panel off.

Function 11, Data Code A6xx 6002:

This topic describes the user input required by the Licensed Internal Code SRC A6xx 6002.

Description: **Warning:** Do you want to destroy all data on all disk units?

The stand-alone Licensed Internal Code install was requested; however, the disk unit that contains storage unit 1 already contains data. Continuing with Licensed Internal Code install will delete all data on the disk unit that contains unit 1. The data on all the remainder of the disk units will not be deleted, but may not be accessible (because of the way data is spread over multiple units on the system).

Function codes may be used to display to which disk the system is attempting to install.

The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are enclosed in parentheses (such as 15-2).

Function Code	*Function*
14 (15-2)	Display type and model of the disk unit that contains unit 1. The first 4 characters displayed in the lights will show the type. The next 4 characters show the model. An example of what would be shown after you select this option follows. (This shows what you would see if the disk was a 9332 Model 400.) 14 (15-2) | 9332 0400
15 (16-2)	Display address of the disk unit that contains unit 1. Eight characters will be shown on the display lights. The meaning of these characters is shown below. ***Character Position*** — ***Description*** 1–2 Bus number (should be zero) 3 IOP card number 4 IOP board number 5–6 Facility address (DFCI) 7–8 Secondary address The following is an example of what would be shown after you select this option: 15 (16-2) | 0010 0700
16 (17-2)	Display the serial numbers of the disk unit that contains unit 1. The following is an example of what should be shown after you select this option: 16 (17-2) | 0012 3456

Reply: Using the Function Select key on the control panel, type one of the following codes and press the Enter key:

Function Code	*Function*
23	Restore the Licensed Internal Code without destroying customer data. Select function code 23 if you want to restore the Licensed Internal Code to the disk unit that contains storage unit 1 without deleting any other data from the system. Selecting this option will copy all system Licensed Internal Code from tape and will exchange the Licensed Internal Code found on unit 1. Select this option if you want to exchange or update an existing system's Licensed Internal Code without losing customer data.
24	Destroy all system data and restore the Licensed Internal Code. Select function code 24 if you want to restore the licensed internal code to the selected disk unit, and you want to *destroy all data* on the system. Selecting this option will first delete all information found on the disk unit that contains storage unit 1 (including customer data) and then will copy all system licensed internal code from tape to disk. The data on all the remainder of the disk units will not be deleted but may not be accessible (because of the way data is spread over multiple units on the system). Select this option if you are installing a new system (which contains no customer data and no Licensed Internal Code), or if you have exchanged the disk unit that contains storage unit 1.
(None)	Cancel this request. If you do not want to continue with the Licensed Internal Code install, turn the control panel off.

Function 11, Data Code A6xx 6003:

This topic describes the user input required by the Licensed Internal Code SRC A6xx 6003.

Description: **Warning:** Do you want to destroy all data on all disk units?

The stand-alone Licensed Internal Code install was requested. The system found a disk unit attached at the correct location for being the disk unit that contains storage unit 1. However, the disk unit found already contains data, and the data is not in the correct format to be unit 1. The wrong disk unit may be attached at the location where the disk unit that contains storage unit 1 should be, or the correct disk unit that contains storage unit 1 may not be turned on (in which case the system detected the wrong disk unit as unit 1).

Continuing with Licensed Internal Code install will delete all data on the disk unit that contains storage unit 1. The data on all the remainder of the disk devices will not be deleted, but may not be accessible (because of the way data is spread over multiple units on the system). This information can be used with the rack configuration list printouts to ensure the system will install or restore to the correct disk.

Function codes may be used to identify which disk the system is attempting to install the Licensed Internal Code. The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are enclosed in parentheses (such as 15-2).

Function Code *Function*

14 (15-2) Display type and model of the disk unit that contains storage unit 1. The first 4 characters displayed in the lights will show the type. The next 4 characters show the model.

The following is an example of what would be shown after you select this option. (This shows what you would see if the disk was a 9332 Model 400.)

14 (15-2)	9332 0400

15 (16-2) Display address of the disk unit that contains storage unit 1. Eight characters will be shown on the display lights. The meaning of these characters is as follows:

Character Position	***Description***
1–2	Bus number (should be zero)
3	IOP card number
4	IOP board number
5–6	Facility address (DFCI)
7–8	Secondary address

The following is an example of what would be shown after you select this option:

15 (16-2)	0010 0700

16 (17) Display the serial numbers of the disk unit that contain storage unit 1.

The following is an example of what should be shown after you select this option.

16 (17-2)	0012 34561

Reply: Using the Function Select key on the control panel, type one of the following codes and press the Enter key:

Function Code *Function*

24 Destroy all system data and restore the Licensed Internal Code.

Select function code 24 if you want to do the following:

- Restore the Licensed Internal Code to the selected disk unit.
- Destroy all data on the system.

Selecting this option will first delete all information found on the disk unit that contains storage unit 1 (including customer data) and then will copy all system licensed internal code from tape to disk. The data on all the remainder of the disk units will not be deleted but may not be accessible (because of the way data is spread over multiple units on the system).

Select this option if you are installing a new system (which contains no customer data and no Licensed Internal Code), or if you have exchanged unit 1.

(None) Cancel this request.

If you do not want to continue with the Licensed Internal Code install operation, turn the control panel off.

Function 11, Data Code A6xx 6004:

This topic describes the user input required by the Licensed Internal Code SRC A6xx 6004.

Description: **Warning:** The disk unit that contains storage unit 1 does not contain Licensed Internal Code.

The stand-alone Licensed Internal Code restore was requested. The system found a disk unit attached at the correct location for being the disk unit that contains storage unit 1. However, the disk unit found already contains data and the data is not in the correct format to be unit 1. The wrong disk unit may be attached at the location where unit 1 should be, or the correct disk unit that contains storage unit 1 may not be turned on (in which case the system detected the wrong disk unit as unit 1).

To restore Licensed Internal Code to this disk unit, the system must first destroy all data on the disk unit that contains storage unit 1 (to correctly format it to be a disk unit that contains storage unit 1). The data on all the remainder of the disk units will not be deleted, but may not be accessible (because of the way data is spread over multiple units on the system).

Function codes may be used to display which disk the system is attempting to restore. This information can be used with the rack configuration list printouts to ensure the system will restore to the correct disk.

The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are enclosed in parentheses (such as 15-2).

Function Code *Function*

14 (15-2) Display type and model of the disk unit that contains storage unit 1. The first 4 characters displayed in the lights will show the type. The next 4 characters show the model.

The following is an example of what would be shown after you select this option. (This shows what you would see if the disk was a 9332 Model 400.)

14 (15-2)	9332 0400

15 (16-2) Display address of the disk unit that contains storage unit 1. Eight characters will be shown on the display lights. The meaning of these characters are as follows:

Character Position	***Description***
1–2	Bus number (should be zero)
3	IOP card number
4	IOP board number
5–6	Facility address (DFCI)
7–8	Secondary address

The following is an example of what would be shown after you select this option:

15 (16-2)	0010 0700

16 (17-2) Display the serial numbers of the disk unit that contains storage unit 1.

The following is an example of what should be shown after you select this option:

16 (17-2)	0012 3456

Reply:

Function Code *Function*

24 Destroy all system data and restore the Licensed Internal Code.

Select function code 24 if you want to do the following:

- Destroy all data on the system.
- Restore the Licensed Internal Code to the selected disk unit.

Selecting this option will first delete all information found on the disk unit that contains storage unit 1 (including customer data) and then will copy all

system Licensed Internal Code from tape to disk. The data on all the remainder of the disk units will not be deleted but may not be accessible (because of the way data is spread over multiple units on the system).

Select this option if you are installing a new system (which contains no customer data and no Licensed Internal Code), or if you have exchanged the disk unit that contains unit 1.

(None) Cancel this request.

If you do not want to continue with the Licensed Internal Code restore operation, turn off the control panel.

Function 11, Data Code A6xx 6005:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6005.

Description: Disk unit that contains storage unit 1 not found.

The disk unit that contains unit 1 cannot be located. Try the following:

- For 9335 devices, ensure that the Enable/Disable switch is in the Enable position.
- Ensure that all disk units are turned on. If the devices are already turned on, turn them off, wait a minute, then turn them back on.

If the disk unit that contains storage unit 1 still cannot be located, call for service support.

Function 11, Data Code A6xx 6006:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6006.

Description: Tape Licensed Internal Code file is incompatible for restoring.

The *load Licensed Internal Code* level found on tape is not compatible with the *Licensed Internal Code* found on disk. Continuing to restore this code will result in system failure.

Reply: Using the Select key on the control panel, select one of the following codes, and press the Enter key:

Function Code	*Function*
24	Destroy all system data and install the Licensed Internal Code. Select this option if you are installing a new system (which contains no customer data) or if you have exchanged the disk unit that contains storage unit 1.
(None)	Cancel this request. If you do not want to continue with the Licensed Internal Code restore operation, turn off the system control panel.

Function 11, Data Code A6xx 6007:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6007.

Description: **Warning:** Disk unit that contains storage unit 1 does not contain Licensed Internal Code.

The stand-alone download disk Licensed Internal Code, function code 32, was requested. The system found the disk unit attached at the correct location for being a disk unit that contains storage unit 1. However, the disk unit found is not in the correct format to be a unit 1. The wrong disk unit may be attached at the location where unit 1 should be, or the correct disk unit that contains storage unit 1 may not be turned on (in which case the system detected the wrong disk unit as the disk unit that contains storage unit 1).

Function codes may be used to display which disk the system is attempting to restore. This information can be used with the rack configuration list printouts to ensure the system will restore to the correct disk.

The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are enclosed in parentheses (such as 15-2).

Function Code	*Function*
14 (15-2)	Display type and model of the disk unit that contains storage unit 1. The first 4 characters displayed in the lights will show the type. The next 4 characters show the model. The following is an example of what would be shown after you select this option. (This shows what you would see if the disk was a 9332 Model 400.)

14 (15-2)	9332 0400

15 (16-2)	Display address of the disk unit that contains storage unit 1. Eight characters will be shown on the display lights. The meaning of these characters is as follows:

Character Position	***Description***
1–2	Bus number (should be zero)
3	IOP card number
4	IOP board number
5–6	Facility address (DFCI)
7–8	Secondary address

The following is an example of what would be shown after you select this option:

15 (16-2)	0010 0000

16 (17)	Display the serial numbers of the disk unit that contains storage unit 1. The following is an example of what should be shown after you select this option:

16 (17-2)	0012 3456

Reply:

Function Code	*Function*
24	Destroy all system data and restore the Licensed Internal Code. Select function code 24 if you want to do the following: • Restore the Licensed Internal Code to the selected disk unit. • Destroy all data on the system. Selecting this option will first delete all information found on the disk unit that contains storage unit 1 (including customer data) and then will copy all system Licensed Internal Code from tape to disk. The data on all the remainder of the disk units will not be deleted but may not be accessible (because of the way data is spread over multiple units on the system). Select this option if you are installing a new system (which contains no customer data and no Licensed Internal Code), or if you have exchanged the disk unit that contains storage unit 1.
32	Forced download of disk Licensed Internal Code. Select function code 32 to force down load of the disk Licensed Internal Code to the disk unit selected.
(None)	Cancel this request. If you do not want to continue with the Licensed Internal Code restore operation, turn off the control panel.

Function 11, Data Code A6xx 6008:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6008.

Description: **Warning:** Load-source disk unit not found.

The vital product data (VPD) within the system indicates that unit 1 will be found at the address specified in the following function codes. Either the disk unit at this address did not report or the device at that address is not a disk unit.

This condition can occur because of one of the following:

1. Either the disk unit at that address is not powered on, or did not report in, or is missing. Verify the following and make any necessary corrections. The install automatically continues when the device is ready and reports in.
   - Ensure that all 9335 and 9336 disk units (if there are any) have their switches set to the Enable position.
   - Ensure that the disk unit at that address is powered on and the Ready light is on. If the disk unit is already powered on and its indicator shows a ready status, you might try powering the disk unit off, waiting a few minutes, and powering the disk unit back on. The install automatically continues when the disk unit becomes ready and reports in.
   - Ensure that any separate disk unit controller attached to the requested disk unit is powered on and ready. Do not power off the controller.
2. The AS/400 service processor has been replaced.

   The service processor card contains some of the VPD information being used. If the card has been replaced, then the VPD information will be incorrect.

   Function 27 can be used to override the VPD information and locate the default load-source disk unit.

   Warning: Read the instructions for function 27 below before taking this option.
3. The disk units on this system has been removed or cabled again, or their address switches have been changed. Ensure that the system is cabled correctly and that the disk units have the correct addresses. Notice that some disk units also contain VPD information about the system. If the units are not in the same positions, the VPD information will be incorrect.

   To locate storage unit 1, power down the system and move and cable the disk units again so that storage unit 1 and its mirrored unit have the correct address. Then run the stand-alone utility again.

   Function 27 can be used to override the VPD information and locate the default unit 1.

   Warning: Read the instructions for function 27 below before taking this option.

The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are in parentheses (such as 15-2).

Function Code — *Function*

14 (15-2) Displays the disk unit address of the candidate for storage unit 1.

The location of the candidate for storage unit 1 is displayed.

Character Position	***Description***
1–2	Bus number (should be zero)
3	IOP card number
4	IOP board number
5–6	Facility address (DFCI)
7–8	Secondary address

The following is an example of what would be shown after you select this option:

14 (15-2)	0010 0000

15 (16-2) Display the serial numbers of the disk unit that contains storage unit 1. If the serial number of the disk unit is available, it is displayed. If it is not available, zeros are displayed.

The following is an example of what should be shown after you select this option:

15 (16-2)	0012 3456

If the system has mirrored protection, 16 (17-2) and 17 (18-2) will contain the information for the other unit in the mirrored pair, if the information is available.

Reply: Press the Function Select key on the control panel until the following code appears and press the Enter key:

Function Code	*Function*
27	Ignore the VPD information. Select the default disk unit. **Using this option may result in the licensed internal code being loaded to the wrong disk unit.** **Note:** Before selecting this option, you should unload the tape, and IPL the system from disk (either A or B side) with the keylock switch in the Manual position. This action should correct the VPD information or displays are shown to indicate the problem and recommend actions. If the IPL of the system cannot be started from disk, then select this function. Select function 27 if you are unable to bring the requested disk unit online. This causes the SAU to select the default disk unit to be unit 1. On high-end systems, this is the disk unit whose address is 0010000 or 00200000. On low-end system, this is the disk unit at address 00100100. **Note:** * On low-end stage 1 systems, the disk unit must be in the bottom slot of the first tower. On low-end Version 2 systems, this restriction does not apply. In some situations, the VPD information found on the default disk unit may be used to locate the correct unit 1.
(None)	Cancel this request. If you do not want to continue installing the licensed internal code, unload the tape and either power off the system or perform an IPL of the system from disk (either A or B side)

Function 11, Data Code A6xx 6009:

This topic describes the user input required for the licensed internal code SRC A6xx 6009.

Description: **Warning:** Mirrored load-source disk unit not found.

The vital product data (VPD) within the system indicates unit 1 will be found at the address specified in the following function codes. Either the disk unit at this address did not report or the device at that address is not a disk unit.

This condition can occur because of one of the following:

1. Either the disk unit at that address is not powered on, or did not report in, or is missing. Verify the following and make any necessary corrections. The install automatically continue when the device is ready and reports in.
   - Ensure all 9335 and 9336 disk units (if there are any) have their switches set to the Enable position.
   - Ensure that the disk unit at that address is powered on and the Ready light is on. If the disk unit is already powered on and its indicator shows a ready status, you might try powering the disk unit off, waiting a few minutes, and powering the disk unit back on. The install automatically continues when the disk unit becomes ready and reports in.
   - Ensure that any separate disk unit controller attached to the requested disk unit is powered on and ready. Do not power off the controller.
2. The AS/400 service processor has been replaced.

   The service processor card contains some of the VPD information being used. If the card has been replaced, then the VPD information will be incorrect.

   Function 27 can be used to override the VPD information and locate the default unit 1. **Caution: Read the instructions for function 27 below before taking this option.**
3. The disk units on this system has been removed or cabled again, or their address switches have been changed. Ensure that the system is cabled correctly and that the disk units have the correct addresses. Notice that some disk units also contain VPD information about the system. If the units are not in the same positions, this VPD information will be incorrect.

   To locate unit 1, power down the system and move and cable the disk units again so that the load-source disk unit and its mirrored unit have the correct address. The run the stand-alone utility again.

Function 27 can be used to override the VPD information and locate the default storage unit 1. **Caution: Read the instructions for function 27 below before taking this option.**

The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are in parentheses (such as 15-2).

Function Code — *Function*

14 (15-2) Displays the disk unit address of the mirrored storage unit 1.

The location of the mirrored disk unit 1 is displayed.

Character Position	***Description***
1–2	Bus number (should be zero)
3	IOP card number
4	IOP board number
5–6	Facility address (DFCI)
7–8	Secondary address

The following is an example of what would be shown after you select this option:

14 (15-2)	0010 0000

15 (16-2) Display the serial number of the disk unit that contains mirrored unit for storage unit 1. If the serial number of the disk unit is available, it is displayed. If it is not available, zeros are displayed.

The following is an example of what should be shown after you select this option:

15 (16-2)	0012 3456

If the system has mirrored protection, 16 (17-2) and 17 (18-2) will contain the information for the other unit in the mirrored pair, if the information is available.

Reply: Press the Function Select key on the control panel until the following code appears and press the Enter key:

Function Code — *Function*

27 Ignore the VPD information. Select the default disk unit.

Warning: Using this option may result in the licensed internal code being loaded to the wrong disk unit.

Note: Before selecting this option, you should unload the tape, and IPL the system from disk (either A or B side) with the keylock switch in the manual position. This action should correct the VPD information or displays are shown to indicate the problem and recommend actions. If the IPL of the system cannot be started from disk, then select this function.

Select function 27 if you are unable to bring the requested disk unit online. This causes the install to continue without the mirrored unit for storage unit 1.

(None) Cancel this request.

If you do not want to continue installing the licensed internal code, unload the tape and either power off the system or perform an IPL of the system from disk (either A or B side).

Function 11, Data Code A6xx 6010:

This topic discusses the user input for licensed internal code SRC A6xx 6010.

Description: **Warning:** Load-source disk unit not found.

The disk unit for storage unit 1, specified in the following function codes, indicates that it is a part of a mirrored pair. Because of the available information, the stand alone utility is unable to verify that the disk unit selected is in the correct mirrored state.

Warning: Continuing this procedure may cause the Licensed Internal Code to be loaded to a suspended or resuming mirrored pair.

Either the vital product data (VPD) within the system indicates that the disk unit specified in the following function codes is unit 1 or a request was made to override the VPD by previously entering

function 27 in response to SRCs A6xx 6008 or A6xx 6009.

This condition can occur because of one of the following:

1. The AS/400 service processor has been replaced.

   The service processor card contains some of the VPD information being used. If the card has been replaced, then the VPD information will be incorrect.

2. The disk units on this system has been removed or cabled again, or their address switches have been changed. Ensure that the system is cabled correctly and that the disk units have the correct addresses. Notice that some disk units also contain VPD information about the system. If the units are not in the same positions, this VPD information will be incorrect.

3. A request to override the VPD information was previously done and the default disk unit location was selected.

The function codes are different for systems with Version 2 hardware. The Version 2 hardware function codes are identified with a **-2** and are in parentheses (such as 15-2).

Function Code — *Function*

14 (15-2) Displays the disk unit address of the candidate for storage unit 1.

The location of the candidate for storage unit 1 is displayed.

Character Position	***Description***
1–2	Bus number (should be zero)
3	IOP card number
4	IOP board number
5–6	Facility address (DFCI)
7–8	Secondary address

The following is an example of what would be shown after you select this option:

14 (15-2)	0010 0000

15 (16-2) Display the serial number of the disk unit that contains storage unit 1. If the serial number of the disk unit is available, it is displayed. If it is not available, zeros are displayed.

The following is an example of what should be shown after you select this option:

15 (16-2)	0012 3456

If the system has mirrored protection, 16 (17-2) and 17 (18-2) will contain the information for the other unit in the mirrored pair, if the information is available.

Reply: Press the Function Select key on the control panel until the following code appears and press the Enter key:

Note: Before selecting this option, you should unload the tape, and perform an IPL of the system from disk (either A or B side) with the keylock switch in the manual position. This action should correct the VPD information or displays are shown to indicate the problem and recommend actions. If the system cannot perform an IPL from disk, then select this function.

Function Code — *Function*

27 Ignore this warning and continue the install. To continue installing the Licensed Internal Code, press the Function Select key until 27 appears. Use the function codes to display the disk unit where the Licensed Internal Code will be installed.

(None) Cancel this request.

If you do not want to continue installing the Licensed Internal Code, power off the control panel.

Function 11, Data Code A6xx 6011:

This topic discusses the user input for Licensed Internal Code SRC A6xx 6011.

Description: Verify selection to load model-unique Licensed Internal Code.

The Load Model-Unique Licensed Internal Code Utility was selected. Continuing this procedure will

copy the model-unique Licensed Internal Code from tape to disk. Any existing model-unique Licensed Internal Code existing on disk will be exchanged.

Using the Function Selection switch on the control panel, select one of the following codes and press the Enter button on the control panel.

Function Code	*Function*
23	Load the model unique Licensed Internal Code. Press the Function Select key until 23 appears to continue exchanging the model-unique Licensed Internal Code disk with the model-unique Licensed Internal Code on tape.
29	Cancel this request. Select option 29 if you do not want to load the model-unique Licensed Internal Code. After selecting this option, SRC A6xx 6001 occurs again.

Function 11, Data Code A6xx 6030:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6030.

Description: Primary disk unit is not ready or not operational.

Reply: Prepare the disk unit. See the *System Operator's Guide*, SC41-8082, if you need more information on how to prepare the disk.

Function 11, Data Code A6xx 6041:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6041.

Description: Tape device is not operational.

Reply: See the *System Operator's Guide*, SC41-8082, for the specific type of tape unit and for an explanation of how to make the tape device operational.

Function 11, Data Code A6xx 6042:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6042.

Description: Tape device is not ready.

The 3rd and 4th characters in the SRC give further information on why the tape unit is not ready.

Code	*Definition*
33	Load assistance
37	Cartridge length check
3B	Volume removed early
43	Tape not placed in tape unit
FE	Tape unit turned off
FF	Tape unit turned off

Reply: Prepare the tape device.

See the *System Operator's Guide*, SC41-8082, for the specific type of tape device and for an explanation of how to prepare the tape device.

Function 11, Data Code A6xx 6043:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6043.

Description: Tape device load failure occurred.

The 3rd and 4th characters in the SRC give further information on why a load failure occurred.

Code	*Definition*
11	Door open
12	Reel missing
13	Reel inverted
14	No BOT marker
16	Load failure
1B	Address switch changed
1C	Address switch failure

Reply: Prepare the tape device.

For an explanation of how to make the tape device ready, see the device operator's guide that came with your tape unit.

Function 11, Data Code A6xx 6048:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6048.

Description: New tape volume needs to be installed.

Reply: Insert tape volume, and then prepare the tape device. After the tape is loaded and ready, the system will start reading this tape and continue with the install (or restore) process.

The 3rd and 4th characters in the SRC tell which volume should be loaded.

Code	*Definition*
01	Volume 1
02	Volume 2
03	Volume 3
.	
.	
.	
nn	Volume nn

For an explanation of how to insert a tape, see the device operator's guide that came with your tape unit.

Function 11, Data Code A6xx 6049:

This topic describes the user input required for the Licensed Internal Code SRC A6xx 6049.

Description: Insert correct tape volume.

Reply: You already inserted a new tape volume, but it was not the correct volume.

Code	*Definition*
01	Volume 1
02	Volume 2
03	Volume 3
.	
.	
.	
nn	Volume nn

For an explanation of how to install a tape, see the device operator's guide that came with your tape unit.

Function 11, Data Code A6xx 6051:

This topic describes the user input required for the model-unique Licensed Internal Code SRC A6xx 6051.

Description: Load tape volume containing model-unique Licensed Internal Code located in the system service kit.

Reply: Insert the tape volume containing the model-unique Licensed Internal Code into the alternative IPL tape device, and make the tape device ready. After the tape is loaded and the device is ready, the system will start reading this tape and load the model-unique Licensed Internal Code.

Function 11, Data Code A6xx 6052:

This topic describes the user input required for the model-unique Licensed Internal Code SRC A6xx 6052.

Description: The inserted tape did not contain the model-unique Licensed Internal Code.

Reply: Place the model-unique Licensed Internal Code tape into the tape device used for the IPL, and prepare the tape device.

Appendix B. Initial Program Load (IPL) Process

This appendix describes the processing that can be performed during an IPL of your system. The AS/400 system provides a number of automatic recovery functions that are performed during the next IPL after an abnormal end of the system.

The following circumstances significantly affect an abnormal, or recovery, IPL time:

- The number of access paths that must be rebuilt
- The size of auxiliary storage
- The size of main storage
- The number of add and delete operations performed on files
- The number of objects on the system
- The number of journals on the system

The topics that follow describe in detail the recovery processing that can be performed relative to the above circumstances. This will help you understand why an abnormal IPL can take much longer than a normal IPL, and what recovery the system will automatically do for you.

Time Considerations for an Initial Program Load (IPL)

To shorten the time necessary to do an IPL of the system, do all of the following:

- Install an uninterruptible power supply. An uninterruptible power supply continues to run the system should main power fail and avoids having to load the system programs altogether.
- Specify both checksum protection and access path journaling.
- Save access paths.
- Use user auxiliary storage pools (ASPs). For best protection, place journal receivers and save files in different user ASPs than the files and journals.
- Use the delayed power off completion to power down a system that is not operating.
- Use the End Job Abnormal (ENDJOBABN) command to stop jobs that do not respond to an End Job (ENDJOB) request.
- If the configuration is stable, change the IPL performance system value QPFRADJ to '0'.

Description of Initial Program Load (IPL) Processes

The following list contains the steps the AS/400 system uses to do an IPL. IPL is divided into these stages:

- Service processor IPL
- Licensed Internal Code IPL
- OS/400 IPL

Only the Licensed Internal Code IPL and OS/400 IPL steps perform special recovery processing after an abnormal end of the system. Therefore, only discussions of the recovery processing that may be performed by these two steps is provided.

Licensed Internal Code IPL

The Licensed Internal Code stage of the IPL communicates with the operator through status system reference codes (SRCs) on the control panel. Each step has a unique SRC associated with it. Some of the steps may not be performed or are performed so quickly that the operator may not see the status SRCs on the control panel.

IPL processing for the Licensed Internal Code performs the following recovery processing steps during the next IPL after an abnormal end of the system.

- **SRC C6xx 4021** Recovery (storage management recovery startup)

  This SRC will be shown briefly while some recovery start up is performed.

- **SRC C6xx 4204** Recovery (main storage dump synchronization)

  This step is started if mirrored protection is in effect for the system ASP and a prior IPL resulted in a main storage dump being taken. The main storage dump was written to only one of the disk units of a mirrored pair. During this step, the main storage dump is copied to the other disk unit of a mirrored pair.

- **SRC C6xx 4205** Recovery (mirror synchronization)

  This step is performed when mirrored protection is in effect and one or more of the following conditions exists:

  - Data on one of the load source disk units (unit 1) was changed before the other load source disk unit was available for use. In this case, the changes made to the first load source disk unit are copied to the second load source disk unit. However, if the status of one of the load source disk units is *suspended*, the changes are not copied until the *suspended* load source disk unit is *resuming*.
  - The status of one of the disk units of a mirrored pair is *active* and the status of the other disk unit of the same mirrored pair is *resuming*. In this case, the data is being copied from the *active* disk unit to the *resuming* disk unit.
  - An abnormal end of the system occurred while one or more write operations were in progress for a disk unit that has mirrored protection. In this case, the Licensed Internal Code ensures that the areas of the disk unit being written contain the same data by copying the data from one disk unit of the mirrored pair to the other disk unit of the same mirrored pair.

- **SRC C6xx 4210** Recovery (checksum protection – subset validation)

  After an abnormal end with checksum protection in effect, and when 4230 Recovery is not required to start again, checksum data must be validated. If a copy of main storage is available on unit 1, Licensed Internal Code can determine what operations were in progress and complete them to assure that all checksum data is correct. This part of the validation process requires little time. If a copy of main storage is not available on unit 1, complete validation is done later with 4260 or 4270 Recovery.

- **SRC C6xx 4220** Recovery (checksum configuration change)

  This step is started when checksum protection is first started or when a new disk unit is added to an existing checksum set. When checksum protection is started, the system will put each unit in a checksum set. The number of units in checksum sets determines how long this process will be.

  When checksum protection is first configured, the following occurs:

  - All disk units in the system ASP must be formatted.
  - The proper disk layout must be established (such as checksum stripes, protected storage areas, unprotected storage areas, control information).
  - The initial checksum data must be calculated and written to disk.

  When a disk unit is added to an existing checksum set, the following must be done:

  - The disk unit must be formatted.
  - The layout of the previously existing disk units in the set must be adjusted (although these previously existing disks are not initialized).

- **SRC C6xx 4230** Recovery (checksum protection – data recovery)

  The Save Disk Unit Data option of the disk device recovery function of DST allows a service representative to copy the contents of a disk unit. It is not intended as a method for normal backup. It is important that a service representative attempts to copy data from a failed unit before the system IPL is done. If it is successful in copying data from a failed unit, recovery time for the next IPL is significantly reduced by copying the saved data back to the replacement unit.

  The Save Disk Unit Data option marks unrecoverable sectors with a special code that 4230 Recovery uses to start the rebuilding of data in that sector. If it is not run, or if it fails to complete, all protected sectors are rebuilt on the replacement unit.

  If a disk unit must be replaced, and the save operation is not entirely successful, the data in sectors that could not be saved must be rebuilt from checksum data stored on other disk units in the set.

  Checksum protection does not cancel the requirement to recover access paths. It primarily avoids auxiliary storage initialization

and reloads from a backup copy after loss of data on a failed disk unit.

- **SRC C6xx 4240** Recovery (reclaim main storage copy)

  If a copy of main storage is available on the disk unit that contains storage unit 1, changed pages in main storage are written to their proper locations on disk.

- **SRC C6xx 4250** Recovery (subset directory recovery)

  If 4240 is successful, the storage management directories are updated. If this step is successful, 4260 is unnecessary.

- **SRC C6xx 4260** Recovery (storage directory rebuild)

  Storage management maintains two directories to keep track of disk use: the **permanent directory** (for all permanently-stored objects) and the **free-space directory** (to keep track of free space on disk). An entry in the permanent directory consists of a segment identifier (SID) and up to 4 blocks of sectors next to each other (called *extents*). An entry in the free-space directory consists of a single extent only (no SID.)

  This recovery step starts simultaneous processes to rebuild directories. Each process reads 64 sectors and then decides if the first sector is the beginning of an object extent or a free-space extent.

  If the sector is the beginning of an object extent, an entry is made in the permanent directory. Because the first sector of an object extent includes the number of sectors in the extent, processing continues at the sector following the extent.

  If the extent is the beginning of a free-space extent larger than 4M, 8M, or 16M, an entry is made in the free-space directory. Because the length of the sector is recorded in the first sector, the next read operation starts at the sector following the free-space extent.

  If the sector is the beginning of a free-space extent of less than 4M, the size of the extent is not recorded anywhere. All sectors in the extent must be read to find the end. When the end is found, an entry is made in the free-space directory, and the process continues.

  Notes:

  1. A free-space extent of greater than 16M is recorded as multiple extents of 16M or less.
  2. If 4260 Recovery is done, 4025 Recovery is also done.
  3. If the last system end was abnormal and a copy of main storage was saved to the disk unit that contains storage unit 1, the system will attempt to use the copy to shorten this step.
  4. If the last system end was abnormal and the system was running with checksum protection in effect, and checksum recovery (4230) was not required; then checksum data validation must be run. If a copy of main storage was saved, little time is required to validate checksums. Otherwise, step 4260 can run up to three times longer to perform validation. In rare cases where 4260 is not required after an abnormal end, checksum validation is performed by a special checksum validation step (4270).

     This recovery step initializes disk devices as needed if ASP configuration changes require it.

- **SRC C6xx 4260 or 4270** Recovery (checksum protection – validation)

  After an abnormal end with checksum protection in effect, and when 4230 Recovery is not required to start again, the checksum data must be validated. If a copy of main storage is not available on the disk unit that contains storage unit 1, a complete validation is done during 4260 Recovery. This validation takes up to three times as long as 4260 Recovery without checksum protection because adjacent blocks of sectors that could contain permanent data or checksum data cannot be bypassed.

  In the rare cases for which complete checksum data validation is required but 4260 Recovery is not required, the validation is performed during 4270 Recovery.

- **SRC C6xx 4280** Recovery (get space on the disk unit that contains storage unit 1)

  Main storage is copied to the load source device on selected abnormal ends related to external power failures with an uninterruptible

power supply installed, or related to failures of disk units.

To provide space for this copy, space as large as the system main storage is allocated on the disk unit that contains storage unit 1. This process normally takes place at the following times:

- First IPL after increasing the size of main storage.
- First IPL after initializing the disk unit that contains storage unit 1 and installing the system Licensed Internal Code.

- **SRC C6xx 4023** Recovery (rebuild damaged contexts)

  When an object is created, renamed, or deleted in a library, or if an object is moved between two libraries, and if an abnormal end occurs before the library pages are forced to auxiliary storage, the contexts involved require rebuilding. The amount of time for potential error is usually quite small, so context rebuilding is rarely required.

  If rebuilding is required, the permanent directory is read to locate all objects. The first page of each object is read to get the library SID. This library is then checked to verify an entry for that object. If the object is not in the library, an entry is made for it. If the library itself does not exist, the object can be recovered with the RCLSTG command.

- **SRC C6xx 4025** Recovery (authorization rebuild)

  User Profile Recovery

  This step is done every time the system performs an IPL. Any partially completed user profile table or index rebuild operations are completed. Any user profiles that have not been reformatted to a capacity of approximately 1 000 000 entries are reformatted.

  Recovery From an Abnormal End

  This recovery step is started for any of the following reasons:

  - The system performs an IPL for the first time with Version 2, Release 1 (or later) of the Licensed Internal Code.
  - The system performs an IPL following an abnormal system end, where the machine interface (MI) boundary was not reached.
  - The system performs an IPL and either 4260 Recovery or permanent address recreation is done.

  Because the above conditions can result in storage accounting inaccuracies, all object sizes and auxiliary storage owned by user profiles are suspicious until verified or corrected by the system during normal usage.

  The object size verification or correction occurs the first time the object is referenced in an MI instruction following the abnormal system end. User profile owned auxiliary storage is verified or corrected by the first use of either the MATAUOBJ or RECLAIM MI instruction; until that occurs, the amount of auxiliary storage owned by a given user profile may appear incorrect.

  The following VLOG entries may be created:

  0B00 0009 The current IPL is considered abnormal for any of the reasons previously mentioned. Object size and auxiliary storage owned by user profiles will be verified and corrected as they are used.

  0B00 0001 An object's size was corrected.

  0B00 0008 An object's associated space size was corrected.

  0B00 0002 The auxiliary storage owned by a user profile was corrected.

  0B00 000B The auxiliary storage owned by a user profile was verified and found to be corrected.

  The following information is used to perform authority recovery the first time the object is referenced.

  - The system maintains the IPL number of the last abnormal IPL. This is set to the current IPL number when the IPL is considered abnormal for any of the reasons previously mentioned.
  - Each user profile contains a field totalling the amount of recovered auxiliary storage owned by the user profile. This field is reset to zero during each abnormal IPL. The field that contains the *materialized* value for the amount of owned auxiliary storage is left alone, and may or may not be accurate.

– The header of each object contains the object authority recovery IPL number. If this number is less than the system abnormal IPL number, then when the object is first referenced it is verified or corrected as follows:

  - The object size is corrected if it is incorrect.
  - The object size is added to the *recovered* field for the user profile that owns the object.
  - The object authority recovery IPL number is updated to contain the current IPL number.

  As *recovered* objects are truncated, extended, or deleted, or the owning user profile is changed, changes are made to both the *recovered* and *materialized* fields for the auxiliary storage of the owning user profile.

– Each user profile contains the IPL number under which the auxiliary storage owned by the user profile was verified or corrected. If that IPL number is less than the last abnormal system IPL number when the MATAUOBJ instruction requests owned objects, then the auxiliary storage owned by the user profile is verified and corrected.

  1. The size of each object owned by the user profile is verified or corrected as described above if it has not been done since the last abnormal IPL.
  2. The *recovered* owned auxiliary storage size replaces the *materialized* owned auxiliary storage size. The amount of owned auxiliary storage will now appear correct.
  3. The IPL number in the user profile is updated with the current IPL number to indicate that the auxiliary storage owned by the user profile has been verified or corrected.

  When the RECLAIM instruction is run, the above is done for all user profiles that have not been verified or corrected since the last abnormal IPL.

Object Domain Initialization

If this is an install IPL, and object domains have not been initialized, a background job is started to initialize the domain for each object that has not already had its domain established. It is hoped that this will complete asynchronously during installation in order to avoid synchronous initialization on the next IPL.

If this is not an install IPL, and object domains have not yet been initialized, then the initialization takes place synchronously in this recovery step. In addition, all object size and user profile owned auxiliary storage verification and correction takes place synchronously as well.

- **SRC C6xx 4026** Recovery (journal recovery)

  This recovery step validates the network of objects that supports journaling. It verifies that each file being journaled has a usable journal and usable journal receivers. For two receivers, the system will select which receiver to use for possible recovery.

- **SRC C6xx 4027** Recovery (database recovery)

  The system determines which database files and access paths are not synchronized. If the last system end was abnormal and a copy of main storage was saved, the copy will be used for this analysis.

  The Licensed Internal Code uses an internal recovery table known as the **database in-use table** to limit the number of database files subjected to recovery. This table resides on disk and is readable.

  If the database in-use table is usable:

  – Use it to locate all database files that were open at the last system end and to verify whether the physical files and access paths are synchronized.

  – For those not synchronized, mark the access path as not valid (to be rebuilt later).

  – If any delete operations have been done since the last time the physical file was forced to disk, read all the records in the file to get an accurate count.

  – If any add operations have been done since the last time the physical file was

forced to disk, read all recently added records to get an accurate count.

Because the database in-use table resides on disk, it is possible that a portion of the disk surface containing this table is no longer readable.

Approximately 90 seconds are required to read each 16M of records.

If the database in-use table is *not* usable, locate all database objects by reading the object directory built during 4025 Recovery and perform the steps for 4027. If 4025 Recovery found it unnecessary to build an object directory, build it now; then perform the steps for 4027.

- **SRC C6xx 4028** Recovery (journal synchronization)

  A **journal synchronization point** is caused when a file is closed. This recovery step locates the physical files being journaled by each journal, locates the last synchronization point for each physical file, then starts an APYJRNCHG command.

  The journaling function automatically limits the number of journal entries that must be applied during recovery. by establishing implicit synchronization points as appropriate.

  If access path journaling is in effect, the journal is read from the last synchronization point for the journaled access path. This is done before images presenting the original state of access path pages are written to the existing access path, thereby returning it to the condition existing at the last synchronization point. In addition, a delayed maintenance log entry is made for each journal entry following any synchronization point, which represents an access path change. Thus, access path rebuilding during the OS/400 IPL becomes a delayed maintenance rebuilding of the changes that occurred since the last synchronization point.

- **SRC C6xx 4029** Recovery (commitment control recovery)

  The system performs a rollback operation for any commitment definition that was active at the time the system ended abnormally. This rollback operation backs out the pending record-level changes only for these commitment definitions. Object-level changes and API commitment resources are recovered later in the OS/400 portion of the IPL, during SRC C9xx 2AA0.

- **SRC C6xx 4030** Recovery (update object recovery list)

  The object recovery list is a table built during this stage of the IPL. It serves as the repository for information about objects that the OS/400 IPL must recover. During this step of the IPL, the table is updated to identify database objects that will be recovered.

- SRC C6xx 4031 Recovery (journal cleanup)

  This recovery step reinitializes the journal's internal tables to show that all journaled files have been synchronized.

- SRC C6xx 4032 Recovery (commitment control cleanup)

  This recovery step empties commit blocks of runtime values, such as locks.

OS/400 Initial Program Load (IPL)

The OS/400 part of the IPL communicates status to the operator by displaying system reference codes (SRCs) on the control panel. Each step has a unique SRC.

During an attended OS/400 IPL, the system also communicates with the operator through messages that appear as *BREAK messages sent to the QSYSOPR or as status messages sent to the work station message queue. If the SEVERITY filter is set too low on QSYSOPR, the messages are shown, and the IPL process is suspended until the operator exits the message display by pressing the Enter key or F3. When the controlling subsystem is started, QSYSOPR is left in the *BREAK type of operation to the console.

IPL processing for the OS/400 program performs the following processing steps.

1. **SRC C9xx 28C0** verifies the work control block table.

   The work control block table has an entry for each job on the system. Each entry contains pointers to the following permanent objects:

   - Job message queue
   - Spool control block
   - Local data area

This recovery step first checks the work control block table for damage.

If the work control block table has no damage, this recovery step verifies the existence of the job message queue and spool control block for the IPL start control program function (SCPF) job, then continues. The SCPF job does not have a local data area.

If the work control block table has damage, or if clear job and output queues was specified during the OS/400 install operation, this recovery step rebuilds the work control block table with an entry for only the IPL SCPF job. In addition, because the work control block table no longer relates the job message queues, spool control blocks, and load data areas to specific jobs, they must be deleted. These objects are not in a library so the QSYS user profile is searched to find them for later deletion during the work control block table processing step.

2. **SRC C9xx 28C0** retrieves the object recovery list.

   The object recovery list was built during the Licensed Internal Code IPL and contains the information regarding objects that the IPL SCPF job must deal with during its portion of IPL.

3. **SRC C9xx 2910** verifies selected message queues.

   The message handler validates the existence of the QSYSOPR and QHST message queues, and re-creates those it finds missing or damaged. If clear job and output queues has been specified, both message queues are re-created. The QSYSOPR message queue is cleared and its size reduced to its original allocation.

   The QSYSOPR message queue is also reduced to its original allocation whenever all messages are cleared (that is, by pressing F13 on the Message Queue display).

4. **SRC C9xx 2920** – library cleanup.

5. **SRC C9xx 2940** – console configuration. Creating QCTL and QCONSOLE and varying on during an attended IPL.

6. **SRC C9xx 2960** – attended IPL processing. Attended IPL processing occurs at this point and the IPL displays appear. For more information, see the discussion on starting the system in the *Operator's Guide*.

7. **SRC C9xx 2970** – recovers databases.

   This recovery step verifies the object recovery list and sends a message to QHST for each database file that was being used or changed during an abnormal system end, and has not yet been recovered.

   This recovery step sends a message to QHST for each journal object and each commitment control object that could not be successfully recovered. It also writes the commit ID to the user's notify object.

   This recovery either rolls back or completes certain uncompleted operations. Each of the following functions keeps an audit trail of its progress in the QRECOVERY library to be used in case of an abnormal system end.

   - Backed out:
     - Add member
     - Create file
   - Completed:
     - Change file
     - Change object owner
     - Delete file
     - End journal access path
     - End journal physical file
     - Grant object authority
     - Start journal access path
     - Start journal physical file
     - Move object
     - Recover SQL view
     - Rename member
     - Remove member
     - Rename object
     - Restore file
     - Revoke object authority

8. **SRC C9xx 2970** – recovers access paths.

   Journaled access paths are recovered at this time. A *STATUS message appears for each access path as it is being built. Recovery can usually be accomplished very quickly because only the changes that have occurred since the last synchronization point need to be applied.

   This recovery step shows the Edit Rebuild of Access Paths display only during an attended IPL, and only if files with keyed access paths are in a nonsynchronized condition (they require rebuilding) and were not journaled.

Any open keyed access path that has had at least one change will generally be in a non-synchronized condition unless the access path is journaled. Functions that can force synchronization include:

- RPG III FEOD operation
- FRCRATIO
- FRCACCPTH
- Full CLOSE

Each of the functions just listed causes the index header page to also be forced to disk with the synchronized bit set on. On the next unforced index change, the index header page is again forced to disk with the synchronized bit set off.

```
                    Edit Rebuild of Access Paths                RCHAS331
                                                        05/12/90  13:49:34

IPL threshold . . . . . . . . . . . . . 88_  0-99

Type sequence, press Enter.
Sequence:  1-99, *OPN, *HLD

                    ------------Access Paths----------  Unique  Estimate
Seq   Status        File        Library     Member      Keyed   Time
25__  IPL           FILE234512  LIBRARY111  MBR1234567  No      00:00:56
25__  IPL           FILE234513  LIBRARY111  MBR1234567  No      00:00:56
75__  IPL           FILE234514  LIBRARY111  MBR1234567  Yes     00:00:56
75__  IPL           FILE234515  LIBRARY111  MBR1234567  Yes     00:00:56
88__  IPL           FILE234516  LIBRARY111  MBR1234567  No      00:00:56
99__  AFTIPL        FILE234517  LIBRARY111  MBR1234567  Yes     00:00:56
*OPN  OPEN          FILE126789  L123456789  MBR4567890  Yes     12:34:56
*OPN  OPEN          FILE346789  L123456789  MBR4567890  No      12:34:56
*HLD  HELD          F123336789  L123456789  MBR4567890  No      10:30:06
*HLD  HELD          F123456789  L123456789  MBR4567890  Yes     99:56:01
                                                          More...

F5=Refresh   F11=Display member text   F13=Repeat all   F15=Sort by
F16=Repeat position to   F17=Position to
```

For additional information on recovering access paths, refer to the *Database Guide*.

On the Edit Rebuild of Access Paths display, the user can select which access paths are to be rebuilt during the IPL and which can be built later. Those paths that must be rebuilt during the IPL can be done at a rate of at least 10 000 index entries per minute (this rate can vary significantly, depending on key size, processing unit speed, and the amount of main storage available for the function).

Rebuilding access paths at IPL time is always the fastest because no other processes are running, and no main storage pools have yet been established (that is, all of main storage is available for the function). The estimated rebuild times apply only to access paths that are rebuilt during the IPL when the system is in a restricted state.

Rebuilding access paths after the IPL is usually the slowest because it is done in the BASE pool (usually the smallest pool). The function must share the processing unit with other processes and is running at low priority.

Rebuilding access paths at OPEN time is done in the pool of the requesting process and so may be faster than rebuilding after the IPL.

Because rebuilding tends to be process-bound, no benefit can be expected from attempting multiple rebuild processes at the same time on a single processor.

9. **SRC C9xx 2990** – adjusts pool sizes and activity levels, if requested.

   If the system value QPFRADJ is set to '1' (the default), the machine pool, *BASE pool activity level, pool and activity level for spooled jobs, and pool size and activity level for interactive jobs are determined. Subsystem descriptions QSPL, :QINTER, and QBASE are changed to use *INTER and *SPOOL shared pools. The shared pools are changed to reflect the calculated values.

10. **SRC C9xx 29B0** – processes spooled files.

    The QSPL library is searched to get the names of all files and members in those files. Spooled file data is stored within members of a physical file in the QSPL library. When a spooled file is no longer in use, the associated member is kept for possible reuse by other spooled files (thus avoiding the overhead of Remove Member (RMVM) and Add Physical File Member (ADDPFM) commands). However, if a spooled file is not reused after seven IPLs, it is removed. To control the extra time required for this, up to 200 members are removed on each IPL, and an *STATUS message appears during the process. If the last system end was abnormal, this removal process is bypassed.

    All libraries on the system are searched to get the names of all job queues and all output queues. The status in the object header for each queue is updated from this information.

    If the default output queue associated with a device has been damaged, the output queue will be deleted and re-created by using the CRTOUTQ command.

11. **SRC C9xx 29C0** – processes work control block table.

This step reorganizes the work control block table:

- If the work control block table was not rebuilt:

  Verify if the three permanent objects exist for each work control block table entry. For the objects that represent jobs, determine the status or location of the job. If a job was active, close all related spooled files. If clear incomplete job logs was specified on the IPL Options display, the job message queue is cleared for each job that was active at the last system end.

  Compress the work control block table to improve the job search time and reduce the storage used (delete the unneeded job message queues, spool control blocks, and load data areas). During normal operations, the work control block table grows to the size necessary to accommodate the greatest number of *JOBQ, *OUTQ, and active jobs at any one time. Compression occurs only during the IPL, and then only to value specified for the initial total number of jobs (QTOTJOB system value) and sufficient additional total number of jobs (QADLTOTJ system values) to account for all jobs that will exist when the IPL is complete (primarily *JOBQ, *OUTQ, and incomplete job log jobs).

- If the work control block table was rebuilt:

  Delete all job message queues, spool control blocks, and local data areas that existed when it was discovered that the work control-block table required rebuilding. Create a job message queue, spool control block, and local data area for each of the jobs specified in the system values QTOTJOB and QADLTOTJ.

12. **SRC C9xx 2A80** – creates temporary job structures.

    Processing access groups (PAGs) and QTEMP libraries are created for the number of jobs represented by the sum of the system values QACTJOB and QADLACTJ.

13. **SCR C9xx 2B20** – automatic configuration, if requested.

    If the system value QAUTOCFG is set to '1' (the default). the local configuration elements, as determined during the IPL, are compared to the corresponding control unit and device descriptions. The differences are corrected.

14. **SRC C9xx 2B30** – start QLUS job.

15. **SRC C9xx 2B40** – vary on ONLINE(*YES) configuration.

16. **SRC C9xx 2C20** – DIA recovery.

17. **SRC C9xx 2C30** – SNADS recovery.

Simultaneous Initial Program Load

Several IPL steps take place after the controlling subsystem is started (for example, parallel with normal user processing):

1. Clear QRPLJOB.
2. Additional DIA recovery.
3. Access path rebuilding specified for *AFTIPL (in the SCPF job running at RUNPTY 52) takes place at the same time as normal user processing.
4. Create command analyzer work spaces, two for each active job as represented by the sum of the system values QACTJOB and QALDACTJ.
5. Complete job logs for jobs which were not completed normally during the previous system end.

   Note: During an attended IPL, there is an option to clear these incomplete job logs. The default is *NO.

Licensed Internal Code Completion

Forced Licensed Internal Code completion (delayed off) allows the system to complete AS/400 machine interface instructions (MI instructions) before ending the system abnormally.

Without forced Licensed Internal Code completion, when an MI instruction is not allowed to complete, the system and user objects it refers to may be marked as unusable. At the next IPL, these objects are examined and a recovery process is required to make them usable again. (For example, access paths marked as unusable will

have to be rebuilt.) However, when you use forced Licensed Internal Code completion, the system attempts to complete the MI instruction(s). If completed, the objects referred to by it are marked as usable, and a long recovery process is avoided.

Using Forced Licensed Internal Code Completion

To use this function, turn the Power switch, not the Unit Emergency switch on the control panel to the Delayed Off position. The keylock must be in the Manual position. This requests that the system do a delayed power down of the entire system. The actual power down is delayed while the machine attempts to complete all interrupted MI instructions. After the forced Licensed Internal Code completion processing is finished, the power is turned off.

Note: If you do an IPL of the system, or switch off the emergency power without first turning off the power, you will cause an immediate interruption of machine processing. This does not provide any delay to force Licensed Internal Code completion and can result in significant additional recovery processing due to the immediate interruption of machine processing that occurs.

If every interrupted instruction can be completed, the amount of time needed to make every object usable can be significantly reduced.

How Forced Licensed Internal Code Completion Ends: Forced Licensed Internal Code completion will stop if:

- All interrupted instructions are completed within 16 minutes, and the machine shuts down normally.
- The system is unable to successfully complete all MI instructions within 16 minutes. In this case, a main storage copy is performed and the machine automatically shuts down. The main storage copy is used on the next IPL to avoid directory recovery and minimize database recovery. In this case, some access paths may be marked as unusable. That is, the MI instructions that could not complete successfully may have been performing an operation on an access path when they were stopped. When you restart the system, it must then rebuild any unusable access paths (unless they are journaled), and perform all normal recovery steps. The IPL is labeled abnormal to allow certain OS/400 functions to perform additional recovery operations.

Normally, there are few invalidated access paths that are not usable if interrupted machine interface (MI) instructions are completed. If access paths are unusable, the Edit Rebuild Access Paths display is shown allowing you to select how the system will rebuild these access paths. (If all access paths are usable, this display is not shown.)

Appendix C. Main Storage Dump Space Not Available (CPI0987)

It is assumed that you are at this procedure because you have received message CPI0987: `Main storage dump space is not available.`

The system does not have sufficient free space in the system ASP (ASP 1) for the operating system to reserve main storage dump space.

For main storage dump space, the system reserves 1MB of auxiliary storage for each 1MB of main storage. The main storage dump space is used to isolate problems and reduce recovery time. Failure to provide sufficient main storage dump space may critically affect service and system recovery time.

Not having enough main storage dump space can be caused by the following:

- On all systems, insufficient disk storage space in the system ASP (ASP 1).
- On 9406 Model B70, D70, and D80 system units, insufficient disk storage space on units eligible to contain main storage dump space.

Determining the Cause for Insufficient Main Storage Dump Space

Task Overview

1. Display the disk configuration capacity
2. Determine the problem

To determine the cause of insufficient main storage dump space, do the following:

Start SST by typing the following:

```
STRSST
```

Press the Enter key. The following display is shown.

```
                    System Service Tools (SST)

Select one of the following:

     1. Start a service tool
     2. Work with active service tools
     3. Work with disk units
     4. Work with disk data recovery









Selection
     3

F3=Exit          F10=Command entry          F12=Cancel
```

Step 1. Display the Disk Configuration Capacity

1. Select option 3 (Work with Disk Units) and press the Enter key. The following display is shown.

```
                          Work with Disk Units

Select one of the following:

     1. Display disk configuration
     2. Display checksum configuration
     3. Calculate checksum configuration
     4. Work with ASP threshold
     5. Work with disk unit recovery
     6. Work with disk unit information
     7. Calculate mirrored capacity









Selection
     1

F3=Exit          F12=Cancel
```

2. Select option 1 (Display disk configuration) on the Work with Disk Units display and press the Enter key. The following display is shown.

```
                          Display Disk Configuration

Select one of the following:

     1. Display disk configuration status
     2. Display disk configuration capacity
     3. Display disk configuration protection
     4. Display non-configured units









Selection
     2

F3=Exit          F12=Cancel
```

3. Select option 2 (Display disk configuration capacity) and press the Enter key.

Step 2. Determine the Problem

The following display is shown.

Note: The example of the Display Disk Configuration Capacity display (Figure C-1) is for a system that has mirrored protection for the system ASP (ASP 1). For a system without mirrored protection, use the value in the *Unprotected % Used* column to determine the problem.

```
                    Display Disk Configuration Capacity

                                                    --Protected--  --Unprotected--
ASP  Unit  Type  Model  Threshold  Overflow     Size  %Used     Size  %Used
 1                          90%    No          2354 81.04%        0  0.00%
       1   2800   001                             0  0.00%        0  0.00%
       1   2800   001                             0  0.00%        0  0.00%
       5   9336   020                             0  0.00%        0  0.00%
       5   9336   020                             0  0.00%        0  0.00%
       6   9336   020                             0  0.00%        0  0.00%
       6   9336   020                             0  0.00%        0  0.00%









Press Enter to continue.

F3=Exit   F5=Refresh    F11=Display non-configured units    F12=Cancel
```

Figure C-1. Display Disk Configuration Capacity Display

- *ASP.* The auxiliary storage pool number.
- *Unit.* The number assigned by the system to identify the disk unit.

- *Type*. The disk unit type assigned by the manufacturer.
- *Model*. The disk model that identifies the feature level for a specific type of disk unit.
- *Threshold*. The current threshold. The system notifies you when this percentage of storage is full.
- *Overflow*. Indicates if this ASP is overflowed.
- *Protected Size*. Amount of storage in the ASP that is protected by checksum or mirrored protection.
- *Protected % Used*. Percent of protected size that is used.
- *Unprotected Size*. Amount of storage in the ASP that is unprotected.
- *Unprotected % Used*. Percent of unprotected size that is used.

If the value shown in the *Protected % Used* column for ASP 1 is greater than 97%, then the problem is insufficient disk storage space in the System ASP. Use "Dump Space Recovery Procedure 1" to recover.

If the value shown in the *Protected % Used* column for ASP 1 is less than 97%, then the problem is insufficient disk storage space on units eligible to contain main storage dump space. Use "Dump Space Recovery Procedure 2" on page C-5 to recover.

Recovery Procedures

The following recovery procedures are used when there is not enough main storage dump space available in the system ASP.

Dump Space Recovery Procedure 1

Use one of the following options to recovery.

Option 1. Reduce Storage Use and Perform an IPL

Use the following to reduce storage:

- Delete any unused objects.
- Save the old versions of the system log QHST that are not being used and then delete them.
- Print or delete spooled output files on the system.
- Save objects by specifying STG(*FREE) on the save command.

Option 2. Add Storage Units to the System ASP

- If the system does not have mirrored protection, use the procedure "Adding Units to an Existing ASP" on page 7-14.
- If the system has mirrored protection, use the procedure "Adding Disk Units to a Mirrored ASP" on page 16-1.

Dump Space Recovery Procedure 2

There are two options for this recovery procedure, depending on the type of hardware:

1. Option 1 is the recovery procedure for the 9406 model B70 (stage 1) system unit.
2. Option 2 is the recovery procedure for the 9406 model D70 or D80 system units.

Option 1. Stage 1 Hardware, 9406 Model B70

To provide the additional main storage dump space, you must do the following:

1. Save the entire system using the procedure in the *Basic Backup and Recovery Guide*.
2. Upgrade the current load source device by contacting your service representative.
3. Install the Licensed Internal Code using the procedure in the *Basic Backup and Recovery Guide*.
4. Recreate any user ASPs using the procedure "Creating a User ASP and Adding Disk Units to the New User ASP" on page 7-4.
5. Restore the operating system using the procedure in the *Basic Backup and Recovery Guide*
6. Restore the remaining parts of the system using the procedure in the *Advanced Backup and Recovery Guide*.

If the system ASP had mirrored protection, a mirrored unit must be provided for pairing with the new load source unit. After the ASPs are recreated, start mirrored protection on the ASPs that previously had mirrored protection.

Option 2. Stage 2 Hardware, 9406 Model D70 or D80

To provide for additional main storage dump space, you must add at least 1 2800-001 storage unit to the system ASP (ASP 1). If all non-load source 2800-001 storage units are configured to user ASPs, you must save the user ASP that the unit is being moved from, move the unit, and then restore the user ASP.

- If the system does not have mirrored protection:
  - To move a disk unit from one user ASP to another, use the procedure, "Moving a Disk Unit from an Existing ASP that Has Sufficient Storage" on page 7-28.
  - To add units to an existing ASP, use the procedure "Adding Units to an Existing ASP" on page 7-14.
- If the system has mirrored protection:
  - To move a disk unit from a mirrored ASP to another, use the procedure "Moving a Disk Unit from a Mirrored ASP to Another ASP" on page 16-7.
  - To add units to a mirrored ASP, use the procedure "Adding Disk Units to a Mirrored ASP" on page 16-1.

Appendix D. Example of Configuration Planning for Mirrored Protection

The procedure for starting mirrored protection is simple. However, determining how long each part of the mirroring process takes and how to achieve the maximum level of protection requires some planning. Information specifying elapsed time estimates for different disk units is found in the checksum, mirror, journal performance information chapter of *AS/400 Programming: Performance Capabilities Reference*, ZC41-8166.

Three different configurations are used to illustrate the time estimates found in the *Performance Capabilities Reference* and the integration of new storage units to the physical hardware. They are:

1. 9406 Model B70 system unit using the maximum 9335 disk unit configuration
2. 9406 Model D70 system unit using 12-9336-020 disk units with feature code 1204 installed. This feature provides two additional storage units for each 9336-020 disk unit.
3. 9406 Model D60 system unit using a mixture of 9335, 9332-600, and 9336-020 disk units.

Assumptions: Two different assumptions regarding how and when storage is added are incorporated to demonstrate the effect they have on each configuration in terms of elapse time.

- Disk units are added as part of the start mirrored protection procedure using the Dedicated Service Tools (DST) menu.
- Disk units are currently being used by applications and the database before starting mirrored protection.

Mirrored Protection Rule for the 9406 System Unit: The level of mirrored protection for the models of the 9406 system unit can easily be determined by using this rule:

> I/0 processor-level (or bus-level) protection is achieved if you are mirroring three or more strings (or busses) of disk units and each string (or bus) contains no more than half of the total number of storage units in the ASP that is being mirrored.

The exceptions to the above rule are the 2800-001 storage units (internal) for 9406 Models D and E and Model B (except those B60 or B70 models that were manufactured after June, 1991 or that had a new service processor card installed). The best level of protection these models achieve is controller-level protection.

9406 Model B70 System Unit Using the Maximum 9335 Disk Unit Configuration

The 9406 model B70 system unit can have a maximum of three busses and eight I/0 processors. The maximum disk capacity of a 9406 model B70 system unit is 54.8GB, which consists of 64-9335-B01 disk units (855.8MB each). See the *9406 System Installation and Upgrade Guide, Appendix C*, SY41-0700.

The system consisted of eight 6110 I/O processors before adding disk units to increase the disk capacity to support mirrored protection. Each I/O processor had two 9335-A01 controller units and four 9335-B01 disk units attached. All units were installed in one 9309-002 rack for each string (see Figure D-1).

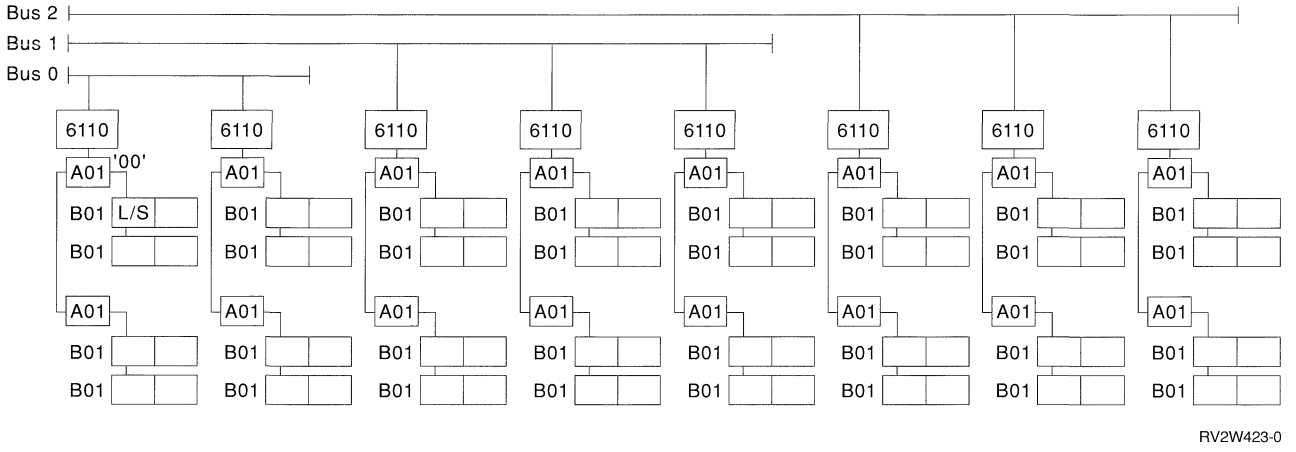


Figure D-1. 9406 Model B70 Original Configuration. The model B70 has eight 6110 I/O processors attached to three busses. Each I/O processor has two 9335-A01 disk controllers and four 9335-B01 disk units.

Additional disk units are connected to the existing strings. This increases the quantity of controller and disk units for each I/O processors to four 9335-A01s and eight 9335-B01 disk units respectively (see Figure D-2).

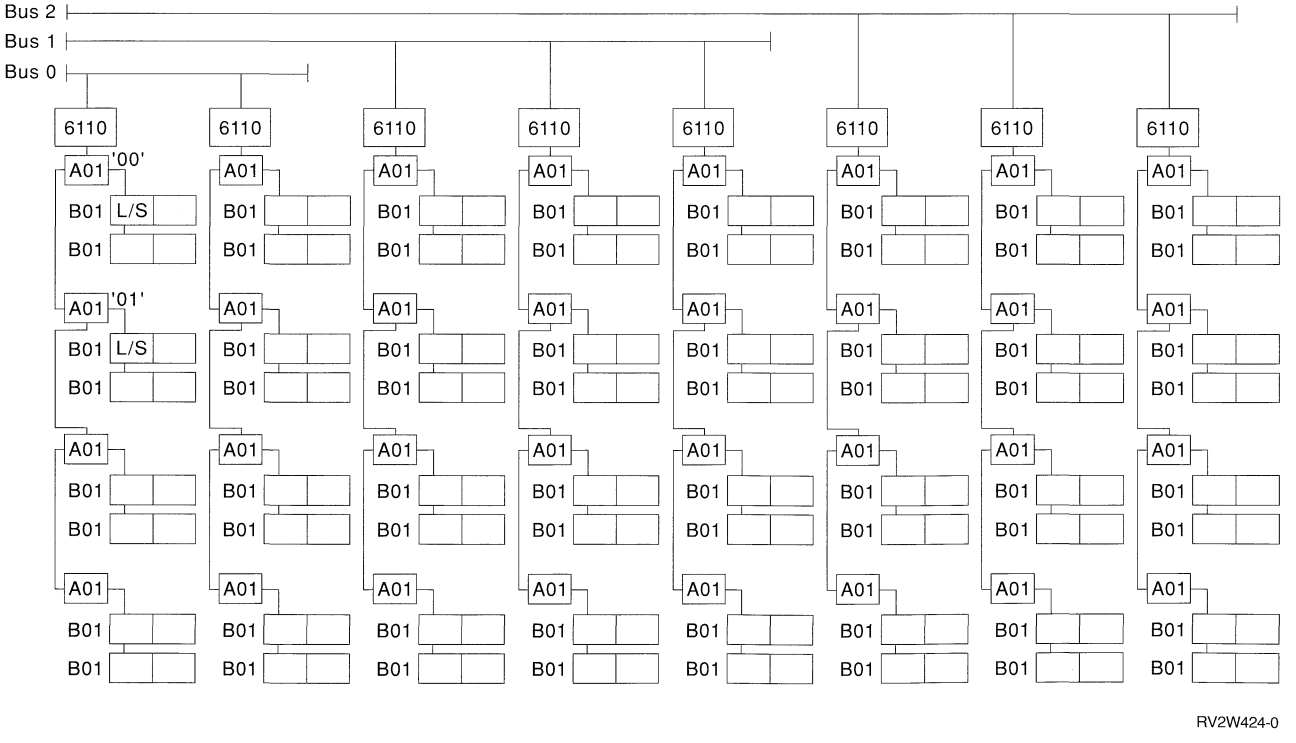


Figure D-2. 9406 Model B70 Mirrored Configuration. The model B70 has eight 6110 I/O processors attached to three busses. Each I/O processor has four 9335-A01 controllers and eight 9335-B01 disk units. If the service processor card on Bus 1 supported the load source unit on the third I/O processor card slot in this configuration, then I/O processor-level of protection would be achieved. However, only controller-level of protection is provided in this configuration. All other storage units achieve bus-level protection.

Because all storage units are added to the system ASP, half of the maximum capacity or 27.4GB is usable for system and user data when mirrored protection is started.

The elapsed time for the start mirrored protection function consists of three parts:

1. Adding disk units to the ASP to be mirrored .

   Note: This part is not actually part of the start mirroring protection function but it is included for planning purposes. Also, this must be included in the estimate because adding disk units is assumed.

2. Moving data from the existing storage units that must be cleared to support the mirrored pairs that are established when start mirrored protection is selected

3. Synchronizing the data from one storage unit to its mirrored unit in the mirrored pair.

Calculating the Time for Adding Disk Units to the System ASP

Storage units in a 9335 disk unit configuration are added sequentially to the ASP by the 9335-A01 disk controller. Up to a maximum of sixteen storage units can be added concurrently on the 9406 system unit. One 9335-B01 storage unit takes 14 minutes to add to an ASP. Add 2 minutes for each level of the configuration because of contention.

Time for one 9335-B01 storage unit		9335-A01		6110 I/O processor		Bus		Total Time
10 minutes	+	2	+	2	+	2	=	16 minutes

Because 16 9335-A01 controllers are added and each 9335-A01 has 4 storage units attached (2 X 9335-B01), the add operation should take approximately:

4 X 16 (minutes) = 64 minutes (1.1 hours)

Four different sets of 16 9335-B01 storage units are being added concurrently to the system ASP.

Calculating the Time to Move Extents

It is very difficult to calculate the time to move extents because there is no published algorithm to use. The system starts at one end of each disk being cleared and moves all the data off. The primary concern is how many storage units have to be cleared to allow one side of all mirrored pairs of storage units to contain the data and the other side is ready to be synchronized with its mirrored unit. Some other concerns are how much data has to moved and the contention existing with the hardware (I/O processors, busses, and disk units).

It takes approximately 15 minutes to clear each storage unit. Many of the move extents operations are processed concurrently so that the cumulative effect is not entirely synchronous for each storage unit.

Storage Units Added Before Starting Mirrored Protection

The unit chosen as the mirrored unit for the load source (storage unit 1) unit may need to be cleared. The configuration rule for a 9406 model B system unit requires that the load source unit be attached to the first or second I/O processor on Bus 0 and the controller attached to the I/O processor is addressed at hex '00' or '01' (see Figure D-3). This rule also applies to the mirrored unit for the load source unit. All other mirrored units are paired with one of the new 9335-B01 storage units added (usually on another bus) before mirrored protection is started. For more information about mirrored protection rules, see the topic "Mirrored Protection Configuration Rules" on page 15-1.

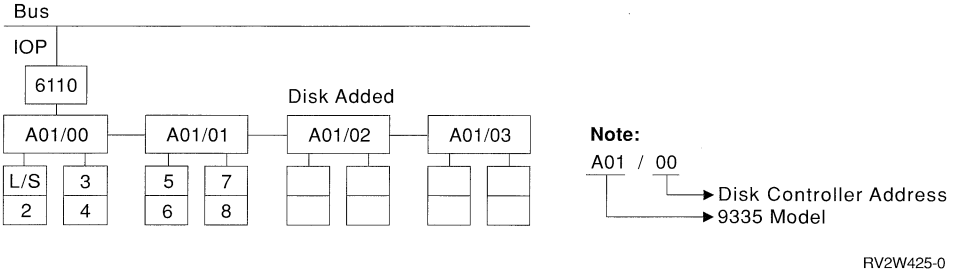


Figure D-3. Adding Disk Units to the Load Source String. This illustration shows the additional 9335-A01 controller and the 9335-B01 disk units being added to the existing string of 9335 disk units. The original load source storage unit is in the first 9335-B01 disk unit following the 9335-A01 controller unit addresses as hex '00'. When mirrored protection is started and the pairs of storage units are determined, it is likely that unit 5 will be cleared to allow it to become the mirrored unit for the load source (L/S) unit. The data that exists on unit 5 will be moved to another storage unit.

Storage Units in Use by the System before Starting Mirrored Protection

This is the extreme situation for moving extents. Because 64 storage units must be cleared to create the mirrored units of a mirrored pair, the move extents process may take several hours.[1]

Determining the Time to Synchronize the Disk

The system performs an IPL to start storage management recovery. This function allows the synchronization of mirroring support to begin. The storage unit containing all of the current data is copied to the storage unit that has been designated as its mirrored unit.

Using the time specified in the *Performance Capabilities Reference* for multiple units across all system unit models, the configuration should take approximately four to five hours. Sixty-four pairs of storage units are synchronizing.

[1] This estimate is based on field reports.

Total Estimate

The following estimates do not include to save the ASP that is being mirrored or any additional IPL time not attributed to starting mirrored protection.

```
Disk units added:

          Add units to ASP ..................  1.1 hours
          Move extents ......................   .4 hours
          Synchronization ...................  5.0 hours

                                        Total:  6.5 hours
```

Disk units in use before starting mirrored protection:

```
          Move extents ......................  5.5 hours
          Synchronization ...................  5.0 hours

                                        Total: 10.5 hours
```

9406 Model D70 Using the 9336 Disk Unit Configuration

In this example, the 9406 model D70 system unit has three busses. See the *9406 System Installation and Upgrade Guide, Appendix C*, SY41-0006.

Bus 0 in the current configuration does not contain any external disk units. Busses 2 and 3 each have three 6112 I/O processors. Each I/O processor has one 9336-020 disk unit attached. Each 9336 disk unit has feature code 1204 installed. This feature code adds two additional 857MB storage units to the 9336 disk unit (see Figure D-4).

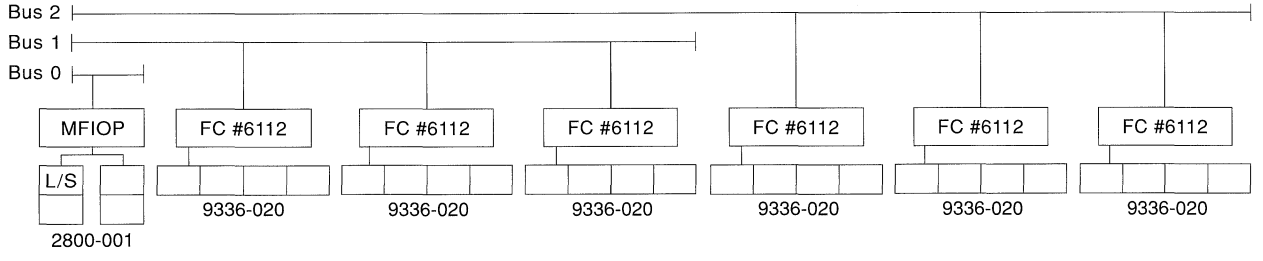


Figure D-4. 9406 Model D70 Original Configuration. The model D70 has six 6112 I/O processors attached to Bus 1 and Bus 2. Bus 1 contains the multiple function (MFIOP) I/O processor and the 2800-001 storage units. Each I/O processor has one 9336 model 020 disk with feature code (FC) 1204 attached.

Additional disk units are added to the existing strings to increase the disk capacity as shown in Figure D-5 on page D-6

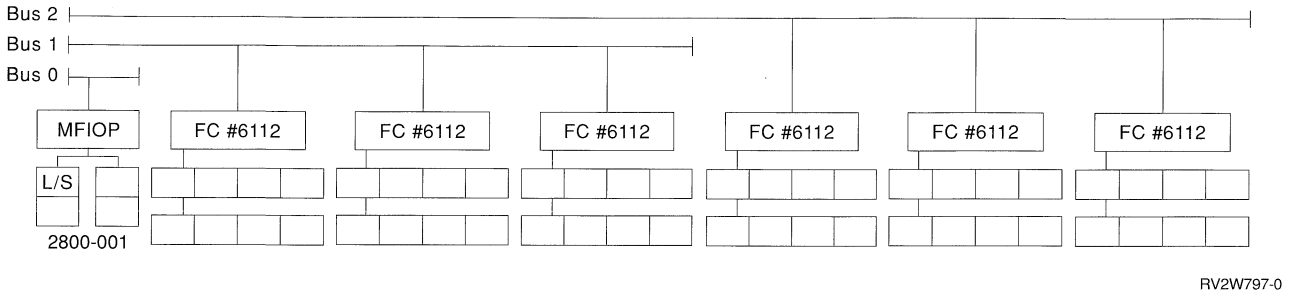


Figure D-5. 9406 Model D70 Mirrored Configuration. The additional 9336-020 disk units are attached to the existing strings. The data on the third and fourth 2800-001 storage units attached to the multiple function I/O processor (MFIOP) are moved to other storage units to allow them to become the mirrored units for unit 1 (load source) and unit 2. The Licensed Internal Code is on the first 2800-001 storage unit.

The load source unit is handled by the 2800-001 internal storage units on the model D70. The 2800-001 storage units create mirrored pairs among themselves when mirrored protection starts. Because all storage units are added to the system ASP, half of the new capacity or 20.6GB is usable for system and user data once it is protected.

Calculating the Time to Add Disk Units to the System ASP

Storage units in a 9336 disk unit are added concurrently to the ASP. The 9406 allows up to a maximum of 16 storage units to be added concurrently. You do not need to select the groups of 16 because the system handles this for you.

It take 14 minutes to add one 9336-020 storage unit to an ASP. For each level of the configuration, add 2 additional minutes.

```
Time for one
9336-020        Disk          6112                        Total
storage unit    controller    I/O processor  Bus          Time

  14 minutes  +    2     +     2     +        2     =    20 minutes
```

Because six disk controllers are working under six I/O processors on two busses, and each 9336-020 added has four storage units attached (a total of 24), this function should take approximately:

```
2 X 20(minutes) = 40 minutes (0.7 hours)
```

Two different sets of 9336-020 storage units are added concurrently to the system ASP.

Calculating the Time to Move Extents

It is very difficult to calculate the time to move extents because there is no published algorithm to use. The system starts at one end of each disk being cleared and moves all the data off. The primary concern is how many storage units have to be cleared to allow one side of all mirrored pairs of storage units to contain the data and the other side is ready to be synchronized with its mirrored unit. Some other concerns are how much data has to moved and the contention existing with the hardware (I/O processors, busses, and disk units).

It takes approximately 15 minutes to clear each storage unit takes. Many of the move extents operations are processed concurrently so that the cumulative effect is not entirely synchronous for each storage unit.

Storage Units Added before Mirrored Protection is Started

The unit chosen as the mirrored unit for the load source unit (storage unit 1) may need to be cleared. If the third and fourth 2800-001 storage units were being used in the system ASP before mirrored protection is started, then they are cleared. This takes approximately 20 minutes. Otherwise, there should be very little time spent in this part of the process.

Storage Units in Use by the System before Mirrored Protection Is Started

In this assumption, 24 storage units are cleared. This can take approximately three and one-half hours. This estimate is based on field reports.

Determining the Time Synchronize

Using the time specified in the *Performance Capabilities Reference* for multiple units across all system unit models, this configuration should take approximately three hours. Twenty-four pairs of storage units are synchronizing.

Total Estimate

The following estimates do not include time to save the ASP that is being mirrored or any additional IPL time not attributed to starting mirrored protection.

```
Disk units added:

              Add units to ASP ..................  0.7 hours
              Move extents ......................   .4 hours
              Synchronization ...................  3.0 hours

                                            Total:   4.1 hours

Disk units in use before starting mirrored protection:

              Move extents ......................  3.5 hours
              Synchronization ...................  3.0 hours

                                            Total:   6.5 hours
```

9406 Model D60 Using the 9332, 9935 and 9336 Disk Unit Configuration

In this example, the 9406 model D60 system unit has three busses. See the *9406 System Installation and Upgrade Guide, Appendix C*, SY41-0006. The current configuration contains a mixture of 9332 and 9335 disk units. As illustrated in Figure D-6 on page D-8, they are connected to Busses 2 and 3. There are four 9332-600 disk units. Two 9332-500 disk units are attached to each 6112 I/O processor.

The 9335 disk unit configuration consists of five 9335-A01 disk controllers and nine 9335-B01 disk units. Two 9335-A01 disk controllers and four 9335-B01 disk units are on each string. Each string is attached to a separate bus (Bus 2 and 3). A third string contains the remaining 9335-A01 disk controller and the 9335-B01 disk unit. The third string is attached to Bus 2. See Figure D-6 on page D-8.

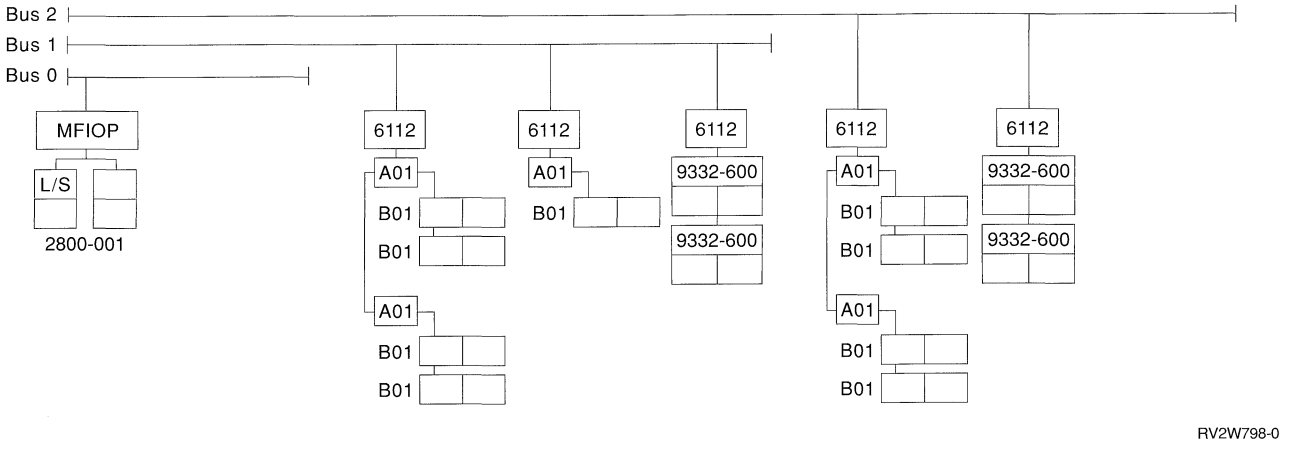


Figure D-6. 9406 Model D60 Original Configuration. The model D60 has five 6112 I/O processors attached to Bus 2 and Bus 3. Bus 0 has the multiple function I/O processor (MFIOP) with four 2800-001 storage units attached. One I/O processor on each bus has two 9335-A01 controllers and four 9335-B01 disk units in a string; one I/O processor on each bus has two 9332-600 disk units in a string; one I/O processor on Bus 2 has one 9335-A01 controller and one 9335-B01 disk unit.

Disk capacity is expanded by adding four 9336-020 disk units with feature code 1203 installed. This feature code adds one additional storage unit to each 9336-20 disk unit. The disk units are divided between two 6112 I/O processors. Each I/O processor is attached to a separate bus (Busses 2 and 3). See Figure D-7.

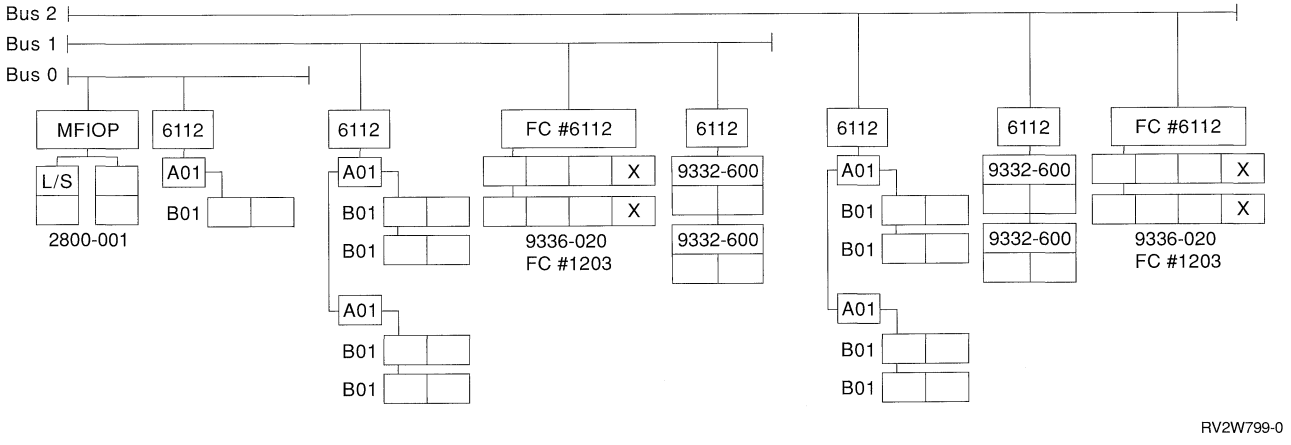


Figure D-7. 9406 Model D60 Mirrored Configuration. The model D60 has seven 6112 I/O processors attached to Bus 1 and Bus 2. Bus 0 has the multiple function I/O processor (MFIOP) and one 6112 I/O processor. On Bus 2 and Bus 3, one I/O processor has a string of two 9335-A01 controllers and four 9335-B01 disk units; one I/O processor has a string of 9332-600 disk units; and one I/O processor has a string of 9336-020 disk units with feature code (FC) 1203 installed.

To prevent a lower level of mirrored protection for the two 9335 disk unit strings on Bus 2, the single 9335-A01 controller and 9335-B01 disk unit are moved to Bus 0. If the single 9335-B01 remains on Bus 2, at least four pairs of 9335-B01 storage units would have only I/O processor-level protection, making Bus 2 a single point of failure (see "Mirrored Protection Rule for the 9406 System Unit" on page D-1).

Note: Moving disk units to new **physical** locations on the same system does not destroy the permanent and free space directories for the objects in the that ASP.

Calculating the Time to Add Disk Units to the System ASP

Storage units in a 9336 disk unit are added concurrently to the ASP. The 9406 allows a maximum of 16 storage units to be added concurrently. It takes 14 minutes to add one 9336-020 storage unit to an ASP. Add two additional minutes for each level of the configuration because of contention.

```
Time for one
9336-020       Disk         6113                      Total
storage unit   Controller   I/O processor  Bus        Time

14 minutes  +    2     +       2       +    2     =   20 minutes
```

Four 9336 disk controllers are attached to two I/O processors. Each I/O processor attached to a separate bus. Each 9336-020 disk unit has 3 storage units (a total of 12). This function should take approximately:

```
1 X 20(minutes) = 20 minutes (0.3 hours)
```

Twelve 9336-020 storage units are being added concurrently to the system ASP.

Calculating the Time to Move Extents

The estimates shown here are based on field reports.

It is very difficult to calculate the time to move extents because there is no published algorithm to use. The system starts at one end of each disk being cleared and moves all the data off. The primary concern is how many storage units have to be cleared to allow one side of all mirrored pairs of storage units to contain the data and the other side is ready to be synchronized with its mirrored unit. Some other concerns are how much data has to moved and the contention existing with the hardware (I/O processors, busses, and disk units).

It takes approximately 15 minutes to clear each storage unit takes. Many of the move extents operations are processed concurrently so that the cumulative effect is not entirely synchronous for each storage unit.

Storage Units Added before Starting Mirror Protection

Half of all the 9332 and 9335 storage units are cleared and the data moved to half of the new 9336 storage units. If the third and fourth 2800-001 storage units are used in the system ASP before mirrored protection is started, then they are cleared. Clearing the 2800-001 storage units take approximately 20 minutes. This entire process should take approximately two or three hours.

Storage Units in Use by the System before Mirrored Protection is Started

In addition to the 9332 and 9335 storage units being cleared, six 9336 storage units are also cleared. The process increases to approximately three to four hours.

Determining the Time to Synchronize the Disk

Using the time specified in the *Performance Capabilities Reference* for multiple units across all system models, the configuration should take approximately two and a half hours. There are 19 pairs of storage units being synchronized.

Total Estimate

The following estimates do not include to save the ASP that is being mirrored or any additional IPL time not attributed to starting mirrored protection.

```
Disk units added:

              Add units to ASP .....................  0.3 hours
              Move extents .........................  2.8 hours
              Synchronization ......................  2.5 hours

                                            Total:    5.6 hours

Disk units in use before starting mirrored protection:

              Move extents .........................  3.8 hours
              Synchronization ......................  2.5 hours

                                            Total:    6.6 hours
```

Assessment of Single-Points-of-Failure

For a multiple ASP system, it takes careful planning and analysis to determine which storage units should be assigned to an ASP. In a mirrored protection environment, you can significantly reduce the effective level of protection if the storage units are chosen randomly. Consider the configuration in Figure D-8.

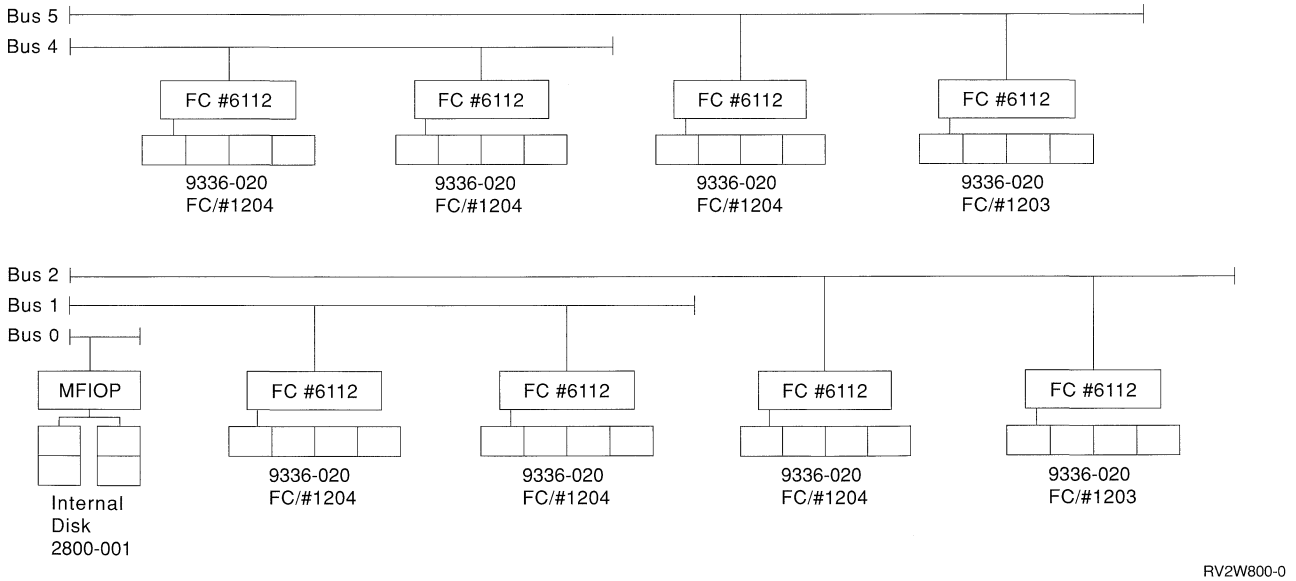


Figure D-8. 9406 Model D80 with 9336 Disk Unit Configuration. The model D80 has eight 6112 I/O processors attached to Bus 1, Bus 2, Bus 4, and Bus 5. Each 6112 I/O processor has two 9336-020 disk units with feature code 1204 installed. The multiple function I/O processor (MFIOP) has the 2800-001 storage units attached to Bus 0.

In Figure D-8, to determine which storage units should be selected for a user ASP, consider the following:

Two scenarios are used to illustrate the need for careful planning when selecting storage units for user ASPs in a mirrored environment.

System 1 The system ASP and the user ASP have mirrored protection.

System 2 Only the system ASP has mirrored protection.

Both systems have eight 9336-020 storage units for the user ASP. When referring to Figure D-8, keep in mind that you must select the highest level of protection with the least number of single-point-of-failures. In System 2, determine which storage units must be selected to minimize disk subsystem failures that can cause the system to stop processing.

System 1

In Figure D-9, one storage unit from each 9336-020 is selected to achieve highest level of protection for both ASPs.

System 2

In Figure D-10 on page D-12, all eight storage units are selected from the two 9336-020 disk units attached to Bus 5 for user ASP 2. This minimizes the number of disk subsystem failures that can cause the system to stop processing.

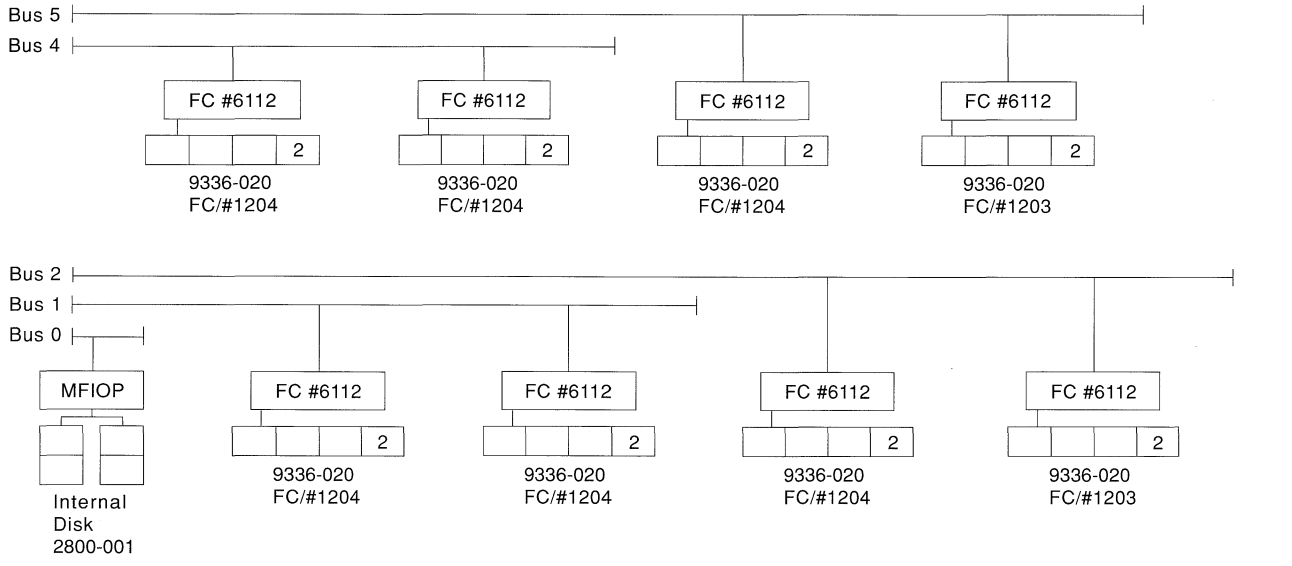


Figure D-9. System 1–9406 Model D80 Configuration. The multiple function I/O processor (MFIOP) and the 2800-001 storage units are attached to Bus 0. User ASP 2 is built by selecting a storage unit from each 9336-020 disk unit. Both the system ASP and the user ASP has mirrored protection.

For the 9336 disk unit, the points-of-failure are:

```
1  -  Bus 5
2  -  6112 I/O processor attached to Bus 5
2  -  Controllers in the 9336 disk units attached to Bus 5
8  -  Storage units
-----------
13  -  points-of-failure
```

All of the 9336 storage units in the system ASP have bus-level protection.

If you selected the storage units for the non-mirrored ASP as shown in Figure D-9, there will be significantly more points-of-failure for the 9336 disk units that could cause the system to stop processing.

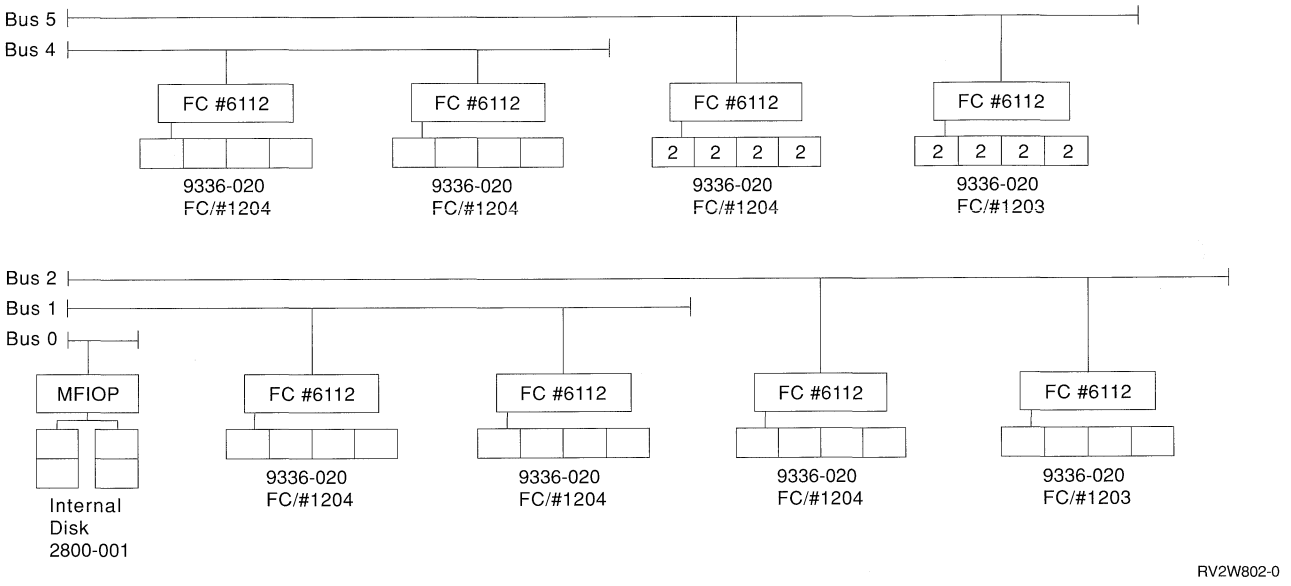


Figure D-10. System 2–9406 Model D80 Configuration. The multiple function I/O processor (MFIOP) with the 2800-001 storage units is attached to Bus 0. User ASP 2 is built from the two 9336 disk units attached to Bus 5. This user ASP is not mirrored. The system ASP has mirrored protection.

The 9336 disk unit points-of-failure are:

```
4  -  Bus 2 through 5
8  -  6112 I/O processors attached to all buses
8  -  Controllers in the 9336 disk unit subsystems
8  -  Storage units
-----------
28  -  Points-of-failure
```

The 9336 storage units in this scenario would really only have device-level protection even though the mirrored image of the storage unit exists on another bus.

When you have non-mirrored disk units on the system, you can see it is important to consider the actual points-of-failure that can cause the system to stop processing. In this case, 13 points-of-failure (System 2) versus 28 point-of-failure (if System 2 uses the configuration of System 1 for ASP 2).

Note: If you look at the Work with Disk Configuration Status display for **either** of these systems, you can see that 24-9336 storage units have bus-level protection. However, when the unprotected storage units in the ASP were chosen as in Figure D-9 on page D-11, the ***practical*** effect is that the disk units have the lowest level (device-level) of protection. Basically, device-level protection provides protection from data-loss.

Converting from Checksum Protection to Mirrored Protection

It can be difficult to calculate the amount of usable disk capacity to determine the capacity of additional disk units to support mirrored protection on a checksum protected system. One reason is that the amount of redundancy storage is not directly observable through any system display.

You can use the Work with Disk Status (WRKDSKSTS) and Work with System Status (WRKSYSSTS) commands to determine the amount of protected storage

and unprotected storage being used (if it is the system ASP). Then, subtract that total from the rated capacity of the storage units being used in the ASP. The result gives you the amount of disk space being used to hold the checksum data for checksum protection. In your calculation for actual space is being used to establish a basis for mirrored protection capacity, take that result and add it to the protected storage total to determine the new capacity when checksum is stopped.

In Figure D-11, the Work with Disk Status display shows rated capacity of each storage unit, the amount of protected storage used, and unprotected storage used.

Work with Disk Status

		Size	%			---Protected---		---Unprotected---	
Unit	Type	(MB)	Used	ASP	Checksum	Size	% Used	Size	% Used
1	9335	427	37.8	1	No	0	.0	427	37.8
2	9335	427	36.5	2		0	.0	427	36.5
3	9335	427	82.5	2		0	.0	427	82.5
4	9335	427	81.7	2		0	.0	427	81.7
5	9335	318	83.6	1	Yes	318	83.6	60	17.7
6	9335	318	83.6	1	Yes	318	83.6	60	18.8
7	9335	316	83.7	1	Yes	316	83.7	60	16.9
8	9335	299	83.6	1	Yes	299	83.6	60	17.9
9	9335	299	83.6	1	Yes	299	83.6	60	17.5
10	9335	316	83.6	1	Yes	316	83.6	60	20.1
11	9335	316	83.6	1	Yes	316	83.6	60	17.1
12	9335	316	83.5	1	Yes	316	83.5	60	19.7
13	9335	316	83.5	1	Yes	316	83.5	60	17.8
14	9335	316	83.5	1	Yes	316	83.5	60	17.4
15	9335	316	83.6	1	Yes	316	83.6	60	17.8
16	9335	316	83.5	1	Yes	316	83.5	60	18.1
17	9335	316	83.6	1	Yes	316	83.6	60	18.8
18	9335	316	83.5	1	Yes	316	83.5	60	17.8

Figure D-11. WRKDSKSTS Display

Fifteen 9335-B01 storage units are in ASP 1 (system ASP) or:

```
427MB X 15 = 6405MB
```

The average for the percentage used (%Used) for protected storage is 83.6. This is based on the 14 storage units that have *Yes* specified in the *Checksum* column. The total amount of protected storage is 4394GB. Therefore, the total capacity used by permanent objects is:

```
4394MB X 83.6% = 3673.4MB
```

Assuming the 9406 model contains 64MB of main storage, 100MB is used for the Licensed Internal Code and the main storage dump space on the load source unit. This usage is added in to the permanent objects because checksum considers this entire storage unit as unprotected storage. 3837.4MB of storage used by permanent objects forms the basis for determining how much storage is added to a checksum protected configuration to support a mirrored protection environment.

If checksum protection is stopped, the percentage used (%Used) for the system ASP is:

```
3837.4MB / 6405 B = 59.9 %
```

If storage units are added to the system ASP from ASPs 2 and 3, more efficient use is made of the current capacity when the system converts to mirrored protection.

Note: For AS/400 users using separate ASPs for save files and journal receivers, it is highly recommended to move these objects back to the System ASP when you are considering full mirrored protection.

Calculate the new system ASP size:

```
6405MB + 1281MB = 7686MB
```

Add the storage used by the save files and journal receivers that exist in ASPs 2 and 3 respectively to the value that was calculated for total protected storage and the Licensed Internal Code.

```
3837.4MB + 857MB = 4694.4 MB
```

The system ASP usage is:

```
4694.4MB / 7686 MB = 61.1%
```

If you want to keep this usage when mirrored protection is started, then approximately 7686MB of disk capacity should be added to the system ASP. If you want to increase or decrease this usage when storage is added, use the *Calculate Mirror Protection* option on the Work with Disk Units display using System Service Tools (SST) or Dedicated Service Tools (DST).

Full Mirrored Protection Versus Partial Mirrored Protection

Full mirrored protection and partial mirrored protection do not provide the same availability results. These two implementations of mirrored protection are quite different. The scenarios of a disk unit on the AS/400 system for each of these mirroring methods requires different user responses.

It does not matter if you are using just the system ASP (ASP 1) or multiple user ASPs (2 through 16), full mirrored protection protects all disk units in the AS/400 system. Partial mirrored protection protects only a portion of the disk units designated by one or more ASPs. However, not all storage units in the disk configuration are protected. Therefore, the planning of the disk unit placement and what ASPs are selected for mirrored protection becomes more difficult (see "Assessment of Single-Points-of-Failure" on page D-10).

Besides the planning of ASPs, the significant difference between the two mirrored protection methods regards availability. With full mirrored protection, you maximize the availability of the AS/400 system when a disk subsystem failure occurs. With this method of mirrored protection, it does not matter which ASP has the failure. With partial mirrored protection, the system continues to run while reporting the failed storage unit to the system operator (QSYSOPR) message queue. However, if the disk failure occurs in an ASP that does not have mirrored protection, SRC A6xx 0266 is sent when that ASP is accessed by any job on the system. Because the storage units in the ASP do not have mirrored units, the storage management directory becomes unusable and all input and output operations to the ASP are suspended.

The disk attention SRC does not mean that the system has ended. All input and output operations are queued to allow the service representative to investigate the cause of the disk failure. If the problem is not with the disk media, the failing cards are replaced, the failed disk unit is powered on, and the system continues from the point that the equipment error occurred. All queued input and output operations resume. However, if a disk media failure occurs, the service representative per-

forms a main storage dump to minimizes the next IPL to OS/400, and allows the system to end processing.

With full mirrored protection, the operation of the system is not interrupted while diagnostics and most repairs to resolve the disk subsystem failure problem are taking place. With I/O processor-level protection, the maximum concurrent maintenance is possible, depending on the error. In any case, the user has complete control over the shutdown of the system should a power-down be required to service the disk problem; the system does not end abnormally.

Although critical data is protected with partial mirrored protection, and a restore operation is not required for the data in the protected ASP, you do not have the maximum availability that is provided by full mirrored protection because of the exposure of the unprotected ASP. If your availability requirements state that your system must be in operation within minutes following a failure or remain active during your business hours, partial mirrored protection is not an option in most cases.

Considerations for Internal Disk Storage

Some models of system have internal storage (2800-001 storage units). Many users asked if mirrored protection can be used on the 2800-001 (internal) storage units on the 9406 model D and E system units and checksum protection used on the auxiliary storage (external) disk units. Technically, it can be done if a user ASP is created from the external disk units and the internal disk units remain in the system ASP. If your system is already installed, objects must be moved from the system ASP using the save and restore procedures because the ASP management routines will likely exhaust all free space in the system ASP. However, other considerations quickly discourage this scenario.

As previously mentioned, the use of both data-loss protection methods requires separate ASPs. Both methods are implemented on an ASP-by-ASP basis. You cannot have mirrored protection and checksum protection active in the same ASP. It seems perfectly logical to align these protection methods with the physical boundaries of the internal and external storage.

The ASP is the means by which storage management defines the boundaries of auxiliary storage to the operating system. Therefore, if a data-loss protection method is selected, only one can be used on an ASP at any time. This is why there is a requirement for separate ASPs if different protection methods are used.

In this scenario, space is a problem as well. There is 1.2GB of auxiliary storage available in the four 2800-001 storage units. When mirrored protection is used, the amount of usable space is reduced by half, leaving 620MB of auxiliary storage to hold the Licensed Internal Code (100MB), main storage dump space (8MB to 512MB), and the IBM licensed programs (200MB to 500MB). If you add the space required for temporary objects, spooled files, documents and folders, and other system requirements, space in the system ASP is quickly reduced to almost nothing.

Performance is another consideration if you are still not convinced. The four internal storage units can handle the minimum space required for the system. However, when you consider that all input and output operations for temporary objects must go to the system ASP, the four storage units become quickly utilized.

If you use the performance tools to configure the workload for model D and E system unit, you will notice additional storage is added to the configuration to maintain satisfactory usage rates.

The idea of making the system ASP exist only on the four internal storage units is not feasible. Enough space for system ASP objects and system input and output requirements would be severely constrained and affect user performance. You must use more than one ASP, which introduces disk management overhead to your operation. Utilization of internal storage would exceed guidelines very quickly with an appreciable amount of workload. If you implement two or more user ASPs, you must add some external disk storage along with the internal disk storage to create the system ASP.

Bibliography

This section lists publications that provide additional information about topics described or referred to in this manual. The manuals in this section are listed with their full title and order number, but when referred to in text, a shortened version of the title is used.

Programming Information

- *Communications and Systems Management Guide (Alerts and Distributed Systems Node Executive)*, SC41-9661
  Short Title: *Alerts and DSNX Guide*

  This manual provides the system operator, programmer, or system administrator with information for configuring the AS/400 system to use the remote management support.

- *Communications: Distribution Services Network Guide*, SC41-9588
  Short Title: *Distribution Services Network Guide*

  This manual provides the system operator or system administrator with information about configuring a network for Systems Network Architecture Distribution Services (SNADS) and the Remote Spooling Communications Subsystem/Professional Office System (RSCS/PROFS) bridge. In addition, object distribution functions and document library and distribution services are discussed.

- *Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590
  Short Title: *ICF Programmer's Guide*

  This manual provides provides the AS/400 programmer with information to write application programs that use the AS/400 communications and OS/400 intersystem communications function.

- *Device Configuration Guide*, SC41-8106
  Short Title: *Device Configuration Guide*

  This manual provides the system operator or system administrator with information on how to do an initial configuration and how to change the configuration. This manual also contains conceptual information about device configuration.

- *Licensed Programs and New Release Installation Guide*, SC41-9878
  Short Title: *Licensed Programs and New Release Installation Guide*

  This manual provides the system operator or system administrator with step-by-step procedures for initially installing, installing licensed programs, program temporary fixes (PTFs), and secondary languages from IBM.

  This guide is also for users who already have an AS/400 system with an installed release and want to upgrade to a new release.

- *Migrating from System/38 Planning Guide*, GC41-9624
  Short Title: *Migrating from System/38 Planning Guide*

  This manual provides the application programmer, system administrator, or data processing manager with information to help them migrate their products and applications using the System/38 to AS/400 Migration Aid. It includes information for planning the details of migration and an overview of the functions on the System/38 Migration Aid.

- *Office Services Concepts and Programmer's Guide*, SC41-9758
  Short Title: *Office Services Concepts and Programmer's Guide*

  This manual provides information about writing applications that use OfficeVision/400 functions. This manual also includes an overview of directory services, document distribution services, document library services, document and folder save and restore and storage management, security services, word processing services, and information on finding new ways to integrate your applications with OfficeVision/400.

- *Physical Planning Guide and Reference*, GA41-9571
  Short Title: *Using OfficeVision/400* Word Processing*

  This manual provides provides the data processing manager, system administrator, and installation planning representative with information for planning to set up the AS/400 system. This guide also includes information on cable considerations, physical specifications, electronic customer support (ECS), and unpacking considerations.

Operations

- *New User's Guide*, SC41-8211
  Short Title: *New User's Guide*

  This manual provides display station operators with information about how to sign on and off; send and receive messages, respond to keyboard error messages, use function keys; and use display, command, and help information to control and manage their own jobs.

- *System Operator's Guide*, SC41-8082
  Short Title: *Operator's Guide*

This manual provides the system operator or system administrator with information about how to use the system unit control panel and console, send and receive messages, respond to error messages, start and stop the system, use control devices, work with program temporary fixes (PTFs), and process and manage jobs on the system.

- *Programming: Control Language Programmer's Guide*, SC41-8077
  Short Title: *CL Programmer's Guide*

  This manual provides the application programmer or programmer with a wide range discussion of the AS/400 programming topics.

- *Programming: Control Language Reference*, SC41-0030
  Short Title: *CL Reference*

  This manual provides the application programmer with a description of the AS/400 control language (CL) and its commands. Each command includes a syntax diagram, parameters, default values, keywords, and an example.

- *Cryptographic Support/400 User's Guide*, SC41-8080
  Short Title: *Cryptographic Support/400 User's Guide*

  This manual provides the system operator or programmer with a description of the data security capabilities of the AS/400 Cryptographic Support. Cryptographic support is not a part of the operating system. You can order the cryptographic licensed program from the IBM Software Division.

- *Distributed Data Management Guide*, SC41-9600
  Short Title: *DDM Guide*

  This manual provides the application programmer or programmer with information about remote file processing. It describes how to define a remote file to OS/400 DDM (distributed data management), how to create a DDM file, what file utilities are supported through DDM, and the requirements of OS/400 DDM as related to other systems.

- *Systems Application Architecture* Structured Query Language/400 Reference*, SC41-9608
  Short Title: *SQL/400* Reference*

  This manual provides the application programmer, programmer, or database administrator with information that describes SQL/400 statements and their parameters.

- *Data Management Guide*, SC41-9658
  Short Title: *Data Management Guide*

  This manual provides the application programmer with information about using files in application programs. Files allow data that is external to an application program to be read from or written to devices attached to the system, such as database files, device files, and files used to communicate with the system.

- *Programming: Performance Tools/400 Guide*, SC41-8084
  Short Title: *Performance Tools/400 Guide*

  This manual provides the programmer with information about what AS/400 Performance Tools are, gives an overview of the tools, and tells how the tools can be used to help manage system performance.

- *Security Reference*, SC41-8083
  Short Title: *Security Reference*

  This manual provides the programmer (or someone who is assigned the responsibilities of a security officer) with information about system security concepts, planning for security, and setting up security on the system. This guide does not describe security for specific licensed programs, languages, and utilities.

- *System Operator's Quick Reference*, SX41-9573
  Short Title: *Operator's Quick Reference*

  This manual provides the system operator with quick reference information when working with the AS/400 system. This guide contains summaries of information such as system values and OS/400 DDS keywords.

- *Programming: Work Management Guide*, SC41-8078
  Short Title: *Work Management Guide*

  This manual provides the programmer with information about how to create a work management environment and how to change it.

- *System Operator's Guide*, SC41-8082
  Short Title: *Operator's Guide*

  This manual provides the system operator or system administrator with information about how to use the system unit operator display, send and receive messages, respond to error messages, start and stop the system, use control devices, work with program temporary fixes (PTFs) and process, and manage jobs on the system.

- *System/36 to AS/400 Migration Aid User's Guide and Reference*, SC09-1166
  Short Title: *System/36 to AS/400 Migration Aid User's Guide and Reference*

  This manual provides the system operator, applications programmer, systems programmer and data processing manager with information about using the S/36* to AS/400* migration aid to move S/36 items to the AS/400 System using menus and displays or commands.

- *System/38 to AS/400 Migration Aid User's Guide and Reference*, SC09-1165

Short Title: *System/38 to AS/400 Migration Aid User's Guide and Reference*

This manual provides the system operator, application programmer, programmer, or data processing manager with information about using the System/38 to AS/400 Migration Aid to move System/38 objects to the AS/400 system using menus and displays, or commands.

- *9406 System Installation and Upgrade Guide*, SY44-0700

Short Title: *9406 System Installation and Upgrade Guide*

This manual provides the service representative with information about upgrading equipment on the 9406 System Unit. It provides information on an entire range of upgrades such as simple memory card additions, device and rack additions, and model upgrades. It is used with the instruction packets that are shipped with the upgrade equipment.

Index

Numerics

A

B

C

D

E

F

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

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

Application System/400
Advanced Backup and Recovery Guide
Version 2

Publication No. SC41-8079-02

Overall, how would you rate this manual?

	Very Satisfied	Satisfied	Dissatis-fied	Very Dissatis-fied
Overall satisfaction				

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

Accurate				
Complete				
Easy to find				
Easy to understand				
Well organized				
Applicable to your tasks				
THANK YOU!				

Please tell us how we can improve this manual:

May we contact you to discuss your responses? __ Yes __ No

Phone: (____) _____ Fax: (____) _____

To return this form:

- Mail it
- Fax it
  United States and Canada: **800+937-3430**
  Other countries: **(+1)+507+253-5192**
- Hand it to your IBM representative.

Note that IBM may use or distribute the responses to this form without obligation.

Name _____ Address _____

Company or Organization _____ _____

Phone No. _____ _____

Readers' Comments—We'd Like to Hear from You!
SC41-8079-02

IBM®

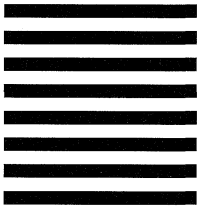
Cut o
Along

Fold and Tape

Please do not staple

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-9986

Fold and Tape

Please do not staple

Fold and Tape

SC41-8079-02

Cut or
Along

Program Number: 5738-SS1

Printed in Denmark by
Scanprint a-s, Viby J.

SC41-8079-02