IBM

MVS/Extended Architecture Checkpoint/Restart User's Guide

Licensed Program

Order Number
GC26-4139-1

Data Facility Product
5665-XA2

Version 2
Release 3.0

IBM

MVS/Extended Architecture Checkpoint/Restart User's Guide

Licensed Program

Order Number
ES1-0857-0

Data Facility Product
5665-XA2

Version 2
Release 3.0

| **Second Edition (June 1987)**

| This is a major revision of, and makes obsolete, GC26-4139-0.

| This edition applies to Version 2 Release 3.0 of MVS/Extended Architecture Data Facility Product, Licensed Program 5665-XA2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Preface

This publication describes checkpoint/restart for MVS/Extended Architecture (MVS/XA). The checkpoint/restart program allows programmers and system analysts to:

- Record information about a job at designated checkpoints.
- Restart a job at the beginning of a step or at a checkpoint within a step.

Checkpoint/Restart runs in 31-bit addressing mode when necessary and can call other components that run in 24-bit addressing mode.

Organization

This publication is organized as follows:

- Chapter 1, "Introduction" on page 1, describes in general terms checkpoint/restart, its components, its dependencies, and information on data set security.
- Chapter 2, "Coding a Checkpoint" on page 7, describes how to set up checkpoints for job steps.
- Chapter 3, "Requesting Restart" on page 31, describes how to request various types of restart.
- Chapter 4, "Checkpoint/Restart Processing" on page 51, describes how to use checkpoint/restart with user data sets, tapes, and other programs.
- Appendix A, "Checkpoint/Restart Codes" on page 71, lists completion codes, return codes, and reason codes issued by checkpoint/restart.
- Appendix B, "End-of-Volume Checkpoint Routine" on page 79, describes how to establish a checkpoint at end-of-volume.
- "List of Abbreviations" on page 81 defines the acronyms used in this book.
- Index is a subject index to this publication.

Prerequisite Knowledge

In order to use this book efficiently, you should be familiar with the following topics:

- Job control language
- Data management

Required Publications

- *MVS/Extended Architecture JCL,* GC28-1148
- *MVS/Extended Architecture Data Administration Guide,* GC26-4140
- *MVS/Extended Architecture Data Administration: Macro Instruction Reference,* GC26-4141

Related Publications

Within the text, references are made to the publications listed in the table below:

Short Title	**Publication Title**	**Order Number**
Checkpoint/Restart SVC Logic	*MVS/Extended Architecture Checkpoint/Restart Supervisor Call Logic*	LY26-3957
Data Administration Guide	*MVS/Extended Architecture Data Administration Guide*	GC26-4140
Data Administration: Macro Instruction Reference	*MVS/Extended Architecture Data Administration: Macro Instruction Reference*	GC26-4141
Data Facility Product: Customization	*MVS/Extended Architecture Data Facility Product Version 2: Customization*	GC26-4267
IBM 3800 Printing Subsystem Programmer's Guide	*IBM 3800 Printing Subsystem Programmer's Guide*	GC26-3846
JCL User's Guide	*MVS/Extended Architecture JCL User's Guide*	GC28-1351
JCL Reference	*MVS/Extended Architecture JCL Reference*	GC28-1352

Short Title	Publication Title	Order Number
JES2 Initialization and Tuning	*MVS/Extended Architecture System Programming Library: JES2 Initialization and Tuning*	SC23-0065
JES3 Initialization and Tuning	*MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning*	SC23-0059
JES3 Introduction	*MVS/Extended Architecture JES3 Introduction*	GC23-0049
JES3 Messages	*MVS/Extended Architecture Message Library: JES3 Messages*	GC23-0062
Magnetic Tape Labels and File Structure Administration	*MVS/Extended Architecture Magnetic Tape Labels and File Structure Administration*	GC26-4145
Supervisor Services and Macro Instructions	*MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions*	GC28-1154
System Generation	*MVS/Extended Architecture Installation: System Generation*	GC26-4148
System Messages	*MVS/Extended Architecture Message Library: System Messages*, Volumes 1 and 2	GC28-1376 and GC28-1377
VSAM Administration: Macro Instruction Reference	*MVS/Extended Architecture VSAM Administration: Macro Instruction Reference*	GC26-4152

Summary of Changes

Release 3.0, June 1987

Restructure

Most of the text from "Appendix B. End-of-Volume Checkpoint Routine" has been removed and placed in *Data Facility Product: Customization.*

Release 1.0, April 1985

Enhancements

- Support for the IBM 3480 Magnetic Tape Subsystem has been added.
- Information to support the IBM 4245, 4248, and 3262 Model 5 Printers has been added under "Use of CHKPT with Other Macro Instructions" on page 24.

Version 2 Publications

The Preface includes new order numbers for Version 2.

Contents

Figures

Chapter 1. Introduction

Checkpoint/Restart is a method of recording information about a job at programmer-designated checkpoints so the job can be restarted at one of these checkpoints or at the beginning of a job step.

A checkpoint is taken when your program issues the CHKPT macro instruction. This macro causes the contents of the program's virtual storage area and certain system control information to be written as a series of records in a data set. These records can be retrieved from the data set if the job terminates abnormally or produces erroneous output, and the job can be restarted. Restart can take place immediately (initiated by the operator at the console) or be deferred until the job is resubmitted. In either case, you can avoid the time-consuming process of rerunning the entire job from the beginning.

Types of Restart

The system allows two types of restart:

- Automatic step restart
- Deferred step restart

Checkpoint/Restart allows two types of restart:

- Automatic checkpoint restart
- Deferred checkpoint restart

Automatic restarts occur in different ways:

- Automatic step restart is restarted at the beginning of a job step that has abended.
- Automatic checkpoint restart is restarted at the last checkpoint taken before the job failed.

Deferred restart occurs when a job is resubmitted with the RESTART parameter specified on the JOB statement.

- Deferred step restart occurs at the beginning of the job step specified in the RESTART parameter of the JOB statement.

- Deferred checkpoint restart occurs at the checkpoint specified in the RESTART parameter of the JOB statement.

Components of Checkpoint/Restart

CHKPT Macro Instruction

The CHKPT macro is coded in your program to cause a checkpoint to be taken.

When a CHKPT macro is executed, the contents of the program's virtual storage data area and certain system control information are written in a data set as a series of records. The series of records is called a checkpoint entry, and the data set in which they are written is called a checkpoint data set. The checkpoint entry, which has a unique programmer-specified or system-generated identification called a *checkid*, is retrieved from the data set when restart occurs.

For a detailed explanation of how to establish a checkpoint, see Chapter 2, "Coding a Checkpoint" on page 7.

End-of-Volume Exit Routine

The end-of-volume exit routine is coded in your program to allow execution of the CHKPT macro instruction each time the processing of a multivolume physical sequential user data set is continued on another volume.
Appendix B, "End-of-Volume Checkpoint Routine" on page 79 contains a brief description of this. For more detailed information about coding the EOV exit for physical sequential data sets, see *DFP Customization.*

Checkpoint at End-of-Volume Facility

A system-supplied routine takes checkpoints at end-of-volume occurrences for multivolume QSAM and BSAM data sets and can be invoked with a JCL parameter. "Checkpoint at End-of-Volume" on page 8 contains a brief description of this. For more detailed information, see *Data Facility Product: Customization.*

RD (Restart Definition) Parameter

When the RD parameter is used, it is coded in the JOB or EXEC statements and is used to request automatic step restart if job failure occurs and/or to suppress, partially or totally, the action of the CHKPT macro instruction. For more detailed information about this parameter, see Chapter 3, "Requesting Restart" on page 31.

RESTART Parameter

The RESTART parameter, coded in the JOB statement, is required when a job is resubmitted for deferred restart. It specifies the step (for deferred step restart) or the step and the checkpoint within that step (for deferred checkpoint restart) where restart should begin. For more detailed information about this parameter, see Chapter 3, "Requesting Restart" on page 31.

SYSCHK DD Statement

The SYSCHK DD statement is required when a job is resubmitted for deferred checkpoint restart. It defines the checkpoint data set for the job being restarted. For more detailed information about this statement, see "SYSCHK DD Statement" on page 49.

SYSCKEOV DD Statement

The SYSCKEOV DD statement defines the checkpoint data set that is to contain the checkpoint records generated when a checkpoint is taken at end-of volume. For more information about this statement, see "SYSCKEOV DD Statement" on page 10.

Checkpoint/Restart Dependencies

CKPTREST System Generation Specification

The CKPTREST macro instruction, coded at system generation, specifies which system completion codes indicate that a step is eligible for restart. During system generation, a standard IBM-defined set of system completion codes (codes produced when the system executes abend) is placed in a table of eligible codes. The table becomes part of the control program. CKPTREST, which is optional, can be used to delete system completion codes from the table and to add user completion codes (codes produced when your program executes abend) to the table.

For more information on the CKPTREST macro, see *System Generation Reference*.

Job Journal Requirements

The job journal is a sequential data set that resides on the spool volume of the job entry subsystem. It contains a set of selected job-related control blocks that are critical to automatic restart processing.

The job journal is necessary because the scheduler control blocks are maintained in the scheduler work area (SWA) in pageable storage, rather than in a job queue on external storage. When a job or the system itself fails, the space that contains the SWA and its job control blocks is lost. Because it preserves up-to-date copies of certain critical control blocks, the job journal makes it possible to reconstruct the SWA. SWA control blocks are restored to the state they were in just prior to the failing step for automatic step restart. For automatic checkpoint restart, SWA control blocks are reconstructed as they appeared in the most recently issued CHKPT.

For JES2 or JES3, job journaling is required before a job is eligible for an automatic restart.

For JES2: Job journaling is provided to a job in JES2 in one of the following ways:

- JOURNAL is specified on the JOB CLASS CHARACTERISTICS statement in the SYS1.PARMLIB member used for initializing JES2 (see *JES2 Initialization and Tuning*).
- The JCL for the job specifies RD parameter on the JOB statement or the EXEC statement (see the publication *JCL*).

For JES3: Job journaling is provided to a job in JES3 in one of three ways:

- JOURNAL=YES is specified on the CLASS initialization statement. For a description of the CLASS statement, see *JES3 Initialization and Tuning.*
- JOURNAL=YES is specified on the MAIN JCL statement that overrides the CLASS initialization statement. For a description of the MAIN statement, see the publication *JCL.*
- The JCL has specified either the RESTART or the RD parameter on the JOB statement or the RD parameter on the EXEC statement.

The system creates a job journal to hold restart information for any job specifying job journaling. After a system failure and the system restart of the failing main processor, the jobs in execution that requested job journaling are restarted (warm started) by the system. If a job is eligible for automatic restart, the operator is sent the message IEF225D asking if the job should be restarted. If the job is not eligible for restart or if the operator indicates that the restart should not be attempted, any scratch or VIO data sets the job allocated are deleted, and the job terminates. Jobs that frequently use scratch and/or VIO data sets should request job journaling.

User Data Set Security

In order to restart a job step that processes RACF-protected data sets, you must be authorized to access all RACF-protected data sets that were open when the checkpoint was taken. In order to restart, all password-protected data sets that were open at checkpoint time must have the pointer to their password in the ACB when the checkpoint is taken, or the password must be known to the operator. For RACF-protected data sets, the userid specified on the RESTART job statement must be authorized to access the data set.

Checkpoint Data Set Security

An unauthorized user (one who is not authorized by the authorized program facility (APF) is not in supervisor state, does not have the system key (keys 0 through 7)), and cannot communicate directly with the checkpoint data set. The unauthorized user can take checkpoints and do restarts, but, after the operation has begun, a security interface is used by the system to prevent unauthorized alteration of the checkpoint data set.

If an unauthorized user tries to access a checkpoint data set directly, an error message appears, and the job terminates. In addition, the unauthorized user cannot take a checkpoint on a new checkpoint data set if another data control block (DCB) is already open to that data set.

Offline security of the checkpoint data set is ensured by the operator.

Chapter 2. Coding a Checkpoint

This chapter defines a checkpoint and explains how you may establish checkpoints to restart job steps.

The CHKPT macro instruction is coded in your program. When the CHKPT macro executes, job step information about the user's program, virtual storage data areas, data set position, and supervisor control is written as a checkpoint entry in a checkpoint data set. The point at which this information is saved becomes a checkpoint from which a restart may be performed. After the checkpoint entry is written, control returns to your program at the instruction following the CHKPT macro.

The topics discussed in this chapter are:

- When to code a checkpoint
- Coding the DCB for a checkpoint data set
- Coding the DD Statement for a checkpoint data set
- Using checkpoint data sets
- Coding the CHKPT macro instruction
- Canceling a checkpoint
- Requesting serially reusable resources

When to Code a Checkpoint

Using the CHKPT macro instruction, you can code a checkpoint within your program or at end-of-volume. The checkpoint routine records information about certain system information, user storage (including user programs in user storage), and the data sets used by the step executing the CHKPT macro. Recorded information includes:

- For all data sets, the information that can be coded on a DD statement; for example, device type and volume serial numbers. (The contents of the step's JFCBs and JFCB extensions are recorded, as well as the contents of the GDGNTs (generation data group name tables).)

- For data sets which are open at the checkpoint, which process on either magnetic tape or direct access devices, and which use the BSAM, QSAM, QISAM, BPAM, VSAM, or EXCP access methods, the information needed to reposition the data set if restart occurs at a checkpoint.

If a checkpoint is taken and an output data set is extended onto a second direct access volume (because end-of-volume occurred on the first volume and there was no more space available on the volume, or the data set contained 16 extents), restart subsequently occurs at that checkpoint and the system does not delete the extension of the data set.

Note that, when a step using the universal character set (UCS) feature is restarted, the system does not determine whether the UCS buffer or FCB buffer is properly loaded, nor does it alert the operator to the UCS or FCB requirements of the step.

Checkpoint at End-of-Volume

The checkpoint at end-of-volume provides an external checkpoint function with no user program modifications. It executes immediately prior to the invocation of any checkpoint in a user EOV exit routine. Redundancy occurs if a user exit routine is supplied and checkpoint at end-of-volume is invoked for the same data set's end-of-volume occurrence.

Checkpoint at end-of-volume is only executable if the CHKPT=EOV parameter is specified for multivolume data sets or for the second, third, etc., data sets of a concatenation. If the first data set of a concatenation is a multivolume data set, this parameter is also valid on that DD statement.

Checkpoint at end-of-volume issues a CHKPT macro; if, at EOV, an unsuccessful return code is presented, it retries the checkpoint execution one more time in the following situation:

> Return code 08—indicates end-of-extent or end-of-volume occurred for the SYSCKEOV DD before completion of the checkpoint entry. Checkpoint will be retried to use secondary space, if provided.

For the other unsuccessful return codes (0C, 10, 14, or 18) and for unsuccessful retry of return code 08, the following message is generated:

```
IEC067I CHKPT=EOV FACILITY EXECUTED UNSUCCESSFULLY
```

This message is preceded by a checkpoint/restart error message (prefixed "IHJ") that describes the nature of the problem. For more detail on these error messages, see *System Messages*.

In any event, processing continues, and this message is generated for each unsuccessful invocation of the checkpoint at end-of-volume until step termination occurs. Operator intervention is required to halt further processing prior to step end.

CHKPT at EOV JCL Parameter

The CHKPT=EOV parameter on a DD statement requests a checkpoint be taken for this job step at EOV for the data set whose DD has this parameter. The following restrictions apply to the use of this JCL parameter:

1. The DD must define a QSAM or BSAM data set.
2. The QSAM or BSAM data set must be a multivolume data set or the second, third, etc., set of a concatenated set of data sets.
3. The DD statement must not define a SYSOUT, DD *, or DD DATA type data set.
4. The JCL parameters DDNAME and DYNAM cannot be specified on the same DD statement with this parameter.
5. The DD must not define a checkpoint data set.

The following actions result if any of these restrictions are not observed:

1,2,5	No action, no checkpoints are taken, processing continues as if the CHKPT=EOV parameter were not specified.
3,4	JCL error messages appear, and processing is not initiated.

Examples:

```
//DD1   DD DSN=DSN1,DISP=OLD,VOL=SER=(TAPE01,TAPE02),
//            UNIT=TAPE,CHKPT=EOV

//DD2   DD DSN=DSN2,DISP=OLD
//      DD DSN=DSNX,DISP=OLD,CHKPT=EOV
//      DD DSN=DSNY,DISP=OLD,CHKPT=EOV

//DD3   DD DSN=DSN3,DISP=NEW,VOL=(,,,5),
//            UNIT=DISK,SPACE=(CYL,(300,300)),CHKPT=EOV
```

Notes:

1. *DD1 – Multivolume data set*
2. *DD2 – Concatenated data set*
3. *DD3 – Multivolume data set on disk*

SYSCKEOV DD Statement

The SYSCKEOV DD statement defines the checkpoint data set to contain the checkpoint records generated from the checkpoint at end-of-volume. The same restrictions that apply to other checkpoint DD statements (see "Coding the DD Statement for a Checkpoint Data Set" on page 11) also apply to this DD statement, with the following exceptions:

- DISP=MOD is recommended to reduce loss of checkpoint data in the event of a system failure while taking a checkpoint.
- This DD must define a sequential BSAM data set (BPAM is not supported).
- All the DCB parameters are provided by the checkpoint at EOV routine and should not be coded on the DD statement.

Example:

```
//SYSCKEOV DD DSN=CKPTDS,UNIT=TAPE,DISP=MOD
```

Checkpoint in Exit Routines

The CHKPT macro instruction must not be used in an exit routine other than the end-of-volume exit routine. You may take a checkpoint when a BSAM or QSAM data set reaches end-of-volume. For more detailed information about coding the EOV exit for physical sequential data sets, see *Data Facility Product: Customzation.*

Coding the DCB for a Checkpoint Data Set

Required DCB Parameters

You must provide either a DCB for the checkpoint data set or the name of the DD statement for the checkpoint data set. (*Data Administration: Macro Instruction Reference* contains detailed information about coding DCBs.) If a DCB is provided, the following parameters must be included:

- DSORG=PS or PO (BSAM or BPAM data set organization)
- MACRF=W (WRITE macro instruction)
- RECFM=U or UT (undefined record format)
- DEVD=DA or TA (direct access or tape device)
- TRTCH=C (data conversion with odd parity; parameter required only if the data set is on a 7-track magnetic tape)
- DDNAME= (name of DD statement for checkpoint data set)

You must code the DSORG, MACRF, and DDNAME operands in the DCB macro. The RECFM, DEVD, and TRTCH operands may be coded in the DCB macro instruction, or in the related DD statement. The RECFM and TRTCH may be used as subparameters of the DCB parameter. Because RECFM and DEVD have default values of U and DA respectively, they need not be provided explicitly in either the DCB macro or the DD statement.

DCB Options

You may, optionally, provide the following DCB parameters:

- OPTCD=W (write validity checking)
- RECFM=UT (track overflow)
- NCP=2 (number of channel programs)
- NCP=2 and OPTCD=C (chained scheduling)

These parameters are discussed in detail in *Data Administration: Macro Instruction Reference.*

Notes on the DCB for a Checkpoint Data Set

The following restrictions apply to the DCB for a checkpoint data set:

- The checkpoint routine writes all checkpoint records in both 400- and 4096-byte blocks.
- Requests for two channel programs or chained scheduling apply to the writing of all checkpoint records. Such requests do not apply to the reading of checkpoint records for a restart.
- OPTCD=Q cannot be specified in the DCB, because checkpoints cannot be taken to an ISO/ANSI/FIPS tape.

Coding the DD Statement for a Checkpoint Data Set

The DD statement for the checkpoint data set must define the data set in the usual way. The publication *JCL* contains detailed information on coding the DD statement. The only restrictions on the DD statement are:

- A checkpoint data set, because it is an output data set, may not be a concatenated data set.
- The UNIT parameter must specify a tape or direct access device supported by BSAM or BPAM. The device can be specified by referring to a specific device, a device type, or a group of devices. **If direct access is specified, the device may not be shared with another processor.** To avoid allocating to a shared DASD, a special generic device name should be generated (at system generation time) to include a nonshared DASD of a single device type, and this generic name should be used.
- DEFER should not be coded in the DD statement.

- DISP=SHARE may not be specified.

- Secondary space allocation may be requested by the increment subparameter (see "Notes on the DD Statement"). The increment subparameter must specify at least enough space to contain two complete checkpoint entries. For information on how to calculate the length of the checkpoint entry, see "Storage Estimates for Checkpoint Data Sets" on page 13.

- The LABEL parameter of the DD statement describes the labels of a data set on magnetic tape. For a checkpoint data set, you must specify only IBM standard labels (SL or SUL). Nonstandard labels (NSL), no labels (NL), or International Organization for Standardization/American National Standards Institute (ISO/ANSI) labels (AL or AUL) cannot be specified for a checkpoint data set. If the label type is not specified, the operating system assumes that the data set has IBM standard labels (SL or SUL).

- OPTCD=Q cannot be specified as a DCB subparameter.

- CHKPT=EOV is ignored (checkpoint resets the value in the JFCB).

- The RLSE subparameter of the SPACE parameter cannot be used on the DD statement for a checkpoint data set.

Notes on the DD Statement

- The initial disposition of the data set (as specified in the DISP operand of the DD statement) is used to position the checkpoint data set each time it is opened. A discussion is under "How Checkpoint Entries Are Written" on page 16. This describes a specific circumstance in which the only checkpoint entry that exists on a data set will be the last checkpoint taken whether that entry is valid or not.

- The final and conditional dispositions of the data set have their normal meanings. However, if termination is occurring and an automatic restart at a checkpoint is to occur, the system automatically keeps all data sets that are in use by the job, including the checkpoint data set.

- If end-of-volume is encountered while writing a checkpoint on a direct access or tape volume, message IHJ0001 (invalid checkpoint) is issued. Control is returned to you with a X'08' return code in register 15 and a reason code of 027 in register 0.

Examples of DD statements for the checkpoint data set are:

```
//ddname  DD   DSN=dsname,UNIT=TAPE,DISP=(MOD,KEEP)

//ddname  DD   DSN=dsname,UNIT=SYSDA,
//             DISP=(NEW,DELETE,KEEP),SPACE=(CYL,(15,17)),
//             VOL=SER=CKPTDS
```

Using Checkpoint Data Sets

The checkpoint data set must be empty at the beginning of a job step. For more information, see “How Checkpoint Entries Are Written” on page 16.

Using a Generation Data Set as a Checkpoint Data Set

You must take precautions when using the generation data set as the checkpoint data set. For a deferred checkpoint restart, the checkpoint data set, as specified in the SYSCHK DD statement, is allocated to the initiator, and an entry is made in the initiator’s generation data group name table (GDGNT). The GDGNT is never changed, unless a new level is cataloged, and remains for the life of the job and any future restarts under the same initiator. The GDGNT uses a data set from the same generation data group and uses the original contents of the GDGNT to obtain the qualified name of the generation data set. For more information, see “Generation Data Sets” on page 52.

After entry into the GDGNT, if the levels of the generation data group change in the catalog and the checkpoints are a different generation, the desired generation at restart time may not be the one retrieved as the checkpoint data set. To avoid this problem, do not make changes to the number of existing generations for the life of the IPL after the checkpoint data set is used for deferred restart. Be sure the same initiator is not used twice to do restarts from different levels of the same generation data group.

Storage Estimates for Checkpoint Data Sets

The checkpoint data set may be on any direct access device not shared by another process or drive. The following information can be used as a guideline in determining the size of this data set.

Figure 1 on page 14 contains the size and number of records written when a checkpoint is taken. The number of tracks or the amount of tape occupied by the checkpoint data set can be determined by applying the number of records and their sizes against either the track capacities of the direct access device or the recording density and type for the magnetic tape device.

Description of Record	Size (in bytes)	Number Required
Checkpoint header record (CHR)	400	1
DD name table (DDNT)	400	D/21
Data set descriptor record (DSDR)	400	A/2
Problem program image record (PPIR)	4096	B
Supervisor record (SUR)	4096	C
Subsystem checkpoint record (SSCR)		
SUBSYSTEM data sets	4096	1/SYSIN or SYSOUT data set
SPIE	20/4096	E/20
VSAM	4096	1

Figure 1 (Part 1 of 2). Determining the Size of a Checkpoint Data Set

Records Code:

A = the number of data sets defined in the job step

B = (Storage allocated - freespace + (X'14' * number of gaps in allocated storage))/X'FEC'

For example:

Storage allocated = 9000 to 4FFFF
and 1A000000 to 2FFFFFFF

Freespace=

Address	**Size**	**Number of Gaps**
A200 to A3FF	200	1
32900 to 35FFF	3700	1
40000 to 47FFF	8000	1
50000 to 19FFFFFF	0	1 (not allocated; counts as a gap)
1F900000 to 210F8FFF	17F9000	1
28700000 to 29FFFFFF	1900000	1
=	3104900	6

Therefore:

B = ((47000 + 16000000) - 3104900 + (X'14' * 6)) / X'FEC'

B = 130C0

C = A variable number of records for system and protected data management control blocks. This will be approximately three records plus one record for every three DASD data sets and one record for every ten non-DASD data sets.

D = Number of JCL-specified data sets dynamically unallocated at checkpoint.

E = One for every 90 ESPIEs established at checkpoint.

Note: All values are in hexadecimal.

Figure 1 (Part 2 of 2). Determining the Size of a Checkpoint Data Set

Dynamic Storage

Checkpoint/Restart's dynamic storage requirement varies with the resources used by the caller. The largest impact is caused by the amount of storage allocated to the job and the amount of storage fragmentation. Checkpoint/Restart's dynamic storage is allocated out of subpools 229 and 230.

Fragmented Storage

User fragmented storage may result in:

- Reduced performance of a checkpoint
- Insufficient storage for a checkpoint to contain a description of the user's storage (checkpoint is not completed).

How Checkpoint Entries Are Written

Nonopened DCB Passed to Checkpoint

- Sequential data sets
  - When a sequential data set's disposition is NEW or OLD, the checkpoint entry is written at the beginning of the data set, even if a previous checkpoint call was written on an entry.
  - When a sequential data set's disposition is MOD, the checkpoint entry is written after the last entry existing in the data set.
- Partitioned Data Sets
  - When the data set is partitioned, each checkpoint entry is a member, and its checkid is its member name. After it writes a checkpoint entry, the checkpoint routine executes the STOW macro to add the checkid of the entry to the directory of the data set.
  - If the data set is partitioned, regardless of its disposition, the checkpoint entry is written after the last entry existing in the data set.
  - If an identical checkid already exists in the directory, the related address of a member is changed and becomes the address of the new checkpoint entry. The initial disposition specified for the checkpoint data set has no effect on the STOW operation.

Opened DCB Passed to Checkpoint

If your program has a DCB and opens the checkpoint data set for output, the checkpoint routine writes a checkpoint entry at the data set's current position and does not close the data set. A user-opened checkpoint data set does not have to be closed after taking the last checkpoint for the job step. All the checkpoint entries are saved in this case, without specifying DISP=MOD. This provides the ability to request a deferred restart from any of the checkpoints. If the data set is partitioned, the checkpoint routine executes the STOW macro at the completion of each entry.

- **ddname passed to checkpoint**

  If your program provides a ddname instead of a DCB, the checkpoint routine opens the checkpoint data set the first time it receives that ddname. If your program has a different ddname on the next execution of the CHKPT macro, the data set for the previous ddname is closed and the data set for the new ddname is opened. The open checkpoint data set is closed by step termination.

  For sequential data sets, DISP=OLD, NEW, or MOD has no effect on how the checkpoint entries are written on the checkpoint data set. The first time each checkpoint data set is used, the checkpoint entry is written at the beginning of the file and each following entry is written after the previous entry.

  Partitioned data sets are treated in the same manner as nonopened DCBs passed to checkpoint.

If end-of-volume is encountered during writing of a checkpoint entry on tape, a second attempt is made to create the checkpoint entry on another tape if that other tape has been allocated. If EOV occurs again before the entry has been completed, message IHJ0001 (invalid checkpoint) is issued. Control is returned to you with a X'08' return code in register 15 and a reason code of 27 in register 0.

The status (opened or closed) and position of a checkpoint data set remain the same at restart as they were after execution of the CHKPT macro instruction that established the checkpoint.

The first time a checkpoint data set is opened, checkpoint routines cause open to request security information for the data set from the operator.

Note: A checkpoint data set must contain only checkpoint entries. If a program attempts to write its own data in a checkpoint data set, message IEC954I will be issued, followed by an abend code of 23F.

How to Ensure Restart with Sequential Checkpoint Data Sets

Because abnormal termination or system failure may occur while the new entry is written (and to ensure that restart at the most recent valid checkpoint is possible), a checkpoint entry must not be written over a preceding checkpoint entry. Four methods by which you can ensure that restart is possible are suggested in the figures below. All four methods use sequential checkpoint data sets.

Note: Because of the characteristics of partitioned data sets, checkpoint entries with dissimilar member names are not written over each other.

Figure 2 on page 18 shows the use of one sequential checkpoint data set, one data control block, and one DD statement (CHECKDD) specifying MOD disposition. The user allows the checkpoint routine to open and close the data set each time it writes a checkpoint entry. Checkpoint entries are written sequentially in the data set.

```
          Program

            .
            .
            .
            CHKPT     CHKDCB
            .
            .
            .
CHKDCB      DCB  DDNAME=CHECKDD,MACRF=W,DSORG=PS
            .
            .
            .

           DD Statement

//CHECKDD   DD  UNIT=TAPE,DISP=(MOD,KEEP)
```

Figure 2. Using One Sequential Checkpoint Data Set to Ensure Restart

An alternative method of using one sequential checkpoint data set, one DCB, and one DD statement is to have the your program open the data set and leave it open. The disposition may be NEW, OLD, or MOD.

Figure 3 on page 19 shows a way to alternate data sets when all checkpoints are taken by one CHKPT macro instruction. The data sets are opened by the control program and identified by two DD statements, CHECKDD1 and CHECKDD2. The data control block initially refers to CHECKDD1. Before the second checkpoint, it is changed to refer to CHECKDD2; before the third checkpoint, it is again changed to refer to CHECKDD1; and so forth. One data control block can be used for two data sets that are not open at the same time.

An alternative method of using two sequential data sets is to use two DCBs and two DD statements specifying NEW or OLD dispositions, and to execute alternately two CHKPT macro instructions, each referring to a different data set.

The method illustrated in Figure 2 saves all checkpoint entries for possible use in deferred restart; the method illustrated in Figure 3 conserves auxiliary storage. None of the methods requires a particular device type.

```
          Program

            .
            .
            .

            DCBD  DSORG=PS             Define IHADCB (dummy
*                                      section that defines
*                                      DCBDDNAM)
            CSECT                      Resume original control
*                                      section
            .
            .
            .

            LA    2,CHECKDCB           Establish CHECKDCB as
*                                      base address for
            USING IHADCB,2             IHADCB
            XC    DCBDDNAM(8),DDHOLD   Exchange ddname in
            XC    DDHOLD(8),DCBDDNAM   CHECKDCB for ddname
            XC    DCBDDNAM(8),DDHOLD   in DDHOLD
            CHKPT CHECKDCB             Open, checkpoint, close
            .
            .
            .

DDHOLD      DC    C'CHECKDD1'
CHECKDCB    DCB   DSORG=PS,MACRF=(W),DDNAME=CHECKDD2

          DD Statements

//CHECKDD1 DD UNIT=SYSDA,DISP=NEW ...
//CHECKDD2 DD UNIT=SYSDA,DISP=NEW ...
```

Figure 3. Using Two Sequential Checkpoint Data Sets to Ensure Restart

How Checkpoint Entries Are Identified

Any number of checkpoint entries can be written in a checkpoint data set, and any number of checkpoint data sets can be used concurrently. In a sequential checkpoint data set, checkids of valid or invalid checkpoint entries in one data set should be unique. In a partitioned data set, checkids of valid entries should be unique.

If you specify checkids instead of having the system generate them, incorrect duplicates may be specified. The system does not recognize this error. When deferred restart at a checkpoint occurs and the checkpoint data set is sequential, the system searches the data set from its beginning for the specified checkpoint entry. It uses the first entry it finds that has the specified checkid.

If the data set is partitioned, the system searches the data set's directory to find the location of the specified checkpoint entry. If two or more entries having the same checkid were written in the data set, the most recent of those entries is the one pointed to by the directory, and restart occurs from the most recent entry.

Checkpoint entries have two identifications—primary and secondary.

- The primary identification is the programmer-generated or system-generated checkid specified or requested by the CHKPT macro instruction. The primary identification is used when a search is made for a checkpoint entry.

- The secondary identification is identical to the system-generated checkid that might have been requested by CHKPT. The secondary identification is then used as a base to compute the system-generated checkids of entries written after restart has occurred.

The search of the data set's directory prevents the system from generating checkids that are duplicates of checkids of existing useful entries.

The control program identifies each checkpoint in a message to the operator; on request, it also makes the identification available to your program. In Figure 4, the CHKPT macro instruction requests the control program to supply an identification ('S' parameter) and place it in the 8-byte field named ID. When the checkpoint is successfully taken, the program prints the identification as part of a message to the programmer.

```
         .
         .
         .
         CHKPT    CHKDCB,ID,'S'    Take checkpoint
         LTR      15,15            Was checkpoint taken
         BNZ      PHASE2           No, branch to PHASE2
         PUT      STEPLOG,MESSAGE  Yes, print
*                                  checkpoint ID
PHASE2
         .
         .
         .
MESSAGE  DS       0F
         DC       H'45'            Record length
         DC       H'0'             Flags
         DC       C'SUCCESSFUL CHKPT AT PHASE2...ID='
ID       DS       CL8
STEPLOG  DCB      DSORG=PS,MACRF=(PM),                        X
                  RECFM=V,BLKSIZE=128,                        X
                  LRECL=124,DDNAME=LOGDD
CHKDCB   DCB      DSORG=PS,MACRF=(W),RECFM=U,                 X
                  DDNAME=CHKDD;
```

Figure 4. Recording a Checkpoint Identification Assigned by the Control Program

Coding the CHKPT Macro Instruction

The CHKPT macro instruction is coded in your program. When the CHKPT macro executes, job step information about the user's program, virtual-storage data areas, data set position, and supervisor control is written as a checkpoint entry in a checkpoint data set. The point at which this information is saved becomes a checkpoint from which a restart may be performed. After the checkpoint entry is written, control returns to your program at the instruction following the CHKPT macro.

The CHKPT macro instruction refers to the data control block (DCB) for the checkpoint data set. If the data set is not open, the checkpoint routine opens it and then closes it after writing the checkpoint entry. If the data set is open, the checkpoint routine writes the checkpoint entry, but does not close the data set.

The checkpoint data set must be on one or more magnetic tape volumes or direct access volumes. A checkpoint data set that resides on a magnetic tape must have IBM standard labels. Direct access volumes cannot be shared. (For restrictions on shared DASD, see "Coding the DD Statement for a Checkpoint Data Set" on page 11.)

The standard form of the CHKPT macro instruction is:

[*symbol*]	**CHKPT**	{*dcbaddr* | **DDNADDR=***ddnaddr* | **CANCEL**} **[,{***checkid addr* | **IDADDR=***checkid addr***}** **[,{***checkid length* | **IDLNG=***checkid length* | **'S' | IDLNG='S'}]]** **[,MF={S | (S,***addr***)} | {L | (L,***name***)} |** **{(E,***addr***) | (E,***addr1***,***addr2***)}]**

The options on the CHKPT macro are discussed in alphabetic order below.

CANCEL
cancels the request for automatic checkpoint restart. CANCEL can be used only with the standard form (MF= omitted or MF=S or MF=(S,NAME)). An automatic checkpoint restart can be suppressed by issuing a CHKPT with CANCEL. See "Canceling a Checkpoint" on page 25.

checkid addr
specifies the address of a programmer-provided field that is to contain a unique, printable identification of the checkpoint entry. This identification is called a checkid. The checkpoint routine writes the checkid as part of the entry and prints it in a message on the operator's console when it finishes writing the entry. The programmer must use the checkid and code it in the JOB statement RESTART parameter to use the corresponding entry to perform a deferred restart at a specific checkpoint. If the *checkid addr* operand is omitted, the *checkid length* or 'S' operand is invalid. Valid characters for a checkid are:

A - Z

0 - 9

$ #

! *) ;   - / , % _ > ? : ' = " and blanks are valid except as first characters or for partitioned data sets.

checkid addr is an RX type address or register (2-12).

checkid length **or 'S'**

checkid length is the length in bytes of the field that contains the checkid. The maximum length of this field is 16 bytes when the checkpoint data set is physical sequential, 8 bytes when it is partitioned. By coding this operand or by omitting it entirely (in which case a length of 8 bytes is implied), you specify that your program will form an identification and store it in the checkid field before CHKPT is executed. If the *checkid addr* operand is omitted, this operand is invalid.

By coding this operand as 'S', you specify that the checkpoint routine is to generate an identification 8 bytes in length and store it in the checkid field. If the *checkid addr* operand is omitted, this operand is invalid.

If both *checkid addr* and *checkid length* or 'S' and the keyword equivalents are omitted, the checkpoint routine generates an identification and writes it in the checkpoint entry and on the operator's console, but does not return it to your program.

If you provide the checkpoint identification and the checkpoint data set is sequential, the identification can be any combination of up to 16 alphamerics, special characters, and blanks. For a partitioned data set, it must be a valid member name of up to eight alphamerics. The identification for each checkpoint should be unique. If two identifications differ only by having a different number of trailing blanks, the control program considers them to be the same. (For further information, see "How Checkpoint Entries Are Identified" on page 19.) When a deferred step restart takes place, this number is reset to 0.

One way to use the *checkid addr* operand, or its keyword equivalent, is to allow your program to select fields in the records of an input data set and use them as checkids. Alternatively, your program may use the *checkid addr* and the 'S' operands and include a system-generated checkid in the current record of an output data set.

dcbaddr

is the address of the DCB for the checkpoint data set. *dcbaddr* is an RX type address or register (2-12).

DDNADDR=

Points to an 8-character, fixed-length field that has the name of the DD statement for the checkpoint data set. (This keyword may be used instead of the first positional operand DCB.) This keyword is mutually exclusive with the positional operand DCB.

ddnaddr is an RX type address or register (2-12).

IDADDR=

Points to a variable-length field that contains, or receives, the checkid. (This keyword may be used instead of the second positional operand, *checkid addr*.) This keyword is mutually exclusive with and follows the same rules as the second positional operand.

checkid addr is an RX type address or register (2-12).

IDLNG=

Specifies the length of the field that contains, or receives, the checkid. (This keyword may be used instead of the third positional operand, *checkid length*.) This keyword is mutually exclusive with and follows the same rules as the third positional operand.

Keyword MF= Standard Form (default)

MF=S

Generates an inline parameter list and executable code.

MF=(S,*name*)

Generates executable code and an inline parameter list with the label name at the beginning of the list. *name* is a symbol.

Keyword MF= List Form

MF=L

Generates a parameter list. (*symbol* is required with this form.)

MF=(L,*name*)

Generates a parameter list with the label name at the beginning of the list. *name* is a symbol.

Keyword MF= Executable Form

MF=(E,*addr*)

Generates executable code reference to the parameter list whose name is *addr*. *addr* can also be expressed as a register, for example, MF=(E,(R5)), if the register is preloaded to point to a parameter list.

MF=(E,*addr1*,*addr2*)

Generates executable code that moves a model parameter list from *addr2* to *addr1* and updates the list in *addr1* with the positional or keyword operands specified. *addr1* and/or *addr2* may be expressed in register form, for example, MF=(E,(R5),(R6)) if the register or registers are preloaded to point to the respective parameter list. This form is useful for programs that must be reentrant.

For further information on the MF= keyword, see *Data Administration: Macro Instruction Reference*.

CHKPT Return Codes

The CHKPT macro returns a code in register 15 to indicate whether the CHKPT macro instruction executed successfully. If the value in register 15 does not equal 0, register 0 gives a more detailed code. Appendix A, "Checkpoint/Restart Codes" on page 71, contains a list of these codes and their meanings.

Use of CHKPT with Other Macro Instructions

EXTRACT: The EXTRACT macro obtains information from the task control block (TCB). TCB information may change when the task terminates and the job step restarts. If the information is needed after restart, the EXTRACT macro instruction should be reissued after the checkpoint is taken, as shown in Figure 5.

```
          .
          .
          .
          EXTRACT    ANSADDR,FIELDS=(ALL)   Obtain TCB
*                                           information
          .
          .
          .
          CHKPT      CHKPTDCB               Establish
*                                           checkpoint
          CH         15,=H'4'               Is restart in
*                                           progress
          BNE        NRESTART               No, branch to
*                                           NRESTART
          EXTRACT    ANSADDR,FIELDS=(ALL)   Yes, obtain new
*                                           information
NRESTART
          .
          .
          .
```

Figure 5. Obtaining Updated TCB Information after Restart

SETPRT: The SETPRT macro in data management selects the universal character set (UCS) buffer for an IBM 3203-5, 3211, 4245, 4248, or 1403 Printer with the Universal Character Set feature and the forms control buffer (FCB) for the 3203-5 or 3211 Printers, 3262 Model 5, or the 3800 Printing Subsystem.

For 3800, 3262, 4248, and 4245 Printers, the buffer contents are not saved when a checkpoint is taken. To ensure that all lines are printed before taking a checkpoint, you should take one of the following actions:

- Close the printer with the CLOSE macro instruction.
- Issue the "Clear Printer" command.
- Reprint the data after restart.

For the 3800 Printing Subsystem, (Model 1 or Model 3 in compatibility mode), the SETPRT macro instruction initially sets or dynamically changes the control information for the printer. For additional information on the SETPRT macro, see *IBM 3800 Printing Subsystem Programmer's Guide* and *Data Administration: Macro Instruction and Reference.*

For the 3800 Printing Subsystem (Model 1 or Model 3 in compatibility mode) at checkpoint restart, the job has the JFCBE parameters as modified during the last JFCBE exit. If no exit was requested or if the JFCBE was not flagged as being modified during the exit, the JFCBE reflects the values coded in the restart JCL. For additional information on the JFCBE exit, see *Data Administration Guide*

WTOR: The reply to a WTOR macro instruction must be received before a CHKPT is issued.

STIMER: A time interval established by the STIMER macro must be completed before a CHKPT is issued.

ATTACH: If ATTACH is issued in the program using CHKPT, all subtasks created must terminate before a CHKPT is issued; the job-step task must be the only task of the step.

SETDEV: If an IBM 3890 Document Processor unit-record device is used, the following SETDEV macro instruction must be issued after a successful restart:

3890 device: SETDEV=*dcbaddr*, DEVT=3890, IREC=*irecaddr* [,PROGRAM=*progname*]

where:

dcbaddr
: is the address of the associated DCB.

irecaddr
: is the address of an initialization record.

progname
: is the name of the stacker control instruction program loaded in SYS1.IMAGELIB (this parameter is optional).

PCLINK: If a PCLINK STACK is issued, a PCLINK UNSTACK must also be issued against all existing stacks before a CHKPT can be issued. There must not be any PCLINK STACKs in existence when a CHKPT macro is issued.

Canceling a Checkpoint

There are certain conditions under which you may not wish to have a job automatically restarted in case of failure.

- If you have not yet taken a checkpoint and the job has already been restarted.
- If you are updating critical data that must be complete and accurate before an automatic restart can be successful.

In these cases, you can take a checkpoint, but avoid automatic restart, by using the CANCEL option on the CHKPT macro. If CHKPT CANCEL is used after a checkpoint is taken, the job is not automatically restarted after failure. For the format of the CHKPT macro, see "Coding the CHKPT Macro Instruction" on page 21.

After being restarted, the job step may again terminate abnormally. If it does, it may be restarted from the same checkpoint, subject to operator authorization. To avoid restarting the job step twice from the same checkpoint, the sequence shown in Figure 6 may be coded.

```
          .
          .
          .
          CHKPT     CHKPTDCB  Establish checkpoint
          CH        15,=H'4'  Is restart in progress
          BNE       NRESTART  No, branch to NRESTART
          CHKPT     CANCEL    Yes, cancel restart request
NRESTART
          .
          .
          .
```

Figure 6. Canceling a Request for Automatic Restart

After the successful initiation of a checkpoint restart, the system places a return code of hexadecimal 04 in register 15 and returns control to your program at the instruction following the CHKPT macro instruction. At this time, a request for another automatic restart at the same checkpoint is normally in effect. In Figure 6, the instruction that follows the CHKPT macro instruction tests the return code to determine whether control has been returned as the result of a restart. If the return code is 04, a restart has just occurred, and a second CHKPT macro instruction is executed. This macro instruction has a CANCEL operand, which cancels the existing request for an automatic restart. If the job step again terminates abnormally after a restart from the checkpoint, automatic restart can occur only at the beginning of the step. It will not occur at the checkpoint preceding the canceled checkpoint.

Shared and Serially Reusable Resources

Requesting Serially Reusable Resources

When a job step terminates, it loses control of serially reusable resources. If the step is restarted, it must request all the resources needed to continue processing. Explicit use of a serially reusable resource is requested when the user's program issues the ENQ macro instruction. If the program issues the ENQ and takes a checkpoint, it must issue the ENQ again whenever restart occurs at the checkpoint.

Figure 7 shows a program that requests a serially reusable resource by issuing an ENQ before establishing a checkpoint. After the checkpoint, it tests for a restart. If one has occurred, it requests the same resource again. It requests the resource again because the job step terminated, lost control of the resource, and then restarted from the checkpoint.

```
            .
            .
            .
            ENQ   (QADDR,RADDR)
            .
            .
            .
            CHKPT CHKPTDCB
            CH     15,=H'4'
            BNE    NRESTRT
            ENQ    (QADDR,RADDR)
NRESTRT
            .
            .
            .
            DEQ    (QADDR,RADDR)
            .
            .
            .
```

Figure 7. Requesting a Resource after Restart

Some serially reusable resources are requested implicitly by issuing data management macro instructions. These resources may be records that you are processing or tracks on a direct access device. To ensure correct processing, you must *not* establish checkpoints during control of these resources:

- If the basic direct access method (BDAM) is used, your program must, before executing the CHKPT macro, execute either the WRITE or the RELEX macro to release a record that is read with exclusive control.

- If BDAM is used to add a record to a data set with variable-length or undefined records, BDAM issues an ENQ macro instruction for the capacity record (R0). Your program must execute the WAIT or CHECK macro to check completion of the write operation before it executes CHKPT. After execution of the WAIT or CHECK macro, the resources are dequeued.

- If the basic indexed sequential access method (BISAM) is used, a checkpoint must not be taken before completion of a write operation. If a record is read for update, a checkpoint must not be taken before writing the updated record and waiting for the write operation to be checked.

- If the queued indexed sequential access method (QISAM) is used, and if a SETL macro instruction was issued, an ESETL macro instruction must be issued before taking a checkpoint. Another SETL macro instruction may be issued after the checkpoint.

- If a VSAM cluster is implicitly specified, the restart program obtains the names of the data set and the index from the catalog and reissues an ENQ macro against each of them; therefore, no special considerations are required.

Tape Volume DEQ at Demount

Tape volume DEQ at demount is not affected by deferred or automatic step restarts. If restart is from a checkpoint, all volumes that are dequeued during the processing of the job are enqueued when the restart job is initiated. This includes all volumes referenced in the JFCB at the time of checkpoint. Subsequent restart processing tests each tape data set open at the time of checkpoint to determine whether the following conditions are satisfied:

1. The data set was open for OUTPUT, OUTIN, EXTEND, OUTINX, or INOUT when the checkpoint was taken.

2. JFCDQDSP was set to 1 in the JFCB at the time of checkpoint.

3. The program restarting is APF-authorized.

If these conditions are satisfied, the DEQ at demount is reestablished so that the data set's current volume is dequeued when it is demounted. All volume serials in the JFCB and JFCB extensions prior to the current volume are dequeued during the restart process for that data set.

Because all volumes, including those dequeued through the DEQ at demount, are enqueued at restart, if only for a short time, automatic checkpoint restart may result in a "waiting for volumes" condition if any of the dequeued volumes are allocated to another job. A deferred checkpoint restart can be timed to avoid such a delay.

Special Operating System Features

Shared DASD

At some installations, a direct access storage device is shared by two or more independent computing systems. This device is a serially-reusable resource. If it is being used when a checkpoint is taken, it must be requested after a restart from the checkpoint. This resource is requested by a special macro, RESERVE, described in *Supervisor Services and Macro Instructions*. For restrictions on shared DASD, see "Coding the DD Statement for a Checkpoint Data Set" on page 11.

When using dynamic allocation, the following should be considered:

- For data sets that are not opened during the original execution when nonspecific tape volumes were requested, the volumes assigned at restart may not be the same as those the referenced DD initially assigned. (This occurs when the DD statement specified VOL=REF= to a DD that was unallocated at the time the checkpoint was taken.)

- If a VIO data set is dynamically allocated prior to a checkpoint and unallocated after checkpoint, the step is ineligible for restart until the next checkpoint is taken.

VIO Data Sets

If a VIO data set is open when the checkpoint is taken, it is supported for automatic restart only.

If restart is deferred, VIO data sets for BSAM or QSAM must be redefined as dummy data sets. If they are not redefined or if an access method other than BSAM or QSAM accesses the VIO data set, the job fails.

Dynamic Allocation

Checkpoint/Restart supports the use of dynamic allocation by problem programs.

Shared Resources

Checkpoint/Restart is supported for the local shared resources feature, but repositioning is not allowed. Also, checkpoint is not allowed and cannot be taken when the global shared resources feature is used.

For automatic checkpoint restart, deleting a data set open at the time the checkpoint was taken or deallocating a SYSOUT data set to the job entry subsystem makes the step ineligible for restart until the next checkpoint is taken.

For automatic step restart, any of the following makes the step ineligible for restart:

- KEEP, CATLG, or UNCATLG a NEW JCL specified data set.
- DELETE JCL specified data set whose initial status upon entry to the step was OLD.
- DELETE an OLD dynamically allocated data set that had volumes specified when allocated.
- UNCATLG JCL specified data set whose initial status upon entry to the step was OLD and did not have volumes specified.

Cross-Memory Support Restrictions

A checkpoint is not allowed when storage access across address space boundaries has been established. When using cross-memory support, the restrictions shown in Figure 8 on page 30 may apply.

Macros	Restriction Code	Instructions	Restriction Code
LXRES	2	EPAR	2
ETCRE	2	ESAR	2
ETCON	2	IAC	2
AXRES	2	IVSK	2
AXEXT	2	MVCP	3
AXSET	2	MVCS	3
ATSET	2	MVCK	3
PCLINK	1	PC*	4
		PT*	3 and 4
		SAC	1
		SSAR	1

Code Definitions:

1. You must release this resource before checkpoint and reestablish it after checkpoint or restart.

2. You must reestablish this resource after a restart.

3. You must be sure that the keys and/or ASIDs are correct after a restart.

4. Any attempt to restart from a checkpoint taken in a program call (PC) routine is unpredictable. If a PC routine issues a PCLINK, the error is detected and a checkpoint request is refused. It is impossible to detect a PC routine when the PC routine does not issue a PCLINK and the PC routine is entered without a space switch. If a PC routine issues a checkpoint, that PC routine must ensure that all the requisite values needed for a PT instruction are current before issuing the PT instruction. These values may change after a restart (for example, the ASID might be different).

Figure 8. Cross-Memory Support Restrictions

Chapter 3. Requesting Restart

This chapter explains how you may request restart. The topics discussed are:

- Automatic restart
  - Automatic checkpoint restart
  - Automatic step restart
- Deferred restart
  - Deferred checkpoint restart
  - Deferred step restart
- Coding the RD and RESTART parameters

Automatic Restart

Because an automatic step restart and an automatic checkpoint restart are similar, they are discussed together in the following sections.

Operator Message Sequence

During processing related to an automatic checkpoint restart, the system issues the following sequence of messages to the operator:

1. A message is issued each time a checkpoint entry is written. Each message contains the checkpoint identification.
2. If the job step terminates because of an abend condition, an abend message is issued for the job step.
3. If the abend code makes the job step eligible for restart and a job journal is present, an authorization for restart message is issued that requires a reply.
4. A message is issued, indicating the virtual storage requirements (beginning address and ending address) of the job step to be restarted.
5. Normal mount and restart messages are issued.

6. If password-protected data sets are to be repositioned at restart time, a password message is issued.

7. If additional checkpoint data sets (other than the data set used for restart) are encountered at restart time, checkpoint data set security messages are issued.

8. A successful restart message is issued.

During processing related to an automatic step restart after a job step has terminated abnormally, the sequence is the following:

1. An abend message is issued for the job step.

2. If the abend code makes the job step eligible for restart, and a job journal is present, an authorization message is issued that requires a reply.

3. Normal mount messages are issued.

Note: The abend message, which is issued as:

IEF4501 *jobname.stepname.procstepname* **ABEND** *code*

is always displayed if a job step terminates abnormally. In addition, if the job step is being executed and the system fails, this message will be displayed during the next IPL if a system-supported restart is performed. The *code* part of the message has the form Shhh (S followed by a 3-character hexadecimal number) if the system executed the abend macro instruction, or Udddd (U followed by a 4-digit decimal number) if your program executed the abend. It is S2F3 if a system failure occurred. For additional information on messages, see *System Messages*.

Operator Considerations

If a step requests automatic restart and is eligible for restart, the system displays the following message to request authorization for the restart:

*xx***IEF225D SHOULD** *jobname.stepname.procstepname*
[*checkid*] **RESTART**

Checkid appears in the message only if restart at a checkpoint is requested. It contains from 1 to 16 characters and identifies the checkpoint entry to be used to perform the restart. The operator must reply to the request for authorization as follows:

REPLY *xx*,**{'YES' | 'NO' | 'HOLD'}**

YES authorizes the restart, HOLD postpones it, and NO prohibits it.

If the advisability of allowing the restart is not readily apparent, the operator should reply HOLD to the authorization message. Later, if restart should be allowed, initiate the restart by using the RELEASE command, thereby achieving the same result as with an initial YES reply. If the decision is to deny the restart authorization, the operator can cancel the job in the HOLD queue. However, that HOLD, as well as YES, causes special disposition processing to occur during the abnormal termination. This processing keeps all OLD (or MOD) data sets and deletes all NEW data sets if a step restart is requested and keeps all data sets if a checkpoint restart is requested. If you decide to disallow the restart but want to allow normal disposition (as requested on the job's DD statements) of data sets that were kept, release the job, wait until restart has begun, and then cancel the job.

If a job is canceled in the HOLD queue, any paging space allocated to a VIO data set is not released until the system is reinitialized. To avoid this situation, the operator should release the job, wait until the restart begins, and then cancel the job.

After the authorization request and before the operator replies YES, using the VARY and UNLOAD commands may cause the system's volume and device configuration during a restart execution of the job to be different from what it is during the original execution of the job. The operator may eliminate use of defective volumes and devices.

The ability to use a different volume usually exists only in the case of a NEW data set on a nonspecific volume. If a checkpoint restart is performed, the data set must not be open at the checkpoint. The ability to use a different device does not apply to the device or devices containing the SYSRES, SPOOL, or PAGE packs. If a checkpoint restart is to be performed and a data set is open at the checkpoint, the same type of device must be allocated to the data set during both the original and restart executions.

Note: When an initiator selects a job for automatic step restart and the job is reinterpreted, no message is issued to the operator regarding virtual storage requirements, because its execution is not location dependent.

If ADDRSPC=REAL is specified on the JOB or EXEC statement, the restart may be delayed by the system waiting for the allocation of storage. If another job is using the required storage, you do not receive a message—only a delay. Enter a DISPLAY A command to see if a system task or another job is using the storage required by the job with a nonpageable region. You may stop or cancel the conflicting task or job.

How to Request Automatic Checkpoint Restart

An automatic checkpoint restart occurs if:

- The job journal option is specified, or
- The step requests restart (RD=R), and
- A successful checkpoint has been taken, and

- The step is eligible for restart because it was terminated by an abend macro that returned an eligible completion code (specified by the CKPTREST macro at system generation), or because system failure occurred.

- The operator authorizes the restart. This authority enables the operator to control the number of restarts of the same step or from the same checkpoint.

If a step fails and an automatic checkpoint restart is requested, restart occurs automatically at the last checkpoint taken.

Execution of the CHKPT macro requests this type of restart and establishes the checkpoint. You must provide an ordinary DD statement for the checkpoint data set.

Figure 9 illustrates a job requesting automatic restart at a checkpoint.

```
//MYJOB     JOB     MSGLEVEL=1[1]
//STEP1     EXEC
             .
             .
             .
//STEP2     EXEC    PGM=MYPROG    MYPROG issues the CHKPT macro
//NAME1     DD      DSN=NAME2     Describes the data set into
//*                               which checkpoint entries
//*                               are to be written
```

Figure 9. Requesting Automatic Checkpoint Restart

Note to Figure 9:

1 MSGLEVEL=1 is optional.

Automatic Step Restart

An automatic step restart occurs if all of the following are satisfied:

- The step requests restart (RD=R or RD=RNC); and

- The step is eligible for restart because it was terminated by an abend macro that returned an eligible completion code (specified by the CKPTREST macro at system generation), or because system failure occurred; and

- The operator authorizes the restart. This authority enables the operator to control the number of restarts of the same step or from the same checkpoint.

If a step fails and automatic step restart is requested, restart occurs automatically at the beginning of the step that failed.

Automatic step restart may be requested by coding the RD parameter (**RD=R** or **RNC**) on the JOB or EXEC statement in the originally submitted job deck. Checkpoint processing is suppressed if **RD=RNC**.

Figure 10 illustrates the JCL of a job requesting automatic step restart.

```
//MYJOB    JOB   MSGLEVEL=1[1],RD=R   Requests automatic
//*                                   restart at the
//*                                   beginning of any step
//*                                   that terminates
//*                                   abnormally
//STEP1    EXEC
//STEP2    EXEC PGM=MYPROG,RD=R[2]    Requests automatic
//*                                   restart of STEP2 if it
//*                                   terminates abnormally
//STEP3    EXEC
```

Figure 10. Requesting Automatic Step Restart

Notes to Figure 10:

1 MSGLEVEL=1 is optional.

2 Note that if RD=R appears on the JOB statement, it is not required on the EXEC statement.

How MOD Data Sets Are Handled during Automatic Step Restart

When automatic step restart is requested, the system saves, for each MOD data set that is on a direct access volume and used by the step, the TTR (and track balance) of the end of the data set. Saving occurs when each data set is first opened. If restart occurs, the saved TTRs indicate the ends of the data sets when the data sets open again. Thus, if the step writes data in such a data set during the original execution, the step writes over the data during the restart. The action described here does not occur if restart at a checkpoint occurs.

If a MOD data set on tape is used in the restart step, the data set is not repositioned at the start of the restart execution. Data written into it during the restart execution follows the data written during the original execution. You may reposition the data set so the data written during the restart execution overlays the data written during the original execution.

If the data set opens with the OUTINX or EXTEND options, and its DD disposition parameter was

DISP=NEW: there are no restrictions, because the data set reallocates prior to the restart.

DISP=OLD or **DISP=SHR:** the data set repositions after the last record added before the job terminated. It is possible to add records twice to the data set.

DISP=MOD: the following restrictions apply to a multivolume data set:

- If the volume sequence doesn't point to the last volume, records overlay or an abend occurs.

- If the volume sequence is not supplied, loading starts on the last volume on which the data set exists. For TAPE data sets only, if the end of data occurs on a volume other than the last allocated volume, processing starts on the wrong volume.

Caution Concerning Automatic Step Restart after a Checkpoint Restart

If a step executes as the result of an automatic or a deferred checkpoint restart, and if you attempt an automatic step restart of this step, and if the JCL of the step refers to any new data sets on direct access volumes, the attempt may be unsuccessful. When the step is initiated during the checkpoint restart, the failure occurs because all the step's data sets that have a NEW disposition are changed to a disposition of OLD by the system.

When the special disposition processing that prepares for a step restart occurs, all data sets used by the step appear as OLD and are kept. When the step restart occurs, the scheduler tries to obtain space for data sets specified as NEW in the JCL for the step. If the attempt for data set space is made on the volume that already contains the data set, the failure occurs because of the apparent presence of a "duplicate DSCB on direct access volume."

JCL Requirements and Restrictions on Automatic Restart

To allow an automatic step restart or an automatic checkpoint restart, you must observe the following rules when preparing the job deck used in the original execution:

1. If a step restart is desired, the RD parameter must be coded to request the restart.

2. If a checkpoint restart is desired, a DD statement for the checkpoint data set must be included in the step that executes the CHKPT macro instruction.

3. The EXEC statements in the job deck must have unique names. (Upon restart, the system searches for a named step.)

4. If commands are in the original deck, the commands are not reexecuted when restart occurs.

5. The DD statement VOL=REF parameter is ignored if restart is attempted from a checkpoint taken when the DD statements are opened out of order and the referenced DD statement requested nonspecific tape volumes.

Resource Variations Allowed in Automatic Restart

The system's device and volume configuration during a restart execution of a job can differ from the status during the original execution of the job.

The ability to use a different volume exists only in the case of a new data set on a nonspecific volume. If a checkpoint restart is to be performed, the data set must not be open at the checkpoint. The ability to use a different device does not apply to the device or devices containing the SYSRES volume and the LINKLIB data set.

Also, if a checkpoint restart is to be performed, the same device type must be allocated to the data set during both the original and restart executions.

How the System Works at Automatic Restart

How Data Set Disposition Is Determined: When a step requests restart and is eligible for restart, disposition processing of the data sets used by the step or by the job does not occur until the operator replies to the request for authorization. If the operator denies restart, disposition processing occurs normally and programmer-specified final or conditional dispositions are performed. If you indicate that a step execute after abnormal termination, the step is executed. The next step in a job that will be executed based on abnormal termination will not execute if a restart is successfully performed.

- If step restart is to occur, all data sets with OLD or MOD dispositions in the restart step and all data sets passed around the restart step are kept, even if they are declared temporary. Temporary data sets normally cannot be kept.
- All data sets with NEW dispositions in the restart step are deleted.
- If a checkpoint restart is to occur, all data used by the job (data sets that were not previously disposed of) is kept.

If the operator authorizes restart, any step to be executed after abnormal termination will not be executed, because the results appear as if abnormal termination did not occur.

If the operator performs an operator-deferred restart by replying HOLD to the request for authorization, a CANCEL command for the job may be issued instead of a RELEASE command. If CANCEL is issued, no further data set disposition processing or step executions occur. The disposition of these data sets remains as it was when the HOLD was issued.

How the Job Deck Is Reinterpreted and the SWA Merged: After disposition processing is completed, the job is reenqueued by the job entry subsystem and is eligible for selection. After the job is selected, the system begins restart processing by reinterpreting the job deck and creating a new scheduler work area (SWA) containing the required job-related information. The system uses an internal representation of the original job deck to perform this function. The job is not read in again.

After SWA is re-created, its contents are updated by the system with information saved on the job journal. If automatic checkpoint restart is performed, the SWA is again updated with information that is saved in the last valid checkpoint entry on the checkpoint data set.

When the information is merged and step restart occurs, the SWA appears the same as it did before the original execution of the restart step. If checkpoint restart occurs, the SWA differs from its original form in the following ways:

- Except for data sets not opened during the original execution that requested nonspecific tape volumes, data sets specified as NEW in the restart step have their dispositions changed to OLD.

- In the case of data sets for which nonspecific volumes are requested in the restart step, the work queue entry describes the device type and serial numbers of the volumes assigned to the data sets during the original execution.

- In the case of multivolume data sets, the work queue entry indicates which volumes are processed at the checkpoint. These volumes, and not the first volumes of the data sets, are mounted (if they have not remained mounted) during the restart. For VSAM, however, volume mounting is based on the present catalog information and may result in the mounting of unneeded volumes; for example, the first volume of a sequential data set may be temporarily mounted although it is no longer required at the checkpoint being restarted and may be demounted almost immediately. This situation may be avoided through specification of the parallel mount subparameter of the DD statement UNIT parameter (UNIT=(,P). For further information, see the publication *JCL.*

- The system-generated name for any temporary data set uses the time stamp obtained when the job was originally interpreted.

In addition, any modification made to the job's environment by the use of the dynamic allocation facilities prior to the last checkpoint reflects in the re-created SWA. The SWA appears as it did at the time of checkpoint.

How a Step Restart Is Initiated

A step restart is initiated in the same way as it is during a normal execution. The devices allocated to the restart step can be different (but of the same device type) from the devices allocated originally. If the allocated devices differ, volumes must be moved from one device to another. If automatic volume recognition (AVR) is used and if the devices are available for allocation, devices containing the required volumes are allocated.

Unless the volumes are already mounted, normal mounting messages request the operator to mount the required volumes on the devices after devices are allocated to the restart step. The requested volumes resume processing.

Deferred Restart

Message Sequence

To perform a deferred checkpoint restart, the job to be restarted is resubmitted in an input job stream. Messages that contain checkpoint entry identifications are displayed on the console during the original execution of the job and thus may be used by the programmer preparing the job for resubmission. When the resubmitted job is restarted, messages appear on the console in the following sequence:

1. A message asking if the checkpoint data set (used for restart) is a secure volume.

2. If additional checkpoint data sets (other than the data set used for restart) are encountered at restart time, checkpoint data set security messages are issued.

3. A message indicating the virtual-storage requirements of the job.

4. Normal mount messages.

5. If password-protected data sets are to be repositioned at restart time, a password message is issued.

6. A successful restart message.

To perform a deferred step restart, the job to be restarted is resubmitted. Normal mount messages are displayed. A message indicating the file protection status of the checkpoint/restart volume is not displayed.

Operator Considerations

When a job is resubmitted to perform a deferred checkpoint restart (the RESTART parameter is coded on the JOB statement with a checkid operand), the processing is essentially the same as that during an automatic checkpoint restart after the restart reader has reinterpreted the job. A message is issued to the operator, indicating the virtual-storage requirements of the job.

The required virtual-storage area can also be unavailable for the following other reasons:

- The REGION size parameter for the step is larger when the job is resubmitted than in the original execution, and the area required is at different addresses than the area available.

- A new IPL is performed and, because of different IPL options specified by the operator, the area required is larger than the area available.

If these conditions exist, a message is displayed, indicating that virtual storage for the job step to be restarted is unavailable.

Deferred Step Restart

You cause a deferred step restart of a job by coding the RESTART parameter on the JOB statement and resubmitting the job. The parameter specifies a job step or a step of a cataloged procedure. It restarts the job at the beginning of the specified step. Steps preceding the restart step are interpreted, but not initiated.

The CHKPT macro instruction may or may not be coded in your program. Figure 11 illustrates a job originally submitted and the same job resubmitted for step restart. Assume that the results of STEP2 were unsatisfactory because of abnormal termination or incorrect data when the job was executed originally.

```
             Original Deck

//MYJOB   JOB      MSGLEVEL=1¹          No automatic restart
//*                                      requested
//STEP1   EXEC
           .
           .
           .
//STEP2   EXEC     PGM=MYPROG
           .
           .
           .
//STEP3   EXEC

             Resubmitted Deck

//MYJOB   JOB      MSGLEVEL=1,¹         Causes restart of job
//                 RESTART=STEP2         at STEP2
//STEP1   EXEC
           .
           .
           .
//STEP2   EXEC     PGM=MYPROG
           .
           .
           .
//STEP3   EXEC
```

Figure 11. Requesting a Deferred Step Restart

Note to Figure 11:

1 MSGLEVEL=1 is optional.

JCL Requirements and Restrictions on Deferred Step Restart

To perform a deferred step restart, you must provide the data set environment required by the restart job. This may be accomplished by using the conditional disposition subparameter in the appropriate DD statements during the original execution of the job. Conditional dispositions in the original deck must be used to:

- Delete all NEW data sets used by the step to be restarted.
- Catalog all data sets passed from steps preceding the restart step, to the restart step, or to steps following the restart step. Abnormal termination of the restart step, when it is originally run, causes cataloging of the passed data sets. The information is available to the following steps when the deck is resubmitted.
- Keep all OLD data sets used by the restart step, other than those passed to the step.

If a MOD data set on tape is used in the restart step, the data set is not repositioned at the start of the restart execution, and data written into it during the restart execution follows the data written during the original execution. You may want to reposition the data set so the data written during the restart execution overlays the data written during the original execution.

Any data sets that are dynamically deallocated have the disposition specified at the time deallocation occurred. Conditional disposition processing is not done during abend.

The following rules apply to the restart deck:

1. Code the RESTART parameter on the JOB statement.
2. If data sets are passed from steps preceding the restart step to the restart step or to steps following the restart step, the DD statements receiving the data sets must entirely define the data sets. They must explicitly specify volume serial number, device type, data set sequence number, and label type, unless this information can be retrieved from the catalog. It is recommended that passed data sets be conditionally cataloged during abnormal termination of the original execution. Label type cannot be retrieved from the catalog.
3. Generation data sets created and cataloged in steps preceding the restart step must not be referred to in the restart step or in steps following the restart step by the relative generation numbers used to create them. They must be referred to by their actual relative generation numbers. For example, a data set created as the +1 data set must be referred to as the 0 data set (assuming that the +2 data set was not also created).
4. The EXEC statement PGM and COND parameters and the DD statement VOL=REF parameter must not be used in the restart step or in steps following the restart step if they contain values of the form *stepname* or *stepname.procstepname*, referring to a step preceding the restart step.
5. The DD statement VOL=REF parameter is ignored if restart is attempted from a checkpoint taken when the DD statements are opened out of order and the referenced DD statement requested nonspecific tape volumes.

Resource Variations Allowed in Deferred Step Restart

A deferred step restart only allows the restarted execution of a job to begin at other than the first step of the job. Job step initiation and allocation of resources are accomplished normally. The following variations are allowed upon restart:

- Variation of device and volume configuration
- Variation in JCL and data in the resubmitted deck

Deferred Checkpoint Restart

You cause a deferred checkpoint restart of a job by the following procedure:

1. Use the option of coding a special form of the RD parameter (RD=NR) in the original job deck. This specifies that, if the CHKPT macro instruction executes, a checkpoint entry is written but an automatic checkpoint restart is not requested.

2. Cause execution of the CHKPT macro instruction, which writes checkpoint entry.

3. Resubmit the job whether or not it terminated abnormally. For example, resubmit it because a volume of one of its input data sets was in error and had caused the corresponding part of an output data set to be in error.

4. Code the RESTART parameter (**RESTART=(***stepname***,***checkid***)**) on the JOB statement of the restart deck. The parameter specifies both the step to be restarted and the *checkid* that identifies the checkpoint entry to perform the restart.

5. Place a SYSCHK DD statement immediately before the first EXEC statement in the restart deck. It specifies the checkpoint data set from which the specified checkpoint entry is read and is additional to any DD statements in the deck that define data sets into which checkpoint entries are written. Figure 12 on page 43 illustrates a job when it is originally submitted and when it is resubmitted for a deferred checkpoint restart. Assume in Figure 12 that STEP2, when originally executed, terminates abnormally at some time after CH04 is written. Note that, in the resubmitted deck, the programmer requests that STEP2 be restarted using the checkpoint entry identified as entry CH04.

```
          Original Deck

//MYJOB   JOB      RD=NR                    Requests that
//*                                         automatic restart not
//*                                         occur (optional)
//STEP1   EXEC
                     .
                     .
                     .
//STEP2   EXEC     PGM=MYPROG               MYPROG issues CHKPT
//*                                         macro
//*
//NAME1   DD       DSN=NAME2,DISP=(,CATLG,KEEP)
//*                                         Describes checkpoint
//*                                         data set
                     .
                     .
                     .
//STEP3   EXEC

          Resubmitted Deck

//MYJOB   JOB      RESTART=(STEP2,CH04)     Request restart at
//*                                         CH04 in STEP2
//SYSCHK DD        DSN=NAME2                Describes data set
//*                                         that contains CH04
//STEP1   EXEC
                     .
                     .
                     .
//STEP2   EXEC     PGM=MYPROG
//NAME1   DD       DSN=NAME2,DISP=(,CATLG,KEEP)
//*                                         Describes data set
//*                                         in which new
//*                                         checkpoint entries
//*                                         will be written
                     .
                     .
                     .
//STEP3   EXEC
```

Figure 12. Requesting a Deferred Checkpoint Restart

JCL Requirements and Restrictions on Deferred Checkpoint Restart

To perform a deferred checkpoint restart, you must provide the data set environment required by the restart job, using conditional dispositions during the original execution.

Conditional dispositions should be used to:

- Keep all data sets used by the restart step.
- Catalog all data sets passed from steps preceding the restart step to steps following the restart step. Although the step that terminates abnormally is not using the passed data sets, its termination causes the cataloging of the data sets if the conditional catalog parameter is used in the preceding steps.

Notes:

1. *Temporary data sets cannot be kept.*
2. *If the DD statement for any data set at checkpoint time had DISP=(,DELETE,CATLG), the data set must be uncataloged after completion of the restarted job or job step by using IEHPROGM.*

Dynamically deallocated data sets have the disposition processed as specified at the time of deallocation. Conditional disposition processing is not performed during abend.

The following rules must be adhered to when resubmitting a job for a deferred checkpoint restart:

1. A RESTART parameter with a *checkid* subparameter must be coded on the JOB statement.
2. A SYSCHK DD statement must be placed in the job deck immediately before the first EXEC statement.
3. The EXEC statements in the job deck must have unique names. (The system searches for the named restart step.)
4. The JCL statements and data in steps preceding or following the restart step can be different from their original forms. However, all backward references must be resolvable.
5. The restart step must have a DD statement corresponding to each DD statement present in the step in the original deck, and the names of the statements must be the same as they were originally. However, the restart step can contain, in any position, more DD statements than it contained originally. The total number of volumes specified at restart must equal or exceed the number specified at the checkpoint.
6. If a DD statement in the restart step in the original deck defined a data set that was open at the checkpoint to be used, the corresponding statement in the restart deck must refer to the same data set, and the data set must be on the same volume, and, in general, have the same extents recorded in its DSCB as it did originally. (See the exceptions in the note that follows.) If the data set is

multivolume and is processed by the sequential access method (SAM), only the part of the data set on the volume in use at the checkpoint needs to be the same as it was originally.

Note: The extents can differ as follows:

- In the DD statement, you can request that additional space be allocated to the data set when the space currently available is exhausted. If space is allocated after a checkpoint is taken, this space is indicated in the DSCB; after restart from the checkpoint, the space is released, and the DSCB contents are changed to what they were at the checkpoint.

- In the DD statement, you can request that unused space be released at the end of the job step. If the space is released, the DSCB may indicate a reduced extent for the data set when deferred restart at a checkpoint occurs; no space is allocated to replace released space. Space is not released when step termination is followed by automatic restart.

JCL-specified data sets that were deallocated prior to the checkpoint are not allocated at restart.

When there is no need to read or modify a data set after restart, the data set can be replaced by a dummy data set if the original data set was processed by SAM and the job step is not restarted from a checkpoint within the data set's end-of-volume exit routine. A VSAM data set using the ISAM compatibility interface cannot be replaced by a dummy data set. Any dummy data set present at the time of the checkpoint must be present as a dummy data set at restart. Allocation is done for each DD statement in the job step where the checkpoint is taken, even if the data set is closed at the time of the checkpoint.

7. The data in the restart step need not be the same as it was originally. If data following a DD * statement was present originally and is entirely omitted in the restart deck, the delimiter (/*) statement following the data may also be omitted. The delimiter statement following a DD DATA statement may not be omitted.

8. The VOL parameter of a DD statement must refer to at least those volumes it refers to at checkpoint time. More volumes may be added if desired. The following DD statement parameters may not be changed: DISP, DCB, LABEL, and UNIT.

9. Except for the requirements stated in rules 4 through 8, the JCL statements and data in the restart step can be different from their original forms. In particular, the DUMMY parameter can be used for any data that was not open at the checkpoint.

10. If data sets are passed from steps preceding the restart step to steps following it, the DD statements receiving the data sets must entirely define them. They must explicitly specify volume serial number, device type, data set sequence number, and label type, unless this information can be retrieved from the catalog. It is recommended that passed data sets be conditionally cataloged during abnormal termination of the original execution. Label type cannot be retrieved from the catalog.

11. The EXEC statement PGM and COND parameters and the DD statement VOL=REF parameter must not be used in the restart step or in steps following the restart step if they contain values of the form *stepname*, or *stepname.procstepname*, referring to a step preceding the restart step.

12. The DD statement VOL=REF parameter is ignored if restart is attempted from a checkpoint taken when the DD statements are opened out of sequence and the referenced DD statement requested nonspecific tape volumes.

13. The volume serial number must be coded on a restart DD for a generation data group if the deferred checkpoint restart volume serial number list is to be different from the original.

Resource Variations Allowed in a Deferred Checkpoint Restart

The system's device and volume configuration can differ from the status during the original execution of the job. The allowable differences are those described under "Resource Variations Allowed in Automatic Restart" on page 36.

System Restrictions for Deferred Checkpoint Restart

The following restrictions apply when submitting a job for deferred checkpoint restart:

- Jobs specifying nonpageable storage require an area of storage identical to the storage originally requested at the time the checkpoint is taken and starting at the same address.

- The link pack area modules in use at the time the checkpoint is taken must reside in the same storage locations for the job submitted for deferred checkpoint restart.

- The nucleus must not be changed between checkpoint and restart.

If the required storage is not available, it may be for one of the following reasons:

- The link pack area expands into the required storage.

- The system queue area expands into the required storage. If either of the above conditions occurs, contact your system programmer for a respecification of the system parameter to modify the area size and repeat initial program loading using the new value.

How the System Works during a Deferred Checkpoint Restart

After the system reads and interprets the restart deck, it reads the specified checkpoint entry and merges information from it into the scheduler work area entry for the job. As a result, the work queue entry differs from the entry existing during the original execution, as described earlier. (See "How the Job Deck Is Reinterpreted and the SWA Merged" on page 37.)

Next, the system initiates the restart step normally. The system reads the specified checkpoint entry again and functions as in automatic restart. Restart is delayed until the required virtual-storage area is available.

When a job restarts correctly, you receive two messages: IHJ006I and IHJ008I. If these messages do not appear for nonpageable storage jobs, enter a DISPLAY A command to see if a system task or another job is using the required storage. You can then stop or cancel the conflicting job.

If you have multiple tape or disk volumes, the system may ask the operator to mount data volumes other than those required at the beginning of the job. In addition, any card input data sets that are used by the failing job step must again be made available to the system.

Coding the RD and RESTART Parameters

RD (Restart Definition) Parameter

The RD parameter is coded in the JOB or EXEC statement to request automatic step restart if failure occurs, and/or to suppress, partially or totally, the action of the CHKPT macro instruction. If the RD parameter requests an automatic step restart if failure occurs, or if the RD parameter is not coded, the action of CHKPT is normal. (CHKPT writes a checkpoint entry and requests a checkpoint restart to be performed if failure occurs.) The RD parameter is ignored for system tasks and generalized start jobs.

The RD parameter, coded on an EXEC statement, applies to the step corresponding to the statement or to all steps of the cataloged procedure referred to by the statement. Coded on a JOB statement, the RD parameter applies to all steps of the corresponding job and overrides an RD parameter coded in any EXEC statement of the job. The syntax of the parameter is:

RD[.*procstepname*]={R | NC | NR | RNC}

The possible definitions are:

RD=R
(Restart) Requests an automatic step restart if failure occurs. If the CHKPT macro instruction is executed in the step, the resulting request for an automatic checkpoint restart overrides the request for an automatic step restart. This parameter is ignored if the job does not contain a job journal. For JES2 and JES3, this parameter forces job journaling.

RD=NC
(No checkpoint) Does not request an automatic step restart. It totally suppresses the action of the CHKPT macro instruction if the macro instruction executes in the step. This allows use of a program containing

CHKPT when the checkpoint function is not wanted. It can also suppress the checkpoint at end-of-volume.

RD=NR
(No automatic restart) Does not request an automatic step restart. It suppresses the request for an automatic checkpoint restart that is made when the CHKPT macro instruction executes in the step. If CHKPT executes, it writes a checkpoint entry normally. A deferred restart can be performed from the checkpoint entry.

RD=RNC
(Restart and no checkpoint) Requests an automatic step restart if failure occurs. It totally suppresses the action of CHKPT if CHKPT executes in the step. It can suppress the checkpoint at end-of-volume. If the job does not contain a job journal, the step is ineligible for automatic restart. For JES2 and JES3, this parameter forces job journaling.

If **RD=***value* is coded on an EXEC statement that invokes a cataloged procedure, the parameter applies to all steps of the procedure and overrides all RD parameters present in the EXEC statements of the procedure. **RD.***procstepname***=***value* can be coded instead of **RD=***value*; it applies to the specified procedure step and overrides the RD parameter that may be coded on the EXEC statement of the procedure step. **RD.***procstepname***=***value* can be coded once for each step of the procedure.

RESTART Parameter

The RESTART parameter requests a deferred restart of a job. It is coded in the JOB statement when the job is resubmitted. If step restart is to occur, this parameter specifies which step to begin. If the restart is to occur at a checkpoint that was taken during a step, both the step and the identification of the particular checkpoint entry are specified. The syntax of the parameter is:

RESTART=({*stepname* | *stepname.procstepname* | ***}[,***checkid***])**

Both operands are used if restart at a checkpoint is to occur. If a step restart is to occur, *checkid* must be omitted; the enclosing parentheses may be omitted.

stepname
is coded as *stepname.procstepname* if a step of a cataloged procedure is to be restarted. The parameter can be coded as * if the first step of the job (possibly a step of a cataloged procedure) is to be restarted.

checkid
can contain up to 16 characters in any combination of alphameric characters, printable special characters, and blanks. If it contains any special characters or blanks, it must be enclosed in single apostrophes, and apostrophes within it must be represented as double apostrophes.

SYSCHK DD Statement

The SYSCHK DD statement must be included in the resubmitted job to perform a deferred checkpoint restart. This DD statement specifies the checkpoint data set that contains the checkpoint entry to be used in the restart. The SYSCHK DD statement may not be included when a deferred step restart is to be performed.

The statement must precede the first EXEC statement in the deck that performs a deferred restart at checkpoint. It must follow the JOBLIB DD statement if the JOBLIB DD is present. The desired checkpoint entry must be named by the *checkid* subparameter of the JOB statement RESTART parameter.

The following requirements and restrictions apply to the SYSCHK DD statement:

- The statement must contain or imply DISP=(OLD, KEEP).
- The statement must define the checkpoint data set. It must specify its name, device type, and volume serial number. The catalog may be used, eliminating the need for device type and volume serial number.
- If the volume containing the checkpoint data set is mounted on a JES3-managed device, the SYSCHK DD statement must not request deferred mounting.
- The SYSCHK data set cannot be multivolume or concatenated. If the checkpoint data set is multivolume, the SYSCHK DD statement must specify, as the first volume of the data set, the volume and data set name that contain the desired checkpoint entry. The serial number of the volume containing a particular entry appears in the console message that is written when the entry is written. A checkpoint data set may not be a concatenated data set.
- If the checkpoint data set is partitioned, the DSNAME parameter on the SYSCHK DD statement must not contain a member name.
- If a RESTART parameter without the *checkid* subparameter is included in a job, a SYSCHK DD statement must not appear before the first EXEC statement of the job.
- If a RESTART parameter is not included in a job, a SYSCHK DD statement appearing before the first EXEC statement in the job is ignored.
- A SYSCHK DD statement appearing in a step or procedure step of a job is treated as an ordinary DD statement; that is, the name SYSCHK has no special meaning in that case.

An example of a SYSCHK DD statement is:

```
//SYSCHK DD   DSN=dsname,DISP=OLD,UNIT=name,
//            VOL=SER=volser
```

Chapter 4. Checkpoint/Restart Processing

This chapter describes how the checkpoint/restart routine handles different types of user data sets, and gives additional guidelines on repositioning and preserving the contents of data sets. There are also some notes at the end of the chapter on using checkpoint/restart with various programs and programming languages.

Notes for User Data Sets

The data set types discussed are listed below:

- BDAM data sets
- Generation data sets
- ISAM data sets
- MSS data sets
- Partitioned data sets
- SAM data sets
- SYSABEND data sets
- SYSIN data sets
- SYSOUT data sets
- TCAM data sets
- Temporary data sets
- VSAM data sets
- Work data sets

BDAM Data Sets

When the basic direct access method (BDAM) is used to process a data set, processing resumes normally upon restart. However, you must ensure that a particular block is read or written before a checkpoint is taken. Your program must complete the BDAM I/O operation by executing the CHECK or WAIT macro before it executes the CHKPT macro.

Any change made to a record between a checkpoint and a restart will remain in the data set. You are responsible for backing out any changes that would otherwise produce invalid results. For more information, see "Preserving Data Set Contents" on page 63. If the program does not complete the operation, the block may be read or written either before or after the checkpoint is taken.

For serially reusable resources, your program must also issue either the WRITE or RELEX macro to release a record that is read with exclusive control, before executing the CHKPT macro.

Generation Data Sets

No automatic cataloging of generation data sets takes place. If certain generation data sets of a generation data group are to be cataloged, you must catalog them. The order in which they are cataloged determines the relative generation numbers of the generation data sets for reference by later jobs. The last generation data set becomes the 0 generation, the next-to-last cataloged generation data set becomes the -1 generation, and so on.

Generation data sets created by one step of a job may be passed to subsequent steps in the same job and may be referenced by the relative generation numbers assigned at the time of creation, whether or not the generations were cataloged.

For deferred step restart, the generation data group name table (GDGNT) is recreated from the catalog. The last generation data set cataloged prior to termination of the job becomes the 0 generation and is used for the base name in the GDGNT. This may not be the same as the base name when the job was initially run; you must know which generation data sets were cataloged and in what order the data sets were cataloged, and the JCL must be modified accordingly.

For a deferred checkpoint restart, no modification of the relative generation number in the JCL is necessary.

When a job is started, the base name (0 generation name) of each generation data group is placed in the GDGNT. The GDGNT is never changed unless a new generation level is cataloged. It is saved at a checkpoint and is available for both automatic and deferred checkpoint restart. The restart takes place without any change in the JCL, whether or not generations previously created by the job were cataloged.

When using Generation Data Groups, the disposition of the data set should be specified correctly to avoid having to change the JCL at restart time. If the DISP=(NEW,CATLG,CATLG) was specified, then the Generation Data Group number is set to (0) at the time of restart and the checkpointed data set is now at GDG(-1), regardless of how the job completed. In order for generation data sets to maintain their respective generation numbers at checkpoint and restart time, the disposition should specify DISP=(NEW,CATLG,KEEP). This will prevent the generation level from changing if the job step fails. If, however, you wish to restart a job that completed without failing, you must change the JCL of the job to be resubmitted to get the correct generation levels.

You must take precautions when using the generation data set as the checkpoint data set. For a deferred checkpoint restart, the checkpoint data set, as specified in the SYSCHK DD statement, is allocated to the initiator, and an entry is made in the initiator's GDGNT. The GDGNT is never updated and remains for the duration of the job and for any future restarts under the same initiator. The GDGNT uses a data set from the same generation data group and uses the original contents of the GDGNT to obtain the qualified name of the generation data set.

After entry into the GDGNT, if the levels of the generation data group change in the catalog and if the checkpoints are of a different generation, the desired generation at restart time may not be the one retrieved as the checkpoint data set. To avoid this problem, do not make changes to the number of existing generations for the duration of the IPL after the checkpoint data set is used for deferred restart. Be sure the same initiator is not used twice to do restarts from different levels of the same generation data group.

Note: A job that contains generation data sets referred to by a relative generation number of +1 or greater and with a disposition of OLD, SHR, or MOD is failed by the JES3 interpreter service unless the UNIT parameter is included in the DD statement. With UNIT specified, the JES3 interpreter service permits this data set to be allocated on a deferred basis.

ISAM Data Sets

A checkpoint should not be taken before an ISAM data set is opened in load mode. A checkpoint should be taken immediately after the data set is opened. Otherwise, an abend results from a restart at a previous checkpoint.

A restart should not be attempted from checkpoints taken during loading of an ISAM data set using QISAM in load mode, if insertions were made on the ISAM data set after it was loaded, and if the insertions were made using the WRITE KN macro.

An ISAM data set that is shared must be closed before taking a checkpoint. Note that, if an ISAM data set is closed immediately after restarting the program at a checkpoint, the data set may not be restored to its original condition.

Any change made to a record between a checkpoint and a restart will remain in the data set. You are responsible for backing out any changes that would otherwise produce invalid results. For more information, see "Preserving Data Set Contents" on page 63.

MSS (Mass Storage System) Data Sets

Restart delays one second for each MSS volume that must be mounted, plus the time required to stage cylinder zero, the VTOC, and each data set.

Partitioned Data Sets

If a partitioned data set is compressed after a checkpoint in which the partitioned data set was open, restart at checkpoint should not be requested.

If a checkpoint is taken and a partitioned data set is opened, another checkpoint should be taken before any records are written into the data set. If the second checkpoint is not taken and restart occurs at the first checkpoint, the OPEN routine positions to the current end of the data set instead of to the original end.

Adding Members

When a partitioned data set is updated, be careful to preserve the contents of the directory. The directory consists of entries that point to each member of the data set.

When a member is added to a partitioned data set, an entry is also added to the directory. If one member is added, the STOW macro may be used to create the entry, or the member name may be specified in the DD statement. In the latter case, the control program creates the directory entry when the data set is closed or when the job step terminates. If more than one member is added, the STOW macro must be used to create an entry for each member, and a new checkpoint should be taken after each use of the STOW macro.

When one or more members are added to a partitioned data set, a checkpoint should be taken immediately after opening the data set. After taking the checkpoint, the new member may be written and its entry added to the directory. If the step is restarted from the checkpoint, the data set is then repositioned; the new member and its directory entry are deleted and are re-created after restart.

Updating Members

To update a member of a partitioned data set, updated records may either be written back to their original locations, or the entire member (in updated form) may be rewritten as a new member of the data set. In the latter case, the directory entry must be updated to point to the rewritten member.

If a checkpoint is taken before rewriting an entire member, one must also be taken immediately after updating the directory, because the control program deletes the updated directory entry if it repositions the data set for restart from the original checkpoint. Because no entry then points to the original member, the postrestart processing will be unsuccessful.

Deleting Members

Members may be deleted from a partitioned data set during a restart. Because this action may delete members written by another job (another job may have been executed between the original and restart executions of the subject job), restart at a checkpoint should not be requested.

If a partitioned data set is open for output at checkpoint time, any new members added later will be deleted during a restart from this checkpoint. **No members will be deleted when a partitioned data set is open for update**. You are expected to be cautious when restarting from checkpoints having partitioned data sets open for update.

SAM Data Sets

When BSAM or QSAM is used to read a data set from a card reader, your program can reposition the data set upon restart. If you provide a repositioning routine, instruct the operator to position the data set to the beginning if a restart becomes necessary. The program might be designed to operate as follows:

- The program saves the first record read from the data set and keeps a count of the number of records read before each checkpoint.
- After a restart, the repositioning routine reads a record from the data set and compares it with the first record read before abnormal termination. If the records are identical, the data set has been positioned to the beginning. The routine then repositions it by reading (without otherwise processing) the number of records read before the checkpoint.

SYSABEND Data Sets

Whether or not checkpoint/restart is used, abnormal termination causes the system to write a SYSABEND (or SYSUDUMP) data set if you provide a SYSABEND (or SYSUDUMP) DD statement. The system uses its own data control block to write the data set, and it opens the data set during abnormal termination processing. You may code or omit the SYSOUT parameter on the SYSABEND DD statement.

When the SYSOUT parameter is coded and automatic restart occurs after abnormal termination, the SYSABEND or SYSUDUMP data set is not printed for step restart. Because the SYSABEND or SYSUDUMP data set is created by the job step, it is deleted during restart.

In all other cases, the SYSABEND or SYSUDUMP data set is printed, whether or not the restart is successful. If a second abnormal termination occurs, a second SYSABEND or SYSUDUMP data set is written. The same rules that apply to the first data sets also apply to the second data set and to all subsequent data sets.

SYSIN Data Sets

When restart at a checkpoint occurs, a SYSIN data set (data following a DD * or DD DATA statement) is repositioned. Unit-record data sets are never repositioned. (The checkpoint routine waits until all requested input/output operations are complete, then requests that the job entry subsystem save positioning information.) When automatic restart occurs, the system keeps the direct access data sets that contain the SYSIN data of the job restarting. During the restart execution, the job can read data from the direct access data sets as it could during the original execution.

To perform deferred restart, you include any necessary SYSIN data in the resubmitted deck. If the restart is to be at a checkpoint, and a SYSIN data set is open and not completely read at the checkpoint to be used, the attributes of the direct access data set (into which the system will write the SYSIN data) must be the same as the attributes of the direct access data set used originally. (The location and number of extents in the data set used during restart need not be the same.)

Information about altering SYSIN data in a restart deck is given under "JCL Requirements and Restrictions on Automatic Restart" on page 36.

SYSOUT Data Sets

The following describes how SYSOUT data sets (data sets having the SYSOUT parameter coded on their DD statements) are handled during various types of restart.

Automatic Restart: Your program writes SYSOUT data into one or more direct access data sets. If step restart is occurring, the direct access data sets used during the original execution are deleted. New direct access data sets are allocated when the step restarts.

If a checkpoint restart occurs, the data sets used during the original execution are kept. If a SYSOUT data set was open when the last checkpoint was taken, it is repositioned to its position at the time the checkpoint was taken.

The checkpoint routine waits until all requested input/output operations are complete, then requests that the job entry subsystem save positioning information. Data written during the restart execution overlays only the data written between the time the last checkpoint was taken and the time the job step terminated abnormally. If a SYSOUT data set is closed at the checkpoint, the data set is not repositioned. If the restart step opens the same data set again, the data written during the restart follows the data originally written. (The data set has an implied disposition of MOD.)

Deferred Checkpoint Restart: When a checkpoint restart occurs, and a SYSOUT data set is open at the checkpoint, the data set written into during the restart is a new data set different from the data set used originally.

TCAM Data Sets

A successful restart of a telecommunications access method (TCAM) data set depends on the following conditions:

- The message control program (MCP) region must be active and have enough virtual storage to build the required control blocks.
- The QNAME parameter in the DD statement of the checkpoint job must be available in the terminal table of the MCP region.

Temporary Data Sets

Direct-access space for temporary data sets can be preallocated to save time in scheduling job steps but cannot be used with checkpoint/restart. Checkpoints and automatic restarts are suppressed for any job step that uses a preallocated temporary data set.

VSAM Data Sets

General Information

For VSAM data sets, you are responsible for handling checkpoint/restart problems that arise because of changes in the data. For example, consider a program that updates records in a data set by adding a number to a value already existing in some field within the record. If the program terminates and is restarted, you must ensure that the records processed between the checkpoint and the termination are not processed again after the restart.

During checkpoint restart processing, no user ACB exits are taken. For checkpoint processing, the AMB exception exit is taken; for restart processing, the AMB exception exit is not taken.

The checkpoint program issues a VSAM temporary CLOSE macro instruction to update the catalog. It records information about VSAM data sets in a checkpoint data set. If a failure occurs, the latest checkpoint record is used to reconstruct the situation that prevailed when the checkpoint was taken.

When the checkpoint routine records positioning for a VSAM data set, all outstanding I/O requests for the data and index are completed before the contents of your address space are saved. If an error occurs while these I/O requests are processed, the checkpoint procedure stops and a code of X'0C' is returned in register 15. You may handle the error condition and reissue the CHKPT macro.

A checkpoint cannot be taken if VSAM data sets are open for regions that are using the control blocks in common (CBIC) option.

The AMP Parameter

The AMP parameter has a subparameter for specifying checkpoint/restart options that handle two special situations in restarting a processing program:

- Modifications to the data set other than records added sequentially to the end of an entry-sequenced data set. The restart program cannot restore a data set to its checkpoint status if there have been internal modifications to it since the checkpoint, so the restart program does not attempt restart processing.

- Addition of records to the end of a data set by way of a job step other than the job step that issued the checkpoint. Any records added to the end of an entry-sequenced data set are erased in restoring the data set to its checkpoint status.

The AMP options for checkpoint/restart are:

- Let restart take its normal action for either situation.

- Override either one or the other of the two actions.

- Override both.

If you override the check for internal modification, the processing program is restarted, even though the data set being processed cannot be restored. If you override erasing data at the end of a data set and the catalog has been updated, the processing program is not restarted unless you also override the check for modification. For more detail on multiple ACBs open against the same data set, see "ACB Macro" in *VSAM Administration: Macro Instruction Reference.*

To prevent data from being erased or to allow restart with modified data sets, the AMP subparameter must be coded in the DD statement for the cluster or data set.

All multiple ACBs open for output connected to the same control block structure must have identical checkpoint restart AMP CROPS options. If this is not done, the results are unpredictable.

Repositioning is mandatory for all VSAM data sets open for create mode processing, except for relative record data sets processed in direct mode. If AMP='CROPS=NRE' or AMP='CROPS=NRC', no checkpoint is taken. If a checkpoint is attempted, message IHJ000I is issued, and a return code of X'08' is returned in register 15. A reason code of 41 in register 0 is also returned to you.

If a VSAM data set supported for repositioning extends to a new volume after the checkpoint, VSAM restart cannot reposition the data set. A restart from that checkpoint is not successful unless the no-reposition option is taken by specifying AMP='CROPS=NRE' or AMP='CROPS=NRC'.

VSAM Data Set Types

VSAM has key-sequenced, entry-sequenced, and relative record data sets. The main difference among the three is the order in which the data records are loaded into the data sets.

Records are loaded into a *key-sequenced data set* (KSDS) in key sequence, the order defined by the collating sequence of the contents of the key field in each of the records. Each record has a unique value in the key field, such as employee number or invoice number. VSAM uses an index and optional free space to insert a new record into the data set in key sequence.

Records are loaded into an *entry-sequenced data set* regardless of the contents of the records. Their sequence is determined by the order in which they are physically arranged in the data set or their entry sequence. New records are stored at the end of the data set.

Records are loaded into a *relative record data set* in relative record number sequence. The data set is a string of fixed-length slots, identified by a relative record number. When a record is inserted, you can assign the relative record number or allow VSAM to assign the record the next available number in sequence. No index is used.

Key-Sequenced Data Sets

A key-sequenced data set is prepared for restart by the restoration of any statistical information (such as number of records inserted) to its checkpoint status.

For a VSAM key-sequenced data set, restart does not erase any data except in create mode. It does, however, detect modification of the data set by either the checkpointed program or another program that used the data set between the checkpoint and the restart. If the data set is modified, the restart is terminated, unless you override the testing of the data set by using the AMP='CROPS=NCK' subparameter in the DD statement for the data set.

Entry-Sequenced Data Sets

A checkpoint may not be taken if a VSAM entry-sequenced data set is open for output with an immediate-upgrade path (or alternate index) open over it, unless the no-reposition option, AMP='CROPS=NRE' or AMP='CROPS=NRC', is specified. VSAM immediate-upgrade data sets are key-sequenced data sets, and repositioning is not supported for them.

For a VSAM entry-sequenced output data set, all data added after the last checkpoint was taken is physically erased unless the AMP='CROPS=NRE' or NRC subparameter is specified in the DD statement for the data set. If data is erased, the catalog record for the data set is updated to reflect the current end of data, and the data-set statistics are adjusted to reflect the new status.

During restart, entry-sequenced output data sets are restored by the elimination of all records added at the end since the checkpoint.

Relative Record Data Sets

A checkpoint may not be taken for a VSAM relative record data set in load mode if direct processing was or is performed.

If a checkpoint is taken with a relative record data set open for load mode nondirect processing and direct processing is performed after the checkpoint (thus affecting data that was loaded before the checkpoint was taken), no attempt is made by restart to reset the data that existed at checkpoint time. After restart, the data set resets to checkpoint status but still contains the results of any direct processing performed on that part of the data set that existed at checkpoint time.

Work Data Sets

Many programs use "work" data sets, which are alternately written and read, rewritten and reread. If a work data set is used, a checkpoint should be taken each time you have finished reading the data set and before rewriting it.

For example, a program may perform the following sequence of operations to produce different versions of data set A:

1. Write and then read back A1.

2. Write and then read back A2.

3. Write and then read back A3.

A checkpoint should be taken at the beginning of operations 2 and 3 before any rewriting of data set A takes place. If, for example, the job step is abnormally terminated while operation 2 is in progress, the job step can be restarted from the checkpoint taken at the beginning of operation 2. At this checkpoint, there is no need for the data in version A1.

Notes for Tape Labels and Files

The tape files and labels discussed are listed below:

- DOS tape files

- ISO/ANSI tape labels

- Nonstandard tape labels

DOS Tape Files

Checkpoints may be taken with DOS tape files opened with the bypass leading tapemark option LABEL=(,LTM) and/or the bypass embedded DOS checkpoint records option DCB=(OPTCD=H) specified. However, a checkpoint must not be taken when an opened data set resides on a DOS 7-track tape, is written in translate mode, and contains embedded checkpoint records.

ISO/ANSI Tape Labels

An ISO/ANSI tape data set may not be used as a checkpoint data set.

If an ISO/ANSI tape data set is open when a checkpoint is requested, the checkpoint is refused. Message IHJ000I is issued and a return code of X'08' is returned in register 15, and a reason code of 102 is returned in register 0.

Nonstandard Tape Labels

You must provide a routine to process nonstandard labels at restart time. This routine must perform input header label processing, because output tapes contain the header labels that were written when the data sets were opened (prior to checkpoint).

At restart time, the control program checks the tape to make sure that the first record is not a nonstandard volume label. If the first record is 80 bytes in length and contains the identifier VOL1 in the first 4 bytes, the label is not an IBM standard label and the tape is not accepted. Restart issues a message directing the operator to mount the correct tape.

If the tape does not contain an IBM standard volume label, restart's routine gives control to the user's routine for processing nonstandard labels. When this routine receives control, the tape is positioned at the interrecord gap preceding the nonstandard label (the tape has been rewound).

If your routine determines that the wrong volume is mounted, a 1 must be placed in the high-order bit position of the SRTEDMCT field of the unit control block (UCB), and control is returned to restart. The control program issues a message directing the operator to mount the correct volume. When the new volume is mounted, restart repeats the above steps.

Before returning control to restart, your routine must position the tape at the interrecord gap that precedes the initial record of the appropriate data set. This applies to both forward and backward read operations. The control program then uses the block count shown in the data control block to reposition the tape at the appropriate record within the data set. This positioning is always performed in a forward direction. If the block count is zero or a negative number, the control program does no positioning. (If you want the control program to reposition the tape during a restart, normal header label routines (OPEN and EOV) must properly initialize the block count field of the data control block during the original creation. The block count field of the data control block must not be altered at restart time.)

If you are using a multivolume tape data set that is described by concatenated DD statements with the BLP (bypass label processing) option, the deferred option should be coded in the UNIT parameter of those DD statements. This prevents a misleading scheduler premount message that asks for the wrong volume to be mounted at restart time.

For additional information about tape labels, see *Magnetic Tape Labels and File Structure Administration*.

Repositioning and Preserving the Contents of Data Sets

The control program repositions data sets, but does not preserve their contents. After taking a checkpoint, you must ensure that data set contents are not changed in such a manner as to make successful postrestart processing impossible.

Recording Data Set Positioning

The checkpoint routine records positioning information for user data sets as follows:

- Data sets are repositioned at restart only if they were open when the checkpoint was taken. The OPEN routine positions all data sets opened after the checkpoint was taken.
- Unit-record data sets (printer, punch, or card reader) are not repositioned at restart.
- If you use EXCP to process a tape data set open at a checkpoint, ensure that the block count in the data set's data control block is correct. If the block count is incorrect, the data set may be incorrectly positioned by restart.

If input/output operations were requested, but not begun (for example, if a READ macro instruction was executed, but the related channel program was not started), the checkpoint routine then stops any processing associated with the I/O request, records the positioning information, and reestablishes I/O operations.

If I/O operations have already begun, the checkpoint routine waits until they are complete before recording positioning information.

Buffering of Data Set Records

When QSAM or QISAM is used to process a data set, an indeterminate number of virtual-storage buffers may contain data when a checkpoint is taken. If restart at a checkpoint occurs, the system's action depends on whether a card reader or another type of device is used to process the data set:

- **Card reader used (QSAM only).** Upon restart, existing buffer contents are released. The buffers are reprimed by reading records from the current data set into them.
- **Another device used.** Upon restart, the buffer contents are restored to virtual storage, and processing continues normally. It is not possible to predict the time (either before or after the checkpoint) at which a given record will be transferred between a buffer and the recording medium.

If you close a QSAM or QISAM data set immediately after restarting the program at a checkpoint, be aware that the data set may not be restored to the same condition it was in when the checkpoint was originally taken.

Preserving Data Set Contents

The system does not save and restore the contents of data sets. You, the user, must ensure that input data sets and system data sets contain all necessary data when restart occurs. If a data set on a direct access volume is open at the checkpoint, the data set's label (the DSCB in the VTOC) must have the same location and reflect the same extents upon restart as it did when the checkpoint was taken. (See "JCL Requirements and Restrictions on Automatic Restart" on page 36.)

If your program reads records from a data set, updates them, and writes them back to their original locations, it may be useless to take a checkpoint before completing this processing. If a checkpoint is taken earlier, postrestart processing is unsuccessful under these circumstances:

- Your program updates a record before abnormal termination and repeats the update after restart, and
- The updated record contents depend on the original contents.

For example, suppose that the purpose of the update is to exchange the positions of two fields in each record. If the record is updated twice, the fields are returned to their original positions, and the results are invalid. In a different application, an update might place a value in a record field, regardless of the field's original contents. You could then restart the step at a checkpoint taken before or during the update procedure, because an updated record would not be changed if updated again after restart.

When data set records are processed in an update-in-place manner (records are read, changed, and written back into their original location in the data set), bad data can be prevented if records updated after the last checkpoint are restored to their original state or if your program keeps track of the records that are updated and avoids updating them again during restart.

Allocating Devices during Checkpoint/Restart

When a job step is restarted from a checkpoint, the type of device allocated for the data set depends on the specification in the UNIT parameter of the DD statement. In addition to assuring the same device type for a checkpoint and restart, the system also attempts to allocate a device with the same optional features present at the time the checkpoint was taken.

- If a device address was specified (for example, UNIT=190), then a device of that type is allocated. It may or may not be the device requested.
- If a device name was specified, then a device of that type is allocated.
- If a user-defined name for a single type of device was specified (for example, UNIT=DISK1), then a device of the defined type is allocated.

- If a name for a mixed group of devices was specified (for example, UNIT=SYSDA), then a device of the same type as that used when the checkpoint was taken is allocated.

However, if the mixed group includes devices with varying optional characteristics (3340 with and without RPS or DASD shared and not shared between processors), a device with the same optional characteristics is not guaranteed. To avoid this, define some generics at system generation time that include only a single group with the same optional characteristics.

In jobs that can restart, you should avoid using generic unitnames or esoteric unitnames (for example, UNIT=TAPE) that contain more than one device type. Allocation failure may result during restart if too few units of a specific device type are available. For example, if UNIT=TAPE includes more than one type of an IBM 3400 Tape Drive (that is 3400-3,3400-4, and 3400-6), allocation failure may occur if the device type available at restart time is not the same device type as allocated in the original run of the job. To avoid this allocation failure, define some esoteric unitnames at sysgen time that include only a single device type.

Handling Checkpoint/Restart Errors

Input/Output Errors

The checkpoint routine issues return code 0C if it encounters a permanent I/O error when:

- Completing an outstanding VSAM I/O request
- Quiescing queued access method I/O operations
  - An exception occurs when QSAM is used and the skip or accept option is specified in the EROPT operand of the data set's data control block. In this case, code 00 is returned.
- Writing the checkpoint data set

When an access method other than QSAM or QISAM is used, your program can ensure that input/output operations are complete before it executes the CHKPT macro instruction; it can thereby avoid having read or written an erroneous record while quiescing.

If a permanent error occurs when the system reads a checkpoint data set to perform a restart, the restart step is terminated abnormally with the system completion code 13F. Further automatic restart of the step is not attempted.

Internal Errors

On return to the caller, register 15 will contain a return code, and register 0 will contain a reason code describing the error. For more information, see Appendix A, "Checkpoint/Restart Codes" on page 71.

Depending on the type of error, you may want to consult a SYS1.LOGREC or a SYS1.DUMPxx data set.

User Errors

Checkpoint will make a number of checks to discover user errors. If errors are discovered *before* the first record of the checkpoint entry is written, the checkpoint request will be refused; if errors are discovered **after** the first record of the checkpoint entry is written, the checkpoint entry will be invalid.

Using Checkpoint/Restart with Other Programs

Job and Job Step Accounting

The system accumulates processor time used for each job step and job. To access these time values, an installation can provide an accounting routine to receive control at step initiation, step termination, and job termination. Accounting routines are discussed in detail in *Supervisor Services and Macro Instructions*. The relationships between checkpoint/restart and the step time and job time values available to the accounting routine are listed below:

- At termination (either normal or abnormal) of an original execution, the step and job times accumulated are available to the accounting routine.
- At initiation of the restart step during an automatic restart, the step and job times accumulated for the original execution are again available to the accounting routine.
- At initiation of the restart step during a deferred restart, the step and job times are zero.
- At termination of a restart step and at all subsequent times when the accounting routine is given control during the restart execution, the step and job times reflect only the time used during the restart execution.
- If the TCBUSER field is to be used as a pointer to accounting information, the field is restored at restart time to its value at checkpoint time.

For example, in an original execution, Step A uses 2 minutes of processor time, and Step B uses 3 minutes of processor time and abnormally terminates. At step termination, the step time is 3 minutes and the job time is 5.

- If automatic restart is performed for Step B, a step time of 3 minutes and a job time of 5 are again available to the accounting routine at the reinitiation of Step B.
- If Step B then uses 4 minutes of processor time and terminates, a step time of 4 and a job time of 4 are available to the accounting routine at step termination.

The two values available at the time the restart step is initiated are provided for information purposes only. They are not reflected in the step and job running times presented at termination time of the restarted job. You need not be charged twice for the time accumulated up to the abend.

Another point to be considered in your accounting routine is the effect of a restart on the step sequence number available to the accounting routine. The following list indicates the sequence number presented to the accounting routine under the various restart conditions:

Condition	Step Sequence Value for Step n
Original execution	n
Automatic step restart	1
Automatic checkpoint restart	n+1
Deferred step restart	1
Deferred checkpoint restart	2

Whenever an automatic restart is performed, the step sequence value accurately reflects the position of the step in the job.

In the case of a deferred restart, the restarting step is the first step of the restart job.

Job Step Time Limit

The EXEC statement TIME parameter specifies a limit on the processor time used by the related step. With any kind of restart, the entire value of the limit specified for the job step applies to the restart step. For a deferred restart, you may specify a limit different from the limit originally specified.

If the processor time used by a step exceeds the specified limit while a checkpoint entry is written, the entry is invalid and an abnormal termination occurs. A preceding checkpoint entry performs a deferred restart. If it is a deferred restart and if a sufficient number of checkpoints are taken during the restart execution, the invalid checkpoint entry is overwritten by a valid entry.

Completion of Step or Job Termination at System Restart

If a step or a job is terminating when system failure occurs, the termination is completed during the system restart that the operator may perform after the failure. This occurs whether or not the step or job uses checkpoint/restart, although the job journal option must be in use.

If other than the last step of a job is terminating when the failure occurs, the termination completes during the system restart, and the next step of the job initiates.

If the last step of a job terminates, or, if the job terminates, all necessary terminations are completed.

If a job requests an automatic restart, then abnormally terminates and system failure occurs before the restart termination processing is complete, the processing completes during the system restart.

JES3

If a job cannot complete executing because of system failure and the job requested job journaling, the job is restarted (warm started) by the system. If the job is eligible for automatic restart, the operator is sent message IEF225D asking if the job should be restarted. If the job is not eligible for automatic restart or if the operator indicated that restart should not be attempted, any scratch or VIO data sets the job allocated are deleted and the job is processed based upon the FAILURE option specified in the MAIN JCL statement. If FAILURE=RESTART is specified, JES3 automatically reschedules the job for execution from the beginning of the first step.

Checkpoint/restart is not available to jobs executing under an ASP main processor. (For information about ASP main processors running under JES3, see *JES3 Introduction.*) However, JOBSTEP=CHKPNT can be specified on the CLASS initialization statement or the MAIN JCL statement to provide a checkpoint of the SYSOUT output at the end of each job step. (See *Initialization and Tuning Guide* for information about the CLASS statement and *JCL* for information about the MAIN statement.) You can see the output through the last completely executed step if the ASP main processor fails and the job is not restarted. (The FAILURE option on the CLASS or MAIN statement is PRINT or CANCEL.)

If automatic restart is requested for a job, the job's I/O is set up by JES3 for the first job step and not for the step being restarted. Canceling a job in a dependent network prevents successor jobs from executing if they are dependent upon successful completion of the canceled job. Any operator commands in the input stream of the job step being restarted are not executed. Restart of JES3-controlled jobs may be accompanied by message IAT2006 and/or IAT2575. For responses to these messages, see *JES3 Messages*.

Note: A job that contains generation data sets referenced by a relative generation number of +1 or greater and with a disposition of OLD, SHR, or MOD is failed by the JES3 interpreter service unless the UNIT parameter is included on the DD statement. With UNIT specified, the JES3 interpreter service permits this data set to be allocated on a deferred basis.

COBOL

The COBOL RERUN clause may be used to provide the COBOL user with linkage to checkpoint/restart. Cautions and restrictions on the use of checkpoint/restart also apply to the use of the RERUN clause. For further information, see the appropriate COBOL reference manual.

Sort/Merge

When performing a sort with the IBM Sort/Merge program, you can, by including the CKPT parameter in sort control statements, cause checkpoint entries to be written and an automatic checkpoint restart to be requested. The job control language can be used to request automatic or deferred step restarts or a deferred restart at a checkpoint.

PL/I

The PL/I user can use automatic and deferred step restart and can also take checkpoints and get automatic and deferred checkpoint restarts. To cause a checkpoint entry to be written and request an automatic checkpoint restart, the user codes **CALL PLICKPT**.

Each checkpoint entry in the checkpoint data set is identified by a system-generated checkid. A system message (including the checkid) at the console notifies the operator that a checkpoint entry is written.

The organization of the checkpoint data set is always physical sequential, and the data set may be written on magnetic tape or a direct access volume. Partitioned organization cannot be used.

A DD statement must be present in the job stream to define the checkpoint data set. The DISP parameter in this DD statement is used to specify whether single or multiple checkpoint entries are to be written. DISP=(NEW,KEEP) specifies a single checkpoint entry; DISP=(MOD,KEEP) specifies multiple checkpoint entries.

If CALL PLIREST is used in PL/I programs, the CKPTREST macro instruction must specify 4092 as an eligible user completion code. For more information, see "CKPTREST System Generation Specification" on page 3.

Virtual Fetch

Modules managed by virtual fetch cannot issue a checkpoint restart. In addition, job steps that call virtual fetch cannot issue a checkpoint restart.

Appendix A. Checkpoint/Restart Codes

Return Codes Associated with the CHKPT Macro Instruction

Some reason codes may also appear in messages associated with abends 13F and 23F. For more detailed information on the IHJxxxx reason codes, see *Checkpoint/Restart SVC Logic* and *System Messages*.

Return Code (Hex)	Meaning
00	**Successful completion.** Code 00 is also returned if the RD parameter was coded as RD=NC or RD=RNC to totally suppress the function of CHKPT.
04	**Restart has occurred** at the checkpoint taken by the CHKPT macro instruction during the original execution of the job. A request for another restart of the same checkpoint is normally in effect. If a deferred restart was performed and RD=NC, NR, or RNC was specified in the resubmitted deck, a request for another restart is not in effect. The following return codes have reason codes in register 0.
08	**Unsuccessful completion.** A checkpoint entry was not written, and a restart from this checkpoint was not requested. A request for an automatic restart from a previous checkpoint remains in effect. One of the following conditions exists.

Return Code	Meaning
001	Bad parameter list from caller
002	Missing DD card for checkpoint data set
003	Insufficient storage for a checkpoint or restart

005	Nonzero key length for a checkpoint data set
006	Incorrect record format for a checkpoint data set
007	Wrong DSORG for checkpoint data set
008	Active timer queue element
009	RB on chain of unacceptable type
010	Open graphics data set on DEB chain
011	Current task is a subtask
012	Multitask environment
013	Outstanding PCLINK
014	Outstanding WTOR
015	Invalid CHECKID length or format
016	Checkpoint data set is not tape or disk
019	Users DCB for checkpoint data set was open for input
021	CHKPT data set must have standard label
027	Checkpoint entry group does not fit on one volume
029	ANSI XLATE on CHKPT data set not allowed
032	ISAM data set open with DISP=SHR
041	VSAM—Create mode and no repositioning
043	VSAM—repositioning requested for ESDS with immediate upgrade
044	VSAM—direct processing of an RRDS in create mode
045	VSAM—GSR option in use
046	VSAM—CBUF processing is in use or was used
047	VSAM—CBIC option in use
048	VSAM—media manager in use
051	Active SPIE
054	Primary address ID and/or secondary address ID is not home address ID

	057	Checkpoint DD statement is concatenated
	061	Secondary addressing mode
	069	Too much VSMLIST data—user's storage is fragmented beyond checkpoint's capabilities
	075	STOW encountered a full directory on the checkpoint data set
	086	An abend occurred during a checkpoint or restart
	102	ISO/ANSI data set open at checkpoint is not supported
	206	VTAM data set is open
	208	Checkpoint data set is not empty
	209	Concurrent open on checkpoint data set
	210	DISP=SHR for checkpoint data set
	211	Checkpoint data set is not secure
	213	Checkpoint data set is a subsystem DS
	214	New checkpoint data set on shared DASDI
	224	SAM-SI
	250	IMAGELIB DCB open

0C **System-related or I/O-related problem:**

- An error occurred during the handling of a system request, such as ESTAE, SETLOCK, or PURGE; or
- An I/O error occurred in processing the CHKPT request; or
- A VSAM error was detected while preparing a VSAM data set for the CHKPT request.

004	Open failure on checkpoint data set
020	I/O error during open of CHKPT data set
022	Error reading or writing a SWA block control
023	I/O error during write on CHKPT data set
026	I/O error during STOW request
030	I/O error on QSAM or BSAM data set with EROPT not equal accept

042	VSAM—Repositioning error
055	VSMREGN failed
056	VSMLIST failed
059	An unexpected return was received by IHJGLU00
186	VSAM—close error; error code included
195	VSAM—no storage available
200	Purge failed
202	Setlock failed
204	WIJOURN failed—this checkpoint is unavailable for restart
205	WIJOURN failed—all checkpoints are unavailable for restart
207	I/O error using subsystem interface
240	ESTAE failed
241	VSAM—indeterminate error
242	VSAM—machine check

10 **Successful completion with possible error condition.** The task has control, by means of an explicit or implied use of the ENQ macro instruction, of a serially reusable resource; if the task terminates abnormally, it will not have control of the resource when the job step is restarted. Your program must, therefore, restore the enqueues.

Additional information regarding explicit and implicit use of the ENQ macro instruction will be found under "Requesting Serially Reusable Resources" on page 26.

000	Possible enqueue error
017	Insufficient storage to check enqueues; possible error
018	GQSCAN found an abnormal condition
028	I/O error during SYNCDEV due to a user data set

14 **Unsuccessful completion.** Internal error detected.

058	A bad parameter list was passed to IHJGLU00
090	Internal program error
091	Error building a message

18	**Error encountered restoring purged I/O.** The checkpoint was successful, but because of the error that was detected, restart may not be possible. In addition, the job now in progress may fail because of I/O errors.

215	Unsuccessful attempt to restore user's I/O during checkpoint.

When one of the errors indicated by code 08, 0C, 10, 14, or 18 occurs, the system prints an error message at the operator's console. The message contains a code that further identifies the error. The operator should report the contents of the message to the programmer.

Completion Codes Issued by Checkpoint/Restart

Completion Code (Hex)	**Meaning**
13F	System abend code 13F indicates that an error occurred during performance of a checkpoint restart. If a SYSABEND card is included in the job, a dump is produced, and the contents of the system control blocks, as shown in the dump, are unpredictable.

One of the following conditions exists.

Reason Code	**Meaning**
003	No storage
024	More than five volumes but JFCB does not point to a JFCBX.
031	I/O error during read from CHKPT data set
033	Cannot reposition to a tape data set or record because the block count in the DCB is negative. The block count can be labeled NL or BLP, and it is open for RDBACK.
034	Missing DD statement
035	Wrong length record during read on CHKPT data set
040	I/O error reading volume header record on a standard label tape during restart
050	A volume serial number is not the same at restart as it was at checkpoint
052	LPA or nucleus module has been deleted since checkpoint

053	LPA or nucleus module has been moved since checkpoint
058	A bad parameter list was passed to IHJGLU00
059	An unexpected return was received by IHJGLU00
063	I/O error while repositioning a tape to a record within a data set
074	Error reading DSCB
076	Extents in a data set's DEB do not equal those in the data set's DSCB
077	ISAM open error
078	Checkpoint was not on a 31-bit supervisor. Restart is.
079	A non-BSAM or non-QSAM data set was changed to dummy at restart
080	A compatibility interface data set was changed to dummy at restart
081	Checkpoint task and restart task V=R mismatch
082	TCAM is not active at restart, but a TCAM data set was open at checkpoint
086	User error causing an abend
087	Record order on checkpoint data set incorrect—detected by restart
088	Storage not allocated as expected—detected by restart
091	Error building a message
092	An error occurred while processing a partitioned data set
094	Nucleus routines or tables have moved or been deleted since checkpoint. LPA module entry points, which reside in the nucleus table, have been altered.
096	An error occurred while deleting a member from a partitioned data set
097	I/O error in nonstandard label routine during restart
098	No UCB available during restart
099	No DSAB found during restart

100	MSS error occurred during mount processing. See SSCR SSI.
101	I/O error reading vol label on DASD
103	A SSCR SSI checkpoint record was written, but the corresponding DCB is not in the open data set table at restart
104	I/O error while repositioning a tape volume to a data set
181	VSAM—during preformat
182	VSAM—during verify
183	VSAM—during put
184	VSAM—during index put
185	VSAM—open error; error code included
190	VSAM—unable to get cluster information
191	VSAM—unable to mount volumes
193	VSAM—DS was in create mode during checkpoint, but not restart
194	VSAM—IMM update was changed
195	VSAM—no storage available
196	VSAM—DS was modified between checkpoint and restart
197	VSAM—CBUF for cross-region sharing
198	VSAM—DS was extended to a new volume
199	VSAM—BLDVRP error
201	PGFIX failed
202	Setlock failed
203	Illegitimate call to restart
207	I/O error using subsystem interface
216	Region allocation at restart is incorrect
219	Password error on a password-protected tape
220	DSCB address has changed since checkpoint

	221	Wrong password given for password-protected data set
	222	Tape header label has changed since checkpoint
	240	ESTAE failed
	241	VSAM—indeterminate
	242	VSAM—machine check
	243	VSAM—bad SSCR

23F System abend code 23F indicates a security violation was detected during restart.

One of the following conditions exists.

251	A data set that was not a checkpoint data set at checkpoint time was found to be open to a secure checkpoint data set at restart.
252	In a job using more than one checkpoint data set, one of the checkpoint data sets (not the one used for restart) was deemed not secure.
255	Security violation—user not authorized to access a RACF-protected data set at restart

2F3 System abend code 2F3 indicates that a job was executing normally when system failure occurred.

33F System abend code 33F indicates that a user data set, (such as a 3480 subsystem), could not be synchronized. An I/O error occurred while writing data from a previous channel program to the media (deferred write error).

Appendix B. End-of-Volume Checkpoint Routine

You can specify, in the related data control block exit list, the address of a routine that receives control when end-of-volume is reached in processing a physical sequential data set (BSAM or QSAM). (For information about forming an exit list and coding the EOV exit list for physical sequential data sets, see *Data Facility Product: Customization.*) The routine is entered after a new volume is mounted and all necessary label processing is completed. If the volume is a reel of magnetic tape, the tape is positioned after the tape mark that precedes the beginning of the data. The end-of-volume exit routine may take a checkpoint by issuing the CHKPT macro instruction. The job step can be restarted from this checkpoint. When the job step is restarted, the volume is mounted and positioned as upon entry to the routine.

The end-of-volume exit routine returns control in the same manner as any other data control block exit routine. Restart becomes impossible if changes are made to the link pack area or nucleus after the checkpoint is taken. When the step is restarted, the virtual storage addresses of end-of-volume modules must be the same as when the checkpoint was taken.

List of Abbreviations

ACB	access method control block
ANSI	American National Standards Institute
APF	authorized program facility
AVR	automatic volume recognition
BDAM	basic direct access method
BISAM	basic indexed sequential access method
BPAM	basic partitioned access method
BSAM	basic sequential access method
DASD	direct access storage device(s)
DCB	data control block
DOS	disk operating system
DSCB	data set control block
ESPIE	extended specify program interruption exits
EXCP	execute channel program
FCB	forms control buffer
GDGNT	generation data group name table
I/O	input/output
IPL	initial program load
ISAM	indexed sequential access method
ISO	International Organization for Standardization
JFCB	job file control block
JFCBE	job file control block extension
MCP	message control program
MSS	mass storage system
PDS	partitioned data set
QISAM	queued indexed sequential access method
QSAM	queued sequential access method
RPS	rotational position sensing
SAM	sequential access method
SCT	special characters table
SIOT	step input/output table
SSCR	sybsytem checkpoint record
SSI	subsystem interface
SWA	scheduler work area
TCAM	telecommunications access method
TCB	task control block
TTR	track record
UCB	unit control block
UCS	universal character set
VIO	virtual input/output
VSAM	virtual storage access method
VTOC	volume table of contents

Index

MVS/Extended Architecture
Checkpoint/Restart
User's Guide
GC26-4139-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you wish a reply, give your name, company, mailing address, and telephone number.

____

____

____

____

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ____________________

Previous TNL ____________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)
Thank you for your cooperation.

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

GC26-4139-1

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape

Please do not staple

Fold and tape

IBM®

MVS/Extended Architecture Checkpoint/Restart User's Guide (File No. S370-40) Printed in U.S.A. GC26-4139-1

MVS/Extended Architecture
Checkpoint/Restart
User's Guide
GC26-4139-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you wish a reply, give your name, company, mailing address, and telephone number.

____
____
____
____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ____________________

Previous TNL ____________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)
Thank you for your cooperation.

GC26-4139-1

Reader's Comment Form

Fold and tape Please do not staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape Please do not staple Fold and tape

IBM®

MVS/Extended Architecture Checkpoint/Restart User's Guide (File No. S370-40) Printed in U.S.A. GC26-4139-1

MVS/Extended Architecture
Checkpoint/Restart
User's Guide

File Number S370-40

GC26-4139-01

Printed in U.S.A.