IBM Technical Newsletter

This Newsletter No. SN26-0932
Date March 30, 1979

Base Publication No. SY26-3825-1
File No. S370-30

Prerequisite Newsletters SN26-0909
SN26-0921

OS/VS2 MVS Virtual Storage Access Method (VSAM) Logic

This technical newsletter, a part of Release 3.8 of OS/VS2, provides replacement pages for the subject publication. These replacement pages remain in effect for any subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

Cover, edition notice	
3, 4	
9 - 14.1	(14.1 added)
25, 26	
63 - 66	
81, 82	
85, 86	
129 - 136	
174.11 - 174.18	(174.12 - 174.18 added)
219, 220	
266.1 - 266.12	(266.2 - 266.12 added)
275, 276	
279 - 282.1	(282.1 added)
305, 306	
311 - 312.1	(312.1 added)
339 - 340.1	(340.1 added)
361, 362	
371 - 378	
380.7, 380.8	
381 - 384.1	(384.1 added)
387 - 388.1	(388.1 added)
407, 408	

Each technical change is marked by a vertical bar to the left of the change.

Summary of Amendments

Changes included in this newsletter are summarized under "Summary of Amendments" following the list of diagrams.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

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

Printed in U.S.A.

SY26-3825-1
File No. S370-30

Systems

OS/VS2 MVS Virtual Storage Access Method (VSAM) Logic

Release 3.8

Includes Selectable Units:

Supervisor Performance #2	**VS2.03.807**
Data Management	**VS2.03.808**

IBM

Second Edition (January 1976)

This edition, as amended by technical newsletters SN26-0909, SN26-0921 and SN26-0932, applies to Release 3.8 of OS/VS2 and to any subsequent releases of that system unless otherwise indicated in new editions or technical newsletters.

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

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California U.S.A. 95150. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

PREFACE

This book describes the internal logic of the Virtual Storage Access Method (VSAM) and contains diagnostic information. It is directed to maintenance personnel and development programmers who require an in-depth knowledge of VSAM's design, organization, and data areas.

Organization of This Book

This book has the following major divisions:

- "Introduction," which describes the use of VSAM, how VSAM fits into the operating system, how VSAM interacts with the operating system and the user's program, and the major components of VSAM.
- "Method of Operation," which describes the functions performed by VSAM.
- "Program Organization," which describes the information contained in VSAM program listings and the flow of control between modules.
- "Directory," which lists VSAM modules and the method of operation diagrams related to each module.
- "Data Areas," which describes control blocks used by VSAM and describes the format of VSAM data and index records.
- "Diagnostic Aids," which contains useful information for locating the cause of problems in the VSAM procedures.
- "Glossary," which defines terms relevant to VSAM, and lists abbreviations and acronyms used in this book and in the VSAM program listings.
- "Index," which is a subject index to the book.

Required Reading

The following book should be read and understood before using this one:

- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838, which introduces VSAM concepts and contains definitive explanations of VSAM macros.

Related IBM Publications

- *Introduction to the IBM 3850 Mass Storage System (MSS),* GA32-0028
- *OS/VS2 MVS Mass Storage System Extensions: Communicator (MSSC),* LY35-0038
- *OS/VS Mass Storage System (MSS) Planning Guide,* GC35-0011
- *OS/VS Message Library: VS2 System Messages,* GC38-1002
- *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications,* GC26-3819
- *OS/VS2 Access Method Services,* GC26-3841
- *OS/VS2 Catalog Management Cross Reference,* SYB6-3843
- *OS/VS2 Catalog Management Logic,* SY26-3826
- *OS/VS2 Checkpoint/Restart Logic,* SY26-3820
- *OS/VS2 DADSM Logic,* SY26-3828
- *OS/VS2 Data Areas,* SYB8-0606
- *OS/VS2 I/O Supervisor Logic,* SY26-3823
- *OS/VS2 JCL,* GC28-0692
- *OS/VS2 MVS Data Management Macro Instructions,* GC26-3873
- *OS/VS2 Open/Close/EOV Logic,* SY26-3827
- *OS/VS2 Supervisor Services and Macro Instructions,* GC28-0683
- *OS/VS2 System Logic Library, Volumes 1-7,* SY28-0713 through SY28-0719 (All seven volumes can be ordered as SBOF-8210.)
- *OS/VS2 System Programming Library: Debugging Handbook, Volume 1,* GC28-0708 *Volume 2,* GC28-0709 (Both volumes can be ordered as GBOF-8211.)
- *OS/VS2 System Programming Library: Service Aids,* GC28-0674
- *OS/VS2 System Programming Library: System Management Facilities (SMF),* GC28-0706
- *OS/VS2 VSAM Cross Reference,* SYB6-3842

Using This Book

This book is designed to be used with the VSAM program listings in the microfiche for VSAM and with *OS/VS2 VSAM Cross Reference,* SYB6-3842, also on microfiche cards. Cross-reference reports are described in "Microfiche Cross-Reference Aids" in "Diagnostic Aids."

The diagrams in "Method of Operation" describe the major functions performed by VSAM; these diagrams are intended to be your key to a module name (and procedure name, as appropriate) in the listing. See "Reading Method of Operation Diagrams" in "Method of Operation" for a description of how to read these diagrams. For information on what is available in the program listings, see "Module Prologues" in "Program Organization."

ILLUSTRATIONS

Figures

Diagrams

SUMMARY OF AMENDMENTS

Release 3.8

OS/VS2 MVS Data Management Support for the Mass Storage System (MSS) Extensions Program Product Number 5740-XYG

MSS Extension Stage by Key Range

Stage by key range provides VSAM support for new MSS applications. Two new interfaces with three new macros allow users to prestage data. Prestaging allows data extents to be acquired in advance of the data's actual use. This reduces the number of page faults that might occur during processing. This function applies to MVS only.

Document Changes (July 1978)

This technical newsletter incorporates and replaces all previous SU information in this publication. Please replace identically numbered pages. Also, minor technical changes have been added. Revision bars appear beside all technical changes.

Changes for ASM Support

VSAM resources belonging to an Auxiliary Storage Management (ASM) RPL can be released.

OS/VS2 MVS Data Management (VS2.03.808)

Alternate Key Support

Feedback code 08 (paired with the 0 indicator in register 15) has been changed. For GET requests, the code indicates that a duplicate key follows; for PUT requests, it indicates that a duplicate key was created in an alternate index with the nonunique attribute.

OS/VS2 MVS Supervisor Performance #2 (VS2.03.807)

Multiple Key Support

Multiple key support provides the ability to acquire an independent, global shared resource pool for each of the system keys 0 through 7. Changes include:

- New fields and bit settings in AMBXN, CSL, HEB, VGTT, VSRT, and WSHD
- Changes to Method of Operation diagrams AD5, AF, and AH2
- Change ACBERFLG code 204 (CC) to reflect protection keys 0-7

SUSPEND/RESUME Processing

SVC 121 has been modified to use SUSPEND/RESUME, which are supervisor functions. SUSPEND places a specified request block in a suspended state; RESUME removes it from a suspended state and attempts to give control directly to the request block. VSAM Record Management has been changed to use SUSPEND/RESUME for synchronous processing. Specifically, new indicators have been defined in the IOMB and the PLH.

Support for ASM Rewrite

The support for ASM Rewrite allows the defining and preformatting of Swap spaces and treats SYS1.STGINDEX as a system data set. New fields and bit settings have been defined in the ACB, AMB, AMBL, OPW, and VGTT.

Document Changes

Control Interval Split

Before VSAM splits a control interval, VSAM writes the control interval to the direct-access device with the CIDF "busy flag" set on. When VSAM completes the control interval split, VSAM sets the busy flag off. Whenever a control interval with a busy flag is accessed, VSAM detects a previous control interval split interruption and attempts to remove any dupicated records from that control interval.

Release 3.7

Control Blocks in Common (CBIC)

This MVS-only function allows the user to specify that for data sets being processed with the improved control interval (ICI) option, all VSAM control blocks are to be built in common storage area (CSA).

SHOWCB Support for High-Allocated RBA

By specifying a new keyword (HALCRBA) in the FIELDS operand of the SHOWCB-ACB macro, the user can learn the high-allocated RBA for either the data or the index component. Whether the data or index RBA is returned is determined by the specified (or defaulted) content of the OBJECT operand.

VSAM SNAP Dump Facility

To increase the serviceability of VSAM, the VSAM SNAP dump facility has been added to provide hexadecimal dumps of VSAM-owned control blocks in CSA. Included in the dump are:

- The JSCBSHR field of the JSCB (used by VSAM to locate the VAT)
- The control blocks for open VSAM data sets processed with the global shared resources (GSR) option
- The control blocks making up the GSR pool
- The VGTT chain for the ASCB associated with the TCB being dumped and any PSBs associated with these VGTTs

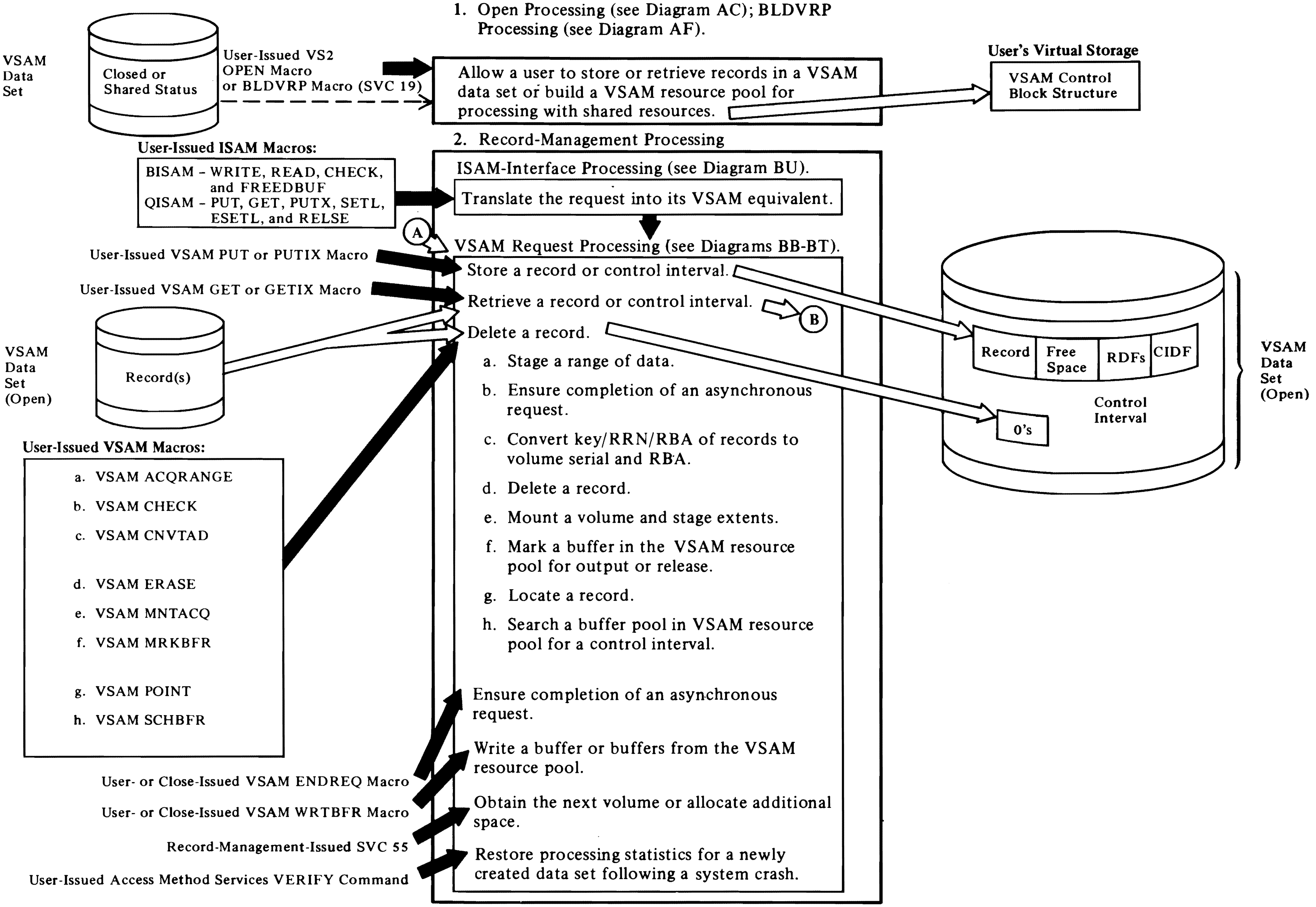
The dump facility is described in "Diagnostic Aids."

Control Block Manipulation Macros

Changes to support improved control block manipulation macro processing were made in

- Diagrams CA and CB
- "Data Areas," where KEYWDTAB, a branch table that controls execution of IDA019C1 and supports processing of the control block macros, is described
- "Diagnostic Aids," where a new return code, issued when a block to be displayed or tested does not exist because the data set is a dummy data set, has been added

Diagram AB. VSAM Overview



1. Open Processing (see Diagram AC); BLDVRP Processing (see Diagram AF).

Allow a user to store or retrieve records in a VSAM data set or build a VSAM resource pool for processing with shared resources.

2. Record-Management Processing

ISAM-Interface Processing (see Diagram BU).

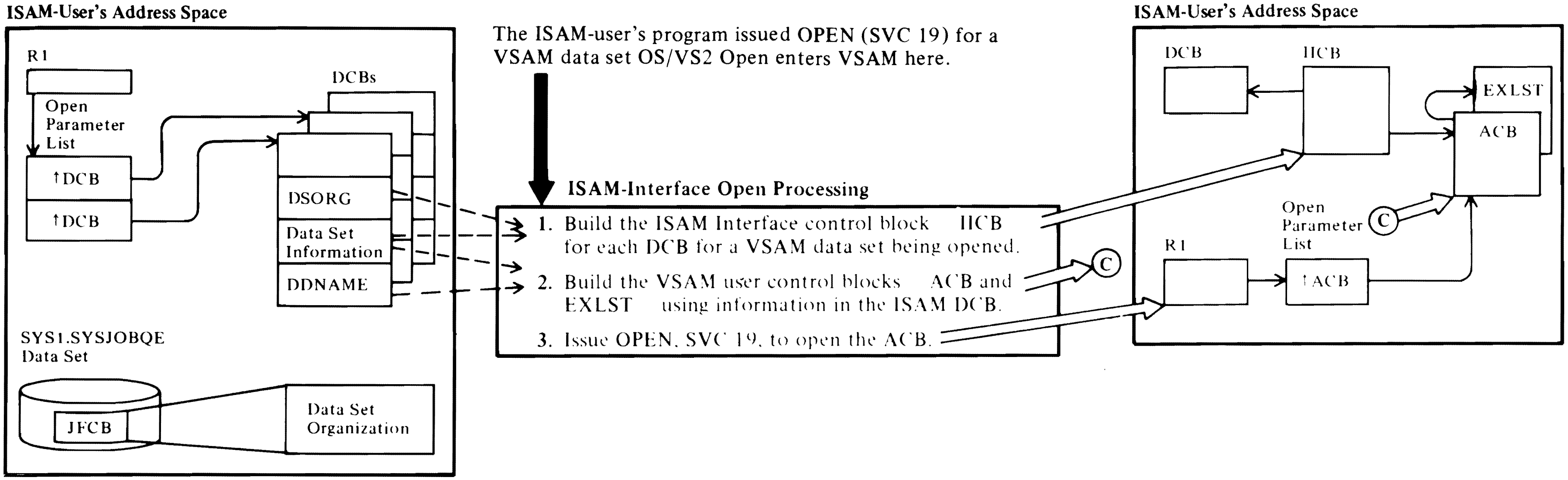
Translate the request into its VSAM equivalent.

VSAM Request Processing (see Diagrams BB-BT).

- Store a record or control interval.
- Retrieve a record or control interval.
- Delete a record.
  - a. Stage a range of data.
  - b. Ensure completion of an asynchronous request.
  - c. Convert key/RRN/RBA of records to volume serial and RBA.
  - d. Delete a record.
  - e. Mount a volume and stage extents.
  - f. Mark a buffer in the VSAM resource pool for output or release.
  - g. Locate a record.
  - h. Search a buffer pool in VSAM resource pool for a control interval.
- Ensure completion of an asynchronous request.
- Write a buffer or buffers from the VSAM resource pool.
- Obtain the next volume or allocate additional space.
- Restore processing statistics for a newly created data set following a system crash.

VSAM Data Set
Closed or Shared Status
User-Issued VS2 OPEN Macro or BLDVRP Macro (SVC 19)
User's Virtual Storage
VSAM Control Block Structure
User-Issued ISAM Macros:
BISAM - WRITE, READ, CHECK, and FREEDBUF
QISAM - PUT, GET, PUTX, SETL, ESETL, and RELSE
A
User-Issued VSAM PUT or PUTIX Macro
User-Issued VSAM GET or GETIX Macro
B
VSAM Data Set (Open)
Record(s)
User-Issued VSAM Macros:
a. VSAM ACQRANGE
b. VSAM CHECK
c. VSAM CNVTAD
d. VSAM ERASE
e. VSAM MNTACQ
f. VSAM MRKBFR
g. VSAM POINT
h. VSAM SCHBFR
User- or Close-Issued VSAM ENDREQ Macro
User- or Close-Issued VSAM WRTBFR Macro
Record-Management-Issued SVC 55
User-Issued Access Method Services VERIFY Command
Record
Free Space
RDFs
CIDF
0's
Control Interval
VSAM Data Set (Open)

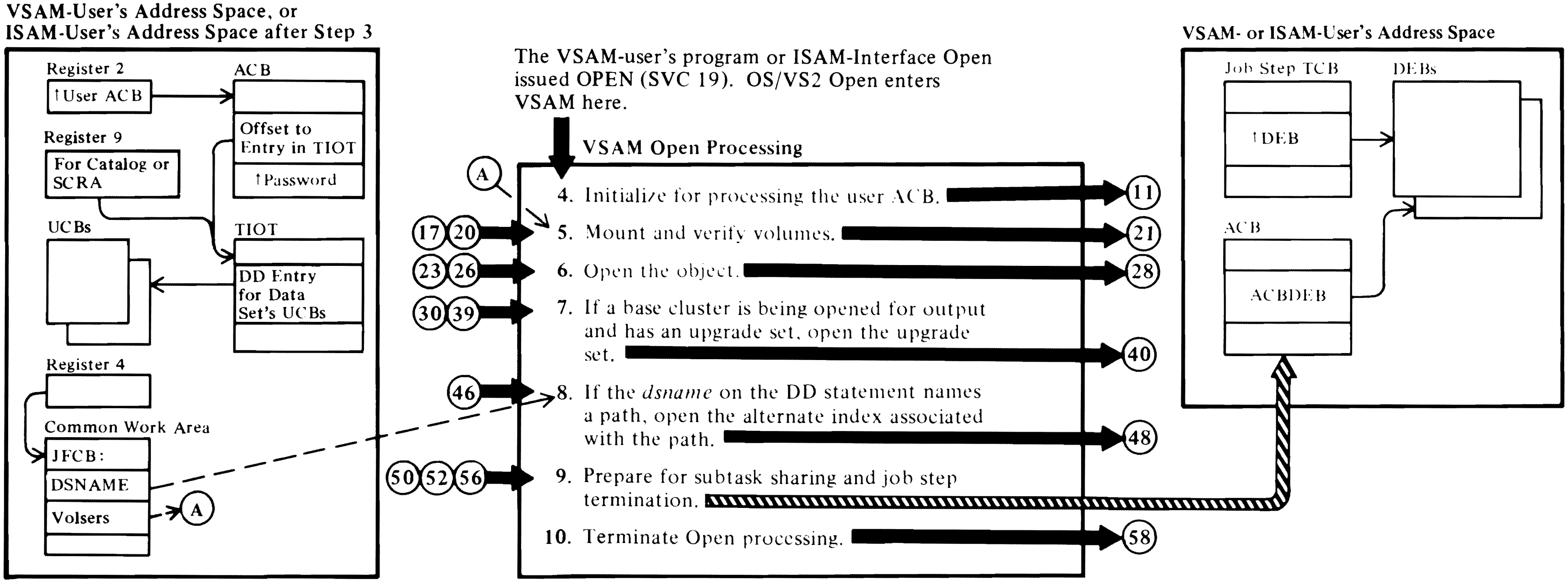
Diagram AC1. VSAM OPEN: Connect a User to a VSAM Data Set



The ISAM-user's program issued OPEN (SVC 19) for a VSAM data set OS/VS2 Open enters VSAM here.

ISAM-Interface Open Processing

1. Build the ISAM Interface control block IICB for each DCB for a VSAM data set being opened.
2. Build the VSAM user control blocks ACB and EXLST using information in the ISAM DCB.
3. Issue OPEN, SVC 19, to open the ACB.



The VSAM-user's program or ISAM-Interface Open issued OPEN (SVC 19). OS/VS2 Open enters VSAM here.

VSAM Open Processing

4. Initialize for processing the user ACB.
5. Mount and verify volumes.
6. Open the object.
7. If a base cluster is being opened for output and has an upgrade set, open the upgrade set.
8. If the *dsname* on the DD statement names a path, open the alternate index associated with the path.
9. Prepare for subtask sharing and job step termination.
10. Terminate Open processing.

Notes for Diagram AH1

The VSAM Task Close Executor (IDAOCEA2) gets control from the VS2 Data Management Resource Manager (IFG0TC0A, also called VS2 Task Close) for normal or abnormal termination of a task or of an address space, including "out-of-core" ABEND.

1 **IDAOCEA2**

2 **IDAOCEA2: JSTERM**

RMPLTCBA gives the location of the terminating TCB. TCBOTC indicates whether the region control task is being terminated.

3 **IDAOCEA2: JSTERM** calls **NMEMTERM**

4 **IDAOCEA2: JSTERM** calls **FALLVGTT**

5 **IDAOCEA2**

6 **IDAOCEA2: AMEMTERM** calls **FALLVGTT**

7 **IDAOCEA2: AMEMTERM** calls **SCANGSR**

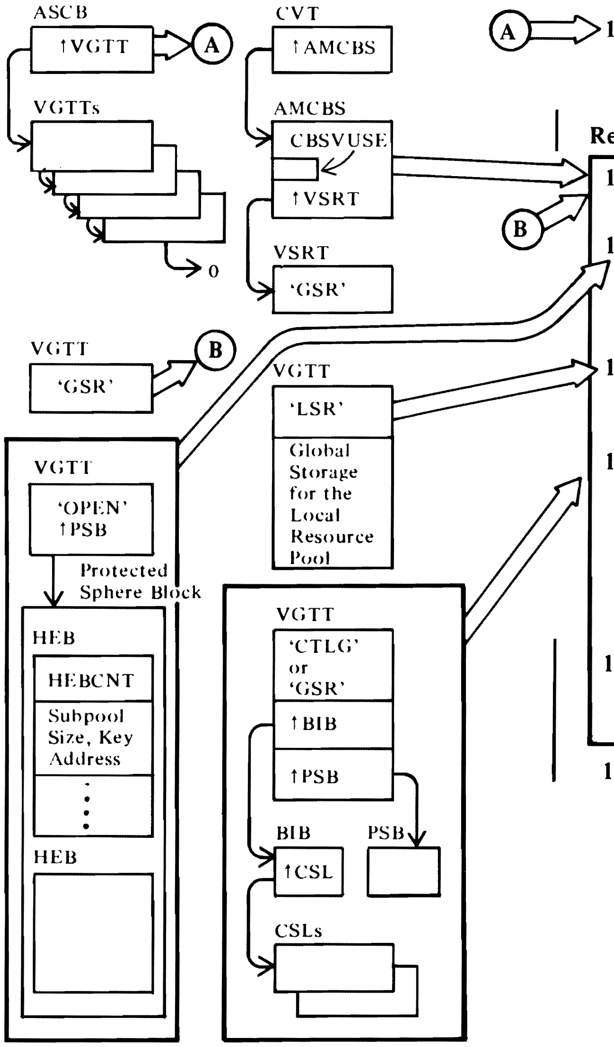
8 **IDAOCEA2**

9 **IDAOCEA2: NMEMTERM** calls **FALLVGTT**

10 **IDAOCEA2: NMEMTERM** calls **SCANGSR**

Diagram AH2. Recovery Termination: VSAM Task Close Executor

Free Storage



12. Remove the chain of VSAM global termination tables from the address space control block to the module work area.

Repeat steps 13–17 for each VGTT in the chain.

13. If the VGTT is for a global resource pool, decrement its use count, and free the VGTT.

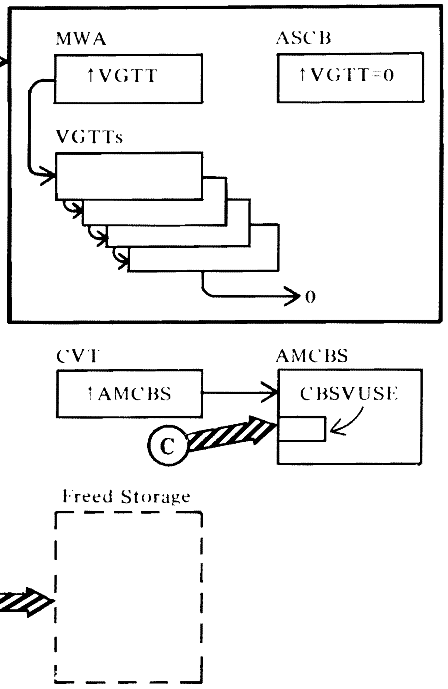
14. If the VGTT is for data sets that failed to close, free storage allocated for them in the common service area or the system queue area (global storage), and free the VGTT.

15. If the VGTT is for a local resource pool, free the VGTT and the IOSB, SRB, and PFL adjacent to it.

16. If the VGTT is for a catalog, a catalog recovery area in system storage, the mass storage volume inventory data set, or a data set that was processed with global shared resources, free storage obtained during open, the protected sphere block, the base information block, and the VGTT.

17. If the VGTT is an SBKR (Stage by Key Range) type, VGTT then free this VGTT and adjacent storage obtained by IDA0192E.

18. Return to the caller.



Notes for Diagram AH2

12 IDAOCEA2: FALLVGTT

The pointer in the address-space control block to the first VGTT in the chain of VGTTs is removed. MWANVGTT is pointed to the first VGTT to make a local VGTT chain.

In processing each VGTT in the chain (steps 13-16), it is made the current VGTT by pointing MWANVGTT to the next one, until there is no next one (in which case MWANVGTT is set to 0).

13 IDAOCEA2: FALLVGTT calls **VDECHAIN,** which calls **DECGVSRT**

If the AMCBS VSRT use count for the particular key isn't 0, it is decremented by the amount in the current VGTT. If the use count becomes negative, it is set to 0.

14 IDAOCEA2: FALLVGTT calls **FOPEN,** which calls **FREECORE**

A data set may not be closed because it was only partially opened or End of Volume or Close failed. The header elements in header element blocks describe storage that has been obtained for each data set. "Virtual-Storage Management" in "Diagnostic Aids" describes HEBs and indicates what subpools contain each type of control block.

FOPEN uses the VS2 GDT (global data area) to determine the address boundaries of global storage. If there is a protected sphere block, FOPEN processes each header element in it, using HEBCNT as an index. If the storage indicated in a header element is within the boundaries of global storage and in subpool 231, 239, 241, or 245, FOPEN uses its key to free it. After all HEBs are processed, FOPEN frees the protected sphere block and the VGTT.

15 IDAOCEA2: FALLVGTT calls **FLSR,** which calls **FREECORE**

When a local resource pool is built (during BLDVRP processing), storage in the system queue area is obtained for each trio of IOSB-SRB-PFL, and a VGTT is prefixed to each.

16 IDAOCEA2: FALLVGTT calls **FCTLG**

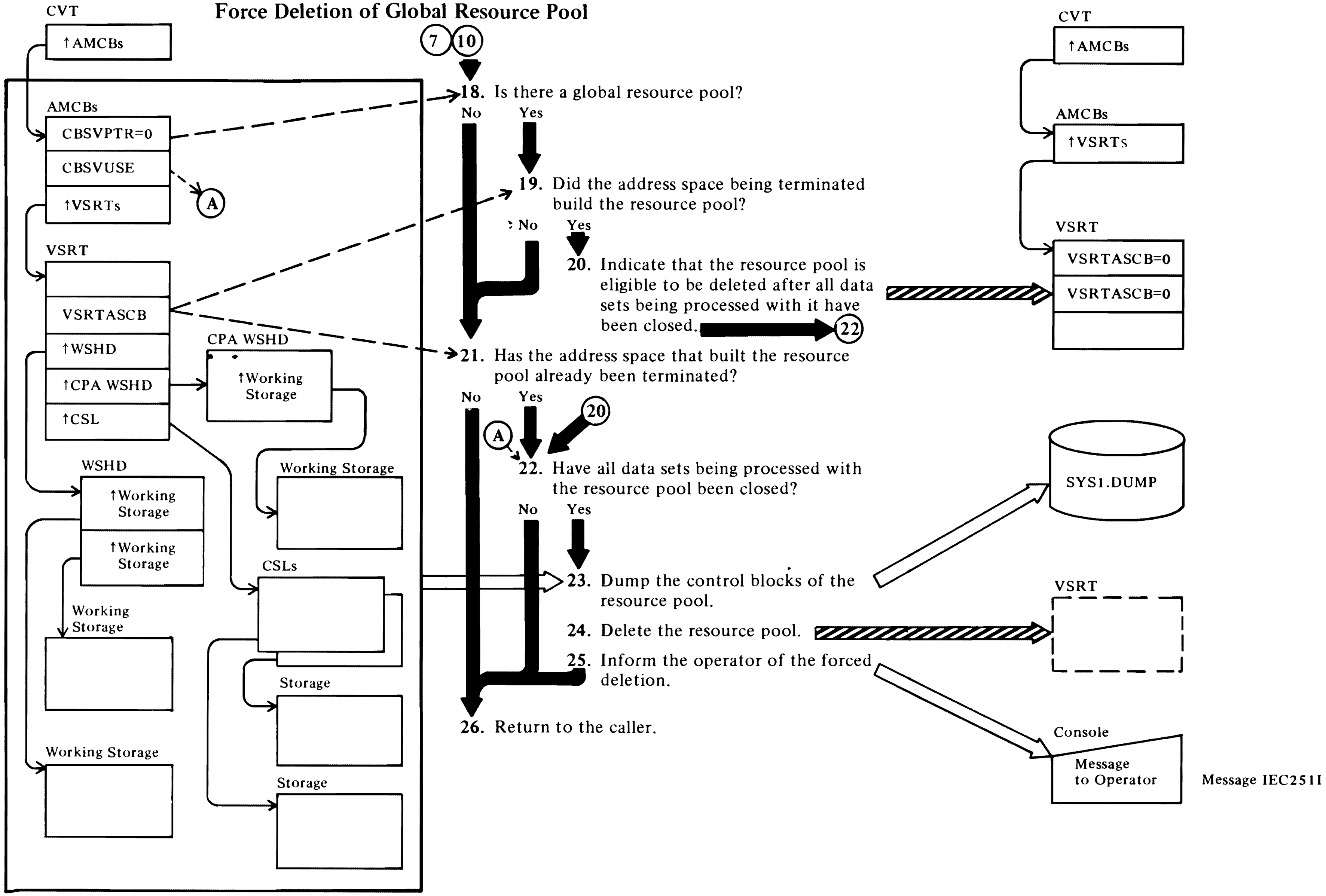
At the end of Open processing for a catalog, a catalog recovery area in system storage, or the mass storage volume inventory data set, Open frees the VGTT, so IDAOCEA2 cleans up for these only for termination that occurs during Open processing.

17 IDAOCEA2: FALLVGTT calls **FSBKR**

When obtaining storage for ECBs to use with ACQUIRE requests, IDA0192E concatentates a VGTT to the ECB storage. If the SBKR request never completes, IDAOCEA2 will free this subpool 241 storage.

Diagram AH3. Recovery Termination: VSAM Task Close Executor

Force Deletion of Global Resource Pool



18. Is there a global resource pool?

19. Did the address space being terminated build the resource pool?

20. Indicate that the resource pool is eligible to be deleted after all data sets being processed with it have been closed.

21. Has the address space that built the resource pool already been terminated?

22. Have all data sets being processed with the resource pool been closed?

23. Dump the control blocks of the resource pool.

24. Delete the resource pool.

25. Inform the operator of the forced deletion.

26. Return to the caller.

Diagram BA. Record Management Table of Contents

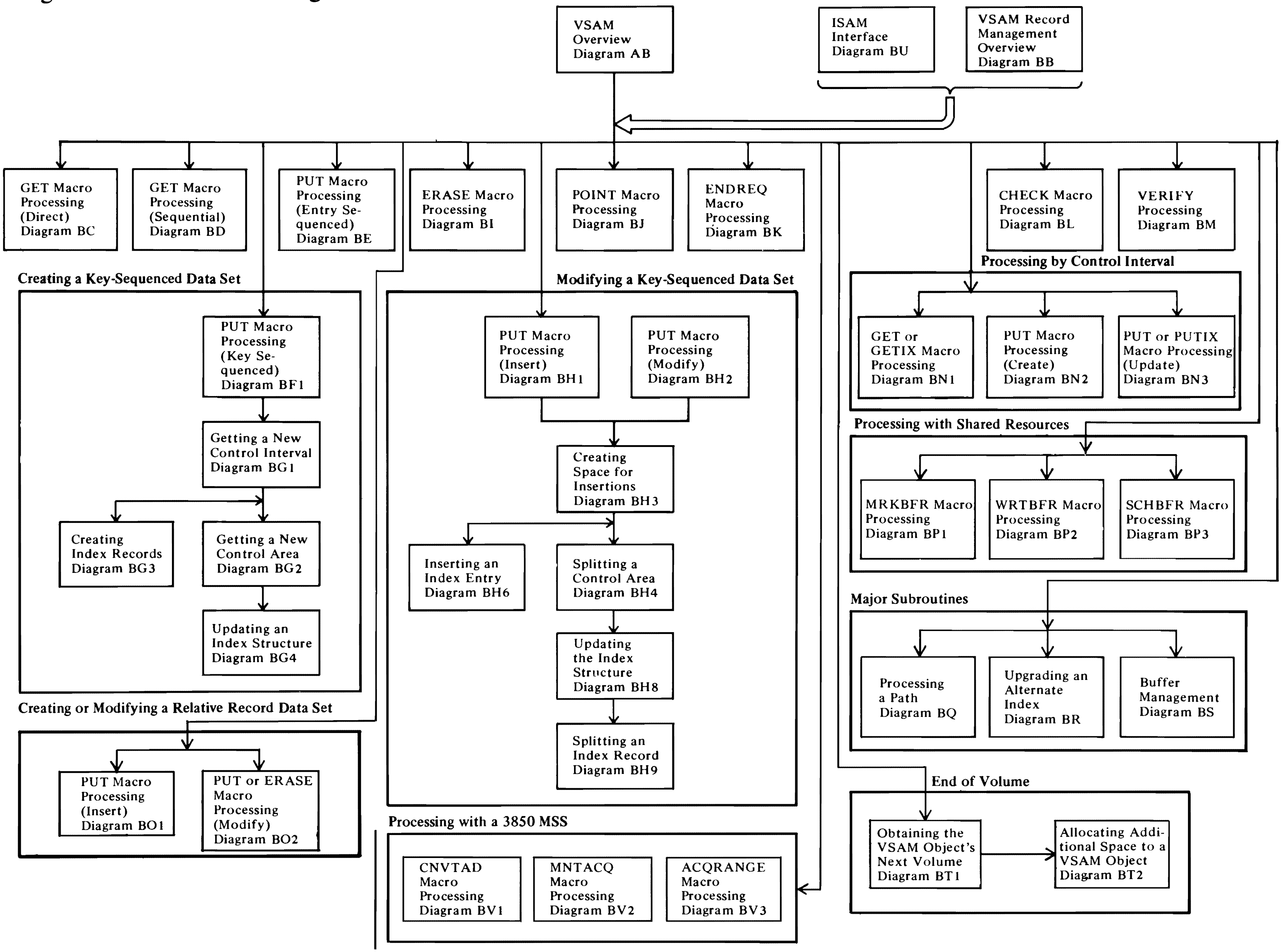
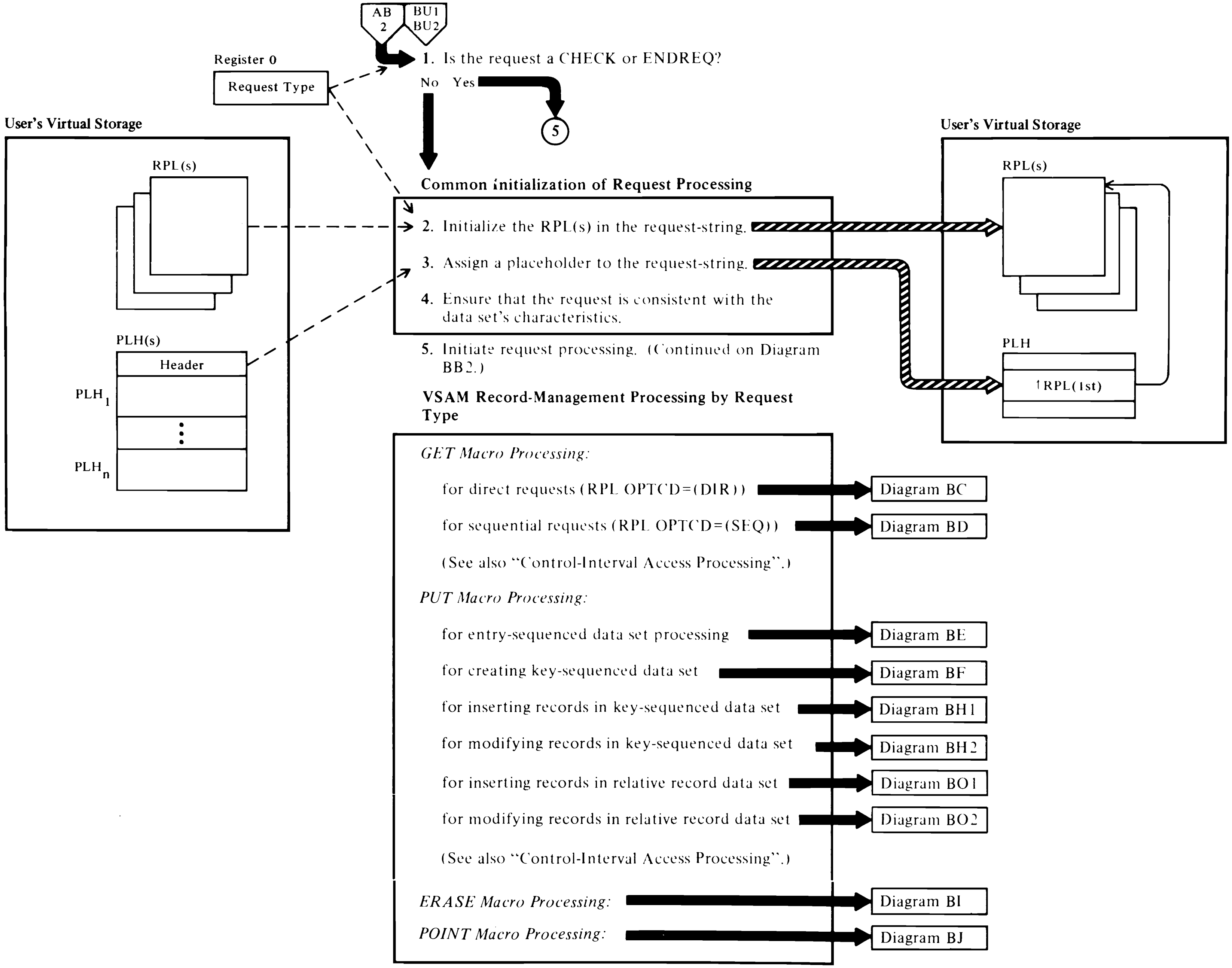

VSAM Overview Diagram AB
ISAM Interface Diagram BU
VSAM Record Management Overview Diagram BB
GET Macro Processing (Direct) Diagram BC
GET Macro Processing (Sequential) Diagram BD
PUT Macro Processing (Entry Sequenced) Diagram BE
ERASE Macro Processing Diagram BI
POINT Macro Processing Diagram BJ
ENDREQ Macro Processing Diagram BK
CHECK Macro Processing Diagram BL
VERIFY Processing Diagram BM
Creating a Key-Sequenced Data Set
PUT Macro Processing (Key Sequenced) Diagram BF1
Getting a New Control Interval Diagram BG1
Creating Index Records Diagram BG3
Getting a New Control Area Diagram BG2
Updating an Index Structure Diagram BG4
Creating or Modifying a Relative Record Data Set
PUT Macro Processing (Insert) Diagram BO1
PUT or ERASE Macro Processing (Modify) Diagram BO2
Modifying a Key-Sequenced Data Set
PUT Macro Processing (Insert) Diagram BH1
PUT Macro Processing (Modify) Diagram BH2
Creating Space for Insertions Diagram BH3
Inserting an Index Entry Diagram BH6
Splitting a Control Area Diagram BH4
Updating the Index Structure Diagram BH8
Splitting an Index Record Diagram BH9
Processing with a 3850 MSS
CNVTAD Macro Processing Diagram BV1
MNTACQ Macro Processing Diagram BV2
ACQRANGE Macro Processing Diagram BV3
Processing by Control Interval
GET or GETIX Macro Processing Diagram BN1
PUT Macro Processing (Create) Diagram BN2
PUT or PUTIX Macro Processing (Update) Diagram BN3
Processing with Shared Resources
MRKBFR Macro Processing Diagram BP1
WRTBFR Macro Processing Diagram BP2
SCHBFR Macro Processing Diagram BP3
Major Subroutines
Processing a Path Diagram BQ
Upgrading an Alternate Index Diagram BR
Buffer Management Diagram BS
End of Volume
Obtaining the VSAM Object's Next Volume Diagram BT1
Allocating Additional Space to a VSAM Object Diagram BT2

Diagram BB1. VSAM Record Management Overview



1. Is the request a CHECK or ENDREQ?

Common initialization of Request Processing

2. Initialize the RPL(s) in the request-string.
3. Assign a placeholder to the request-string.
4. Ensure that the request is consistent with the data set's characteristics.
5. Initiate request processing. (Continued on Diagram BB2.)

VSAM Record-Management Processing by Request Type

GET Macro Processing:

for direct requests (RPL OPTCD=(DIR))

for sequential requests (RPL OPTCD=(SEQ))

(See also "Control-Interval Access Processing".)

PUT Macro Processing:

for entry-sequenced data set processing

for creating key-sequenced data set

for inserting records in key-sequenced data set

for modifying records in key-sequenced data set

for inserting records in relative record data set

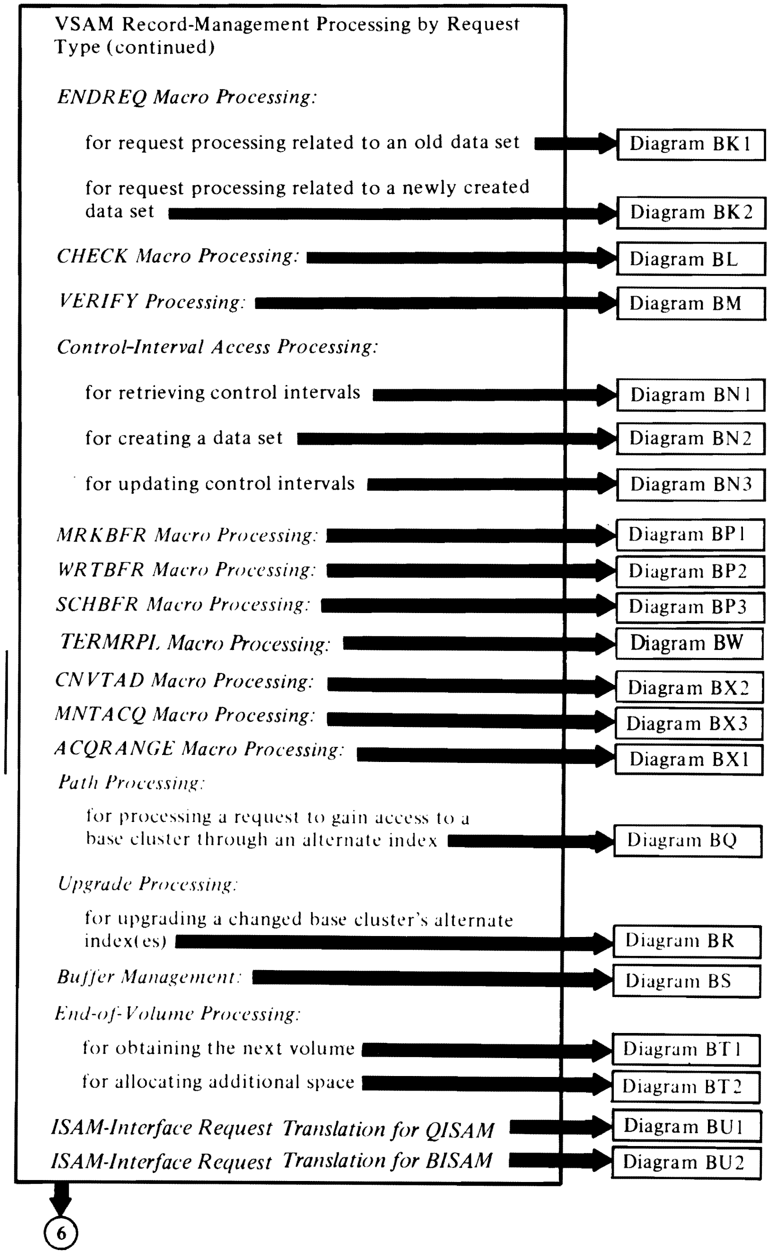
for modifying records in relative record data set

(See also "Control-Interval Access Processing".)

ERASE Macro Processing:

POINT Macro Processing:

Diagram BB2. VSAM Record Management Overview



VSAM Record-Management Processing by Request Type (continued)

ENDREQ Macro Processing:

for request processing related to an old data set → Diagram BK1

for request processing related to a newly created data set → Diagram BK2

CHECK Macro Processing: → Diagram BL

VERIFY Processing: → Diagram BM

Control-Interval Access Processing:

for retrieving control intervals → Diagram BN1

for creating a data set → Diagram BN2

for updating control intervals → Diagram BN3

MRKBFR Macro Processing: → Diagram BP1

WRTBFR Macro Processing: → Diagram BP2

SCHBFR Macro Processing: → Diagram BP3

TERMRPL Macro Processing: → Diagram BW

CNVTAD Macro Processing: → Diagram BX2

MNTACQ Macro Processing: → Diagram BX3

ACQRANGE Macro Processing: → Diagram BX1

Path Processing:

for processing a request to gain access to a base cluster through an alternate index → Diagram BQ

Upgrade Processing:

for upgrading a changed base cluster's alternate index(es) → Diagram BR

Buffer Management: → Diagram BS

End-of-Volume Processing:

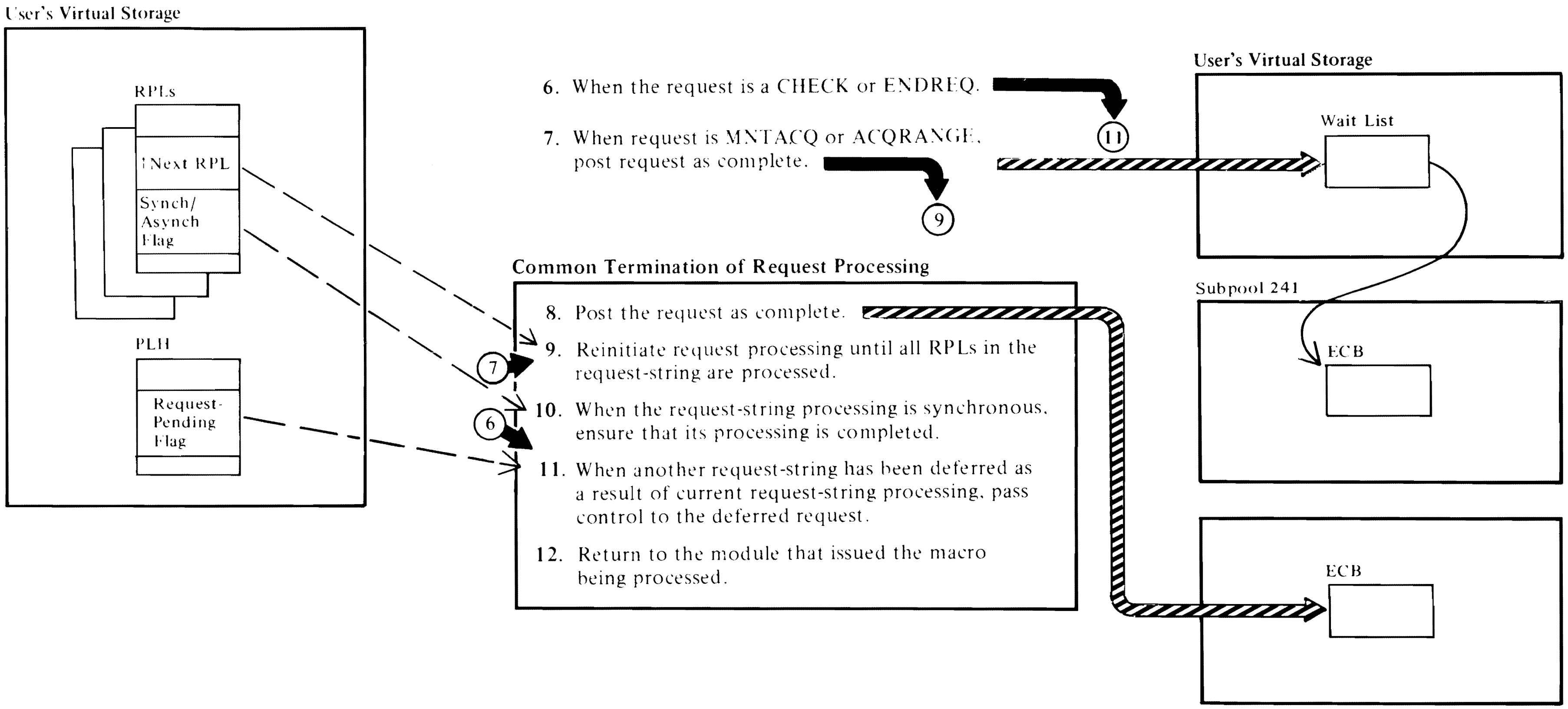
for obtaining the next volume → Diagram BT1

for allocating additional space → Diagram BT2

ISAM-Interface Request Translation for QISAM → Diagram BU1

ISAM-Interface Request Translation for BISAM → Diagram BU2

Diagram BB3. VSAM Record Management Overview



6. When the request is a CHECK or ENDREQ.

7. When request is MNTACQ or ACQRANGE, post request as complete.

Common Termination of Request Processing

8. Post the request as complete.

9. Reinitiate request processing until all RPLs in the request-string are processed.

10. When the request-string processing is synchronous, ensure that its processing is completed.

11. When another request-string has been deferred as a result of current request-string processing, pass control to the deferred request.

12. Return to the module that issued the macro being processed.

Notes for Diagram BJ

1 **Keyed Processing—Key-Sequenced Data Set**

IDA019RA

When the request is keyed, an index search must be performed. The index level where the search begins is determined as follows:

- For skip-sequential processing, the index search starts at the sequence set. The search normally starts at the index record pointed to by the current PLH. If the PLH is invalid, the search starts at the first record in the sequence set.
- For direct processing, the search starts at the highest level of the index.

IDA019RA calls **IDA019RB** which calls **IDA019RZ (IDAGRB)**

The index record at which the search is to start is moved into an index buffer.

IDA019RB calls **IDA019RC**

The index record is searched for an entry that is greater than or equal to the search key.

IDA019RB

When the search is unsuccessful, the next record in logical sequence is searched. If the search is successful and a lower index level exists, the search is performed on the index records in the lower level.

Keyed Processing—Relative Record Data Set

IDA019RR

The relative record number that is specified as a search argument is converted to the RBA of the control interval that contains the record, plus the offset of the record in the control interval.

IDA019RR calls **IDA019RR (IDARRDRL)**

If the RBA is within the data set, the control interval's contents are retrieved. If the RBA is not within the data set, then:

- With KGE, end-of-data is indicated and positioning is established at the end of the data set
- Without KGE, no-record-found is indicated

Addressed Processing

2 **IDA019RA**

The RBA that is specified as a search argument is converted into the RBA of the boundary of the control interval that it falls within.

2 **IDA019RA** calls **IDA019RZ (IDAGRB)**

Relative Record Processing

IDARRDRL calls **IDA019RZ (IDAGRB)**

The control interval is read by RBA.

3 **IDA019RA**

The control interval is scanned to determine whether the key or RBA provided as a search argument is within the retrieved control interval. (Note: The RBA must represent a valid record boundary within the control interval.)

When the key search is unsuccessful, a test is made to determine whether a control interval split has been performed by another request-string operating concurrently with the current request. If a split has occurred, processing returns to step 1 to perform a new index search.

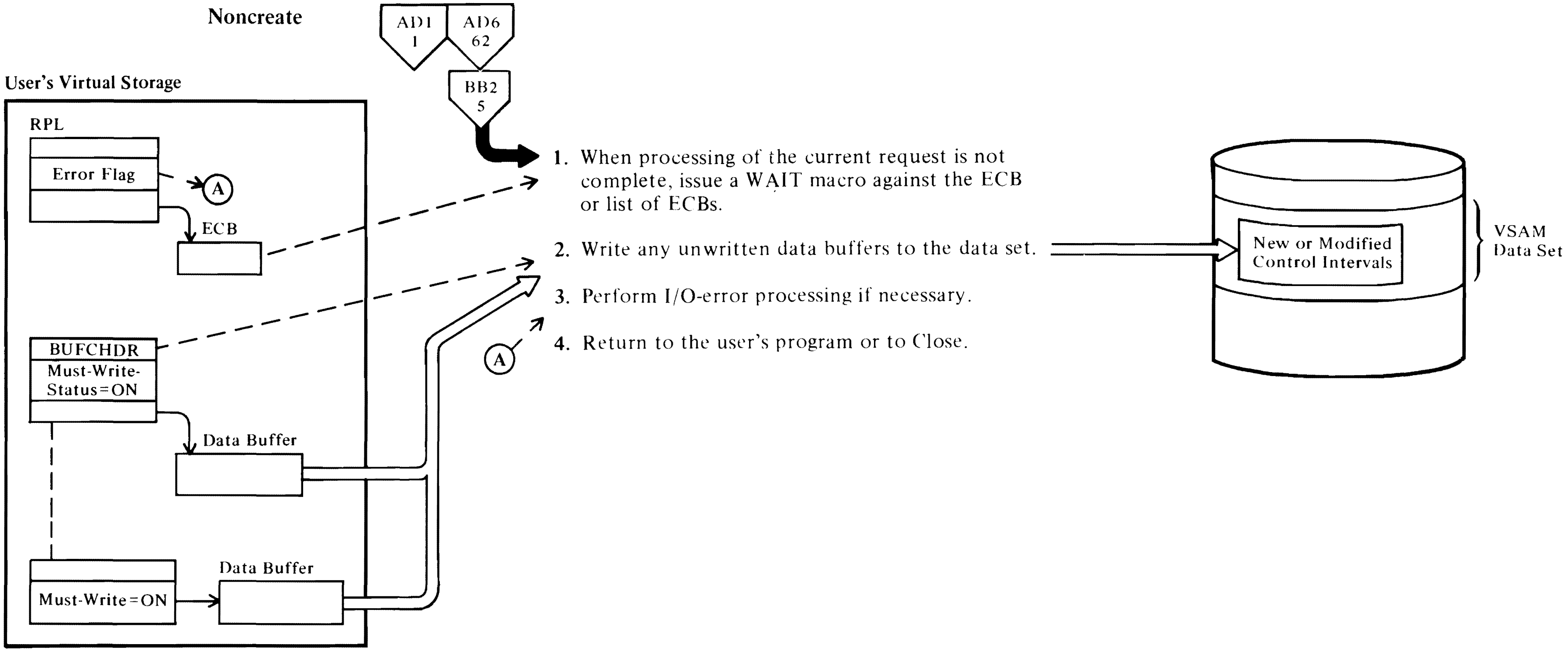
Relative Record Processing

IDA019RR: IDARRDRL

Positioning is established by saving in the PLH pointers to the record and its RDF and the RBA of the control interval.

Diagram BK1. ENDREQ Processing

Noncreate



1. When processing of the current request is not complete, issue a WAIT macro against the ECB or list of ECBs.
2. Write any unwritten data buffers to the data set.
3. Perform I/O-error processing if necessary.
4. Return to the user's program or to Close.

Notes for Diagram BK1

1 **IDA019R1: FINDOPLH**

The placeholder (PLH) for the request string associated with the ENDREQ request is located by searching the placeholder list for a placeholder that points to the RPL identified by the ENDREQ.

IDA019RP: IDAENDRQ

Other RPLs (if any) in the request string are prevented from being processed by setting a flag in the placeholder that indicates that an ENDREQ request is being processed. (Note: Once a request-string starts processing, it continues until all of the RPLs in the string are processed or until an ENDREQ is issued. When an ENDREQ is issued, processing against the request-string is terminated when processing of the current RPL in the string has completed.) If the current request is not complete, the WAIT is issued to ensure completion.

IDA019RP: IDAENDRQ calls IDA019SM if a previous request was MNTACQ or ACQRANGE to issue multiple WAIT for list of ECBs.

IDA019SM issues SVC 109 routing code 6 which calls IGX00006 and then IDA0192E to freemain the WAIT list and the ECBs.

2 **IDA019RP: IDAENDRQ**

Before performing any I/O, the processing is forced into synchronous mode to ensure that control is not returned to the user until I/O associated with the ENDREQ request is completed. When I/O is completed, asynchronous processing is restored if the processing was previously asynchronous.

IDA019RP: IDAENDRQ (calls **IDA019RZ (IDAWRBFR)**)

All unwritten data buffers associated with the current placeholder are written.

3 **IDA019RP**: calls **IDA019R5**

The buffer control block (BUFC) chain for the I/O-Management block (IOMB) in error is searched for a BUFC with an error indicator.

IDA019R1: R1ENDREQ (calls **IDA019R5)**

Error conditions are analyzed and an error message is built.

IDA019RP calls **IDA019R5 (IDAEXITR)**

For processing if a SYNAD routine exists.

4 **IDA019RP: IDAENDRQ** (calls **IDA019RZ (IDASBF)**)

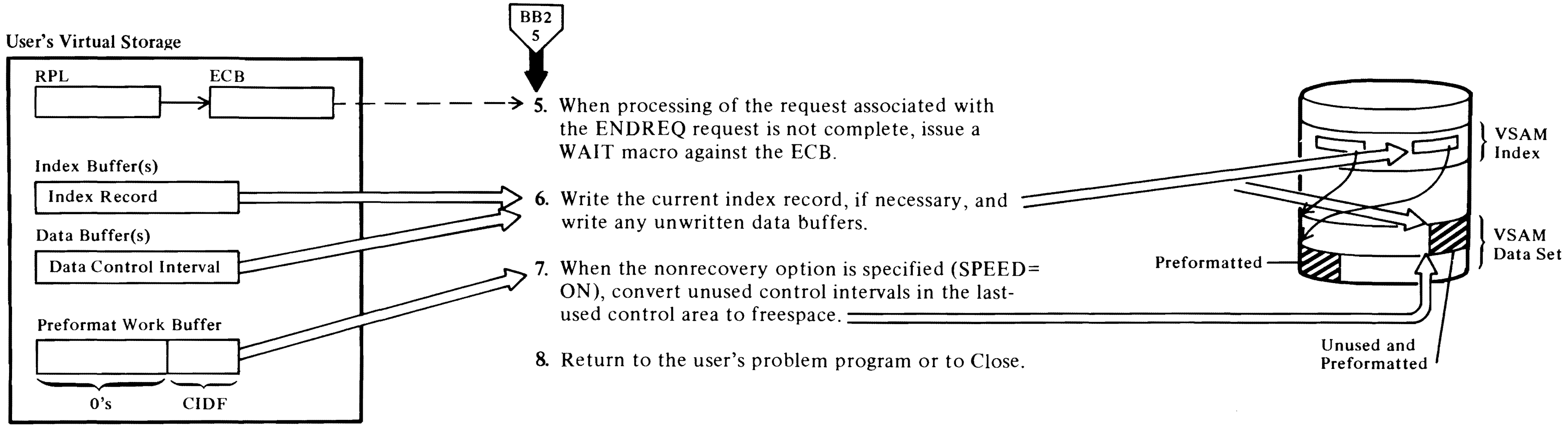
Excess data buffers are released from the current placeholder.

IDA019RP: IDAENDRQ

The placeholder is released from the current request string.

Diagram BK2. ENDREQ Processing

Create



5. When processing of the request associated with the ENDREQ request is not complete, issue a WAIT macro against the ECB.

6. Write the current index record, if necessary, and write any unwritten data buffers.

7. When the nonrecovery option is specified (SPEED=ON), convert unused control intervals in the last-used control area to freespace.

8. Return to the user's problem program or to Close.

Notes for Diagram BK2

5 **IDA019R1: FINDOPLH**

The placeholder for the request string associated with the ENDREQ request is located by searching the placeholder list for a placeholder that points to the RPL identified by the ENDREQ.

IDA019RP: IDAENDRQ

Other RPLs (if any) in the request string are prevented from being processed by setting a flag in the placeholder that indicates that an ENDREQ request is being processed. (Note: Once processing for a request-string starts, it continues until all of the RPLs in the string are processed or until an ENDREQ is issued. When an ENDREQ is issued, processing against the request-string is terminated when processing of the current RPL in the string has completed.) If the current request is not complete, the WAIT is issued to ensure completion.

6

The processing for step 6 ensures that the index entry for the last data control interval in the current data buffer for the current control area will fit in the index record for the current control area. Otherwise, when processing is resumed and when the dummy entry in the index record does not have space for the key, the data control interval would have to be moved to a new control area and have its index entry placed in the index record for the new control area.

IDA019RP calls **IDA019RG**

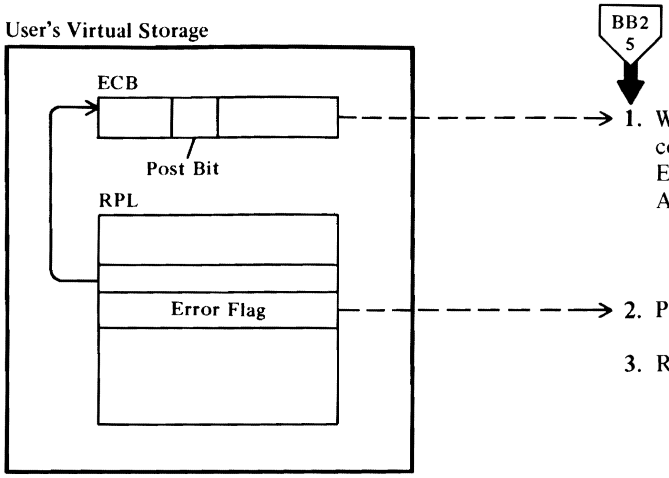
Before writing the index buffer, the following processing is performed: IDA019RG checks the leftmost entry, a dummy entry for the current control interval, in the index record to determine whether a maximum length key will fit in the remaining index record freespace. If there is adequate space to insert a key, IDA019RG writes out the current index record and frees the index-create work area(s) (ICWAs).

If there is inadequate space to contain a key for the control interval in the current data buffer, IDA019RP calls IDA019SA, which recalls IDA019RG, in order to have the entry inserted into the index record. IDA019RG returns a no-fit indicator to IDA019SA, which forces an end-of-control-area situation for IDA019SA (EOCA) processing. In response to the no-fit indicator, IDA019SA (EOCA) writes out any full data buffers (less the current data buffer) to the data set and acquires a new control area.

7 **IDA019RP** calls **IDA019RZ (IDAWRBFR)**

8 **IDA019RP** calls **IDA019RK**

Diagram BL. CHECK Processing



1. When the request's ECB is not posted as being complete, a WAIT macro is issued against the ECB or list of ECBs for MNTACQ or ACQRANGE requests.

2. Perform error processing if necessary.

3. Return to the user's processing program.

Notes for Diagram BL

1 **IDA019R1: FINDOPLH**

The placeholder for the request-string associated with the CHECK request is located by searching the placeholder list for a placeholder that points to the RPL identified by the ENDREQ.

IDA019R1: R1CHECK

A WAIT macro is issued to ensure that the asynchronous request, for which the CHECK was issued, has completed.

IDA019R1: R1CHECK

Also performs CHECK processing for CNVTAD, MNTACQ, and ACQRANGE. If the request was CNVTAD, a WAIT is issued.

R1CHECK calls **IDA019SM**

If the request for which CHECK issued was MNTACQ or ACQRANGE to issue a WAIT for lists of ECBs, IDA019SM issues SVC 109 routing code 6, which calls IGX00006 and then IDA0192E to freemain the WAIT list and the ECBs. See *OS/VS2 MVS Mass Storage System Extensions: Communicator (MSSC)* for a description of IGX0006 and IDA0192E.

2 **IDA019R1** calls **IDA019R5**

The buffer control block (BUFC) chain for the I/O block (IOB) in error is searched for a BUFC with an error indicator.

Error conditions are analyzed and an error message is built.

IDA019R1 calls **IDA019R5 (IDAEXITR)**

For processing if a SYNAD routine exists.

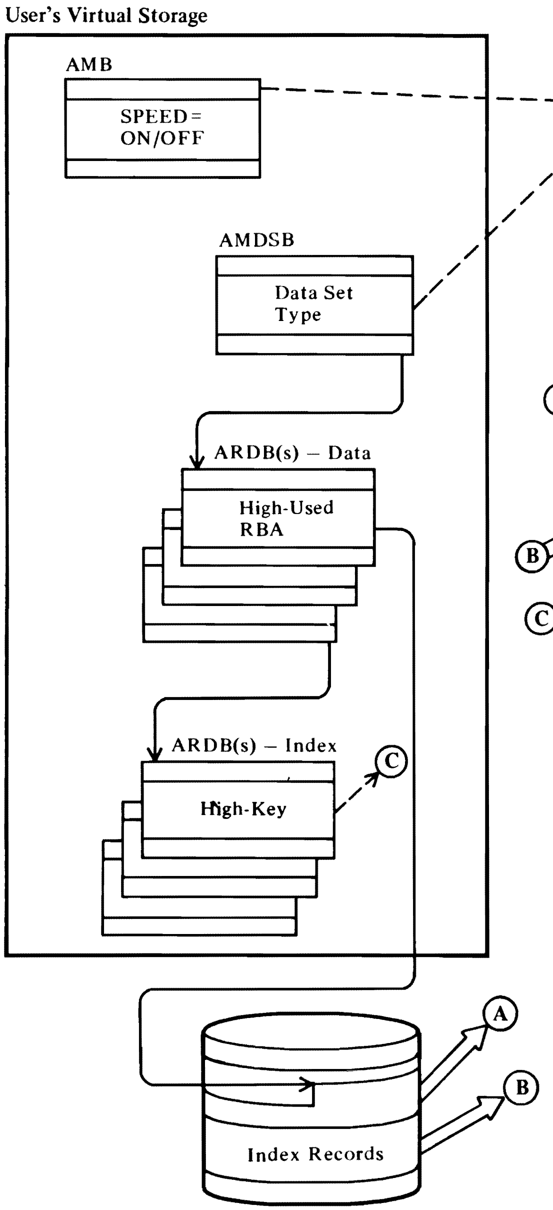
3 **IDA019R1: R1CHECK**

The check process is repeated for each RPL (if any) in the RPL-string associated with the RPL that the CHECK was originally issued against.

The placeholder is released if necessary.

The placeholder remains associated with the current request-string unless the processing is direct. For direct processing, the next request must be repositioned to an address in the data set. For sequential or skip-sequential processing, the positioning information established by a prior request is used by the succeeding request.

Diagram BM. VERIFY Processing



BB2
5

1. When the data set is key sequenced and the recovery option (SPEED=OFF) is specified, perform the following:

   - Search the data space associated with each index and data ARDB for a software end-of-file marker in order to establish a valid high-used RBA in each ARDB.

     BS2 1 BS1 1

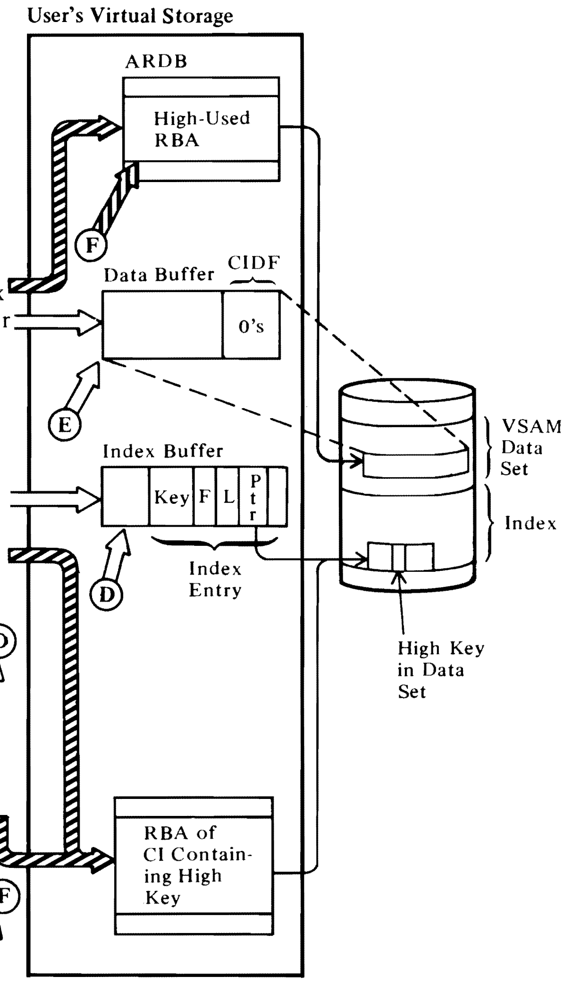
   - Search the index to establish the RBA of the data control interval containing the highest key value in each data ARDB's space.

     BS2 1 BS1 1

2. When the data set is key sequenced and the nonrecovery option (SPEED=ON) is specified, perform the processing described for step 1 except that a high-used RBA cannot be established for the data ARDB(s).

3. When the data set is entry sequenced, establish a high-used RBA for the data ARDB, as described in step 1, only if recovery (SPEED=OFF) is specified.

4. Return to caller.



Notes for Diagram BW

1 **IDA019R1: TERMRPL**

IDA019R1 (TERMRPL) calls **IDA019R1 (FINDOPLH)**

The placeholder (PLH) for the request string associated with the TERMRPL request is located by searching the placeholder list for a placeholder that points to the RPL identified by the TERMRPL.

IDA019R1 (TERMRPL) calls **IDA019SN**

2 **IDA019SN**

Validity checks the data set attributes, RPL options, and processing conditions before performing the TERMRPL request.

Releases all owned VSAM resources that are commonly shared by other requests.

IDA019SN calls **IDA019RZ (IDAFREEB)**

If the data set is a KSDS, IDA019SN calls IDA019RZ (IDAFREEB). IDAFREEB frees the index buffer(s) that belong to the placeholder.

IDA019SN calls **IDA019RZ (IDASBF)**

Excess data buffers are released from the placeholder for reuse.

IDA019SN calls **IDA019SE (IDARSTRT)**

An attempt is made to restart all deferred synchronous requests that are not in the same address space as the RPL identified by TERMRPL.

If a deferred request that needs restarting is asynchronous, an error code will be returned to the user indicating TERMRPL cannot restart an asynchronous request.

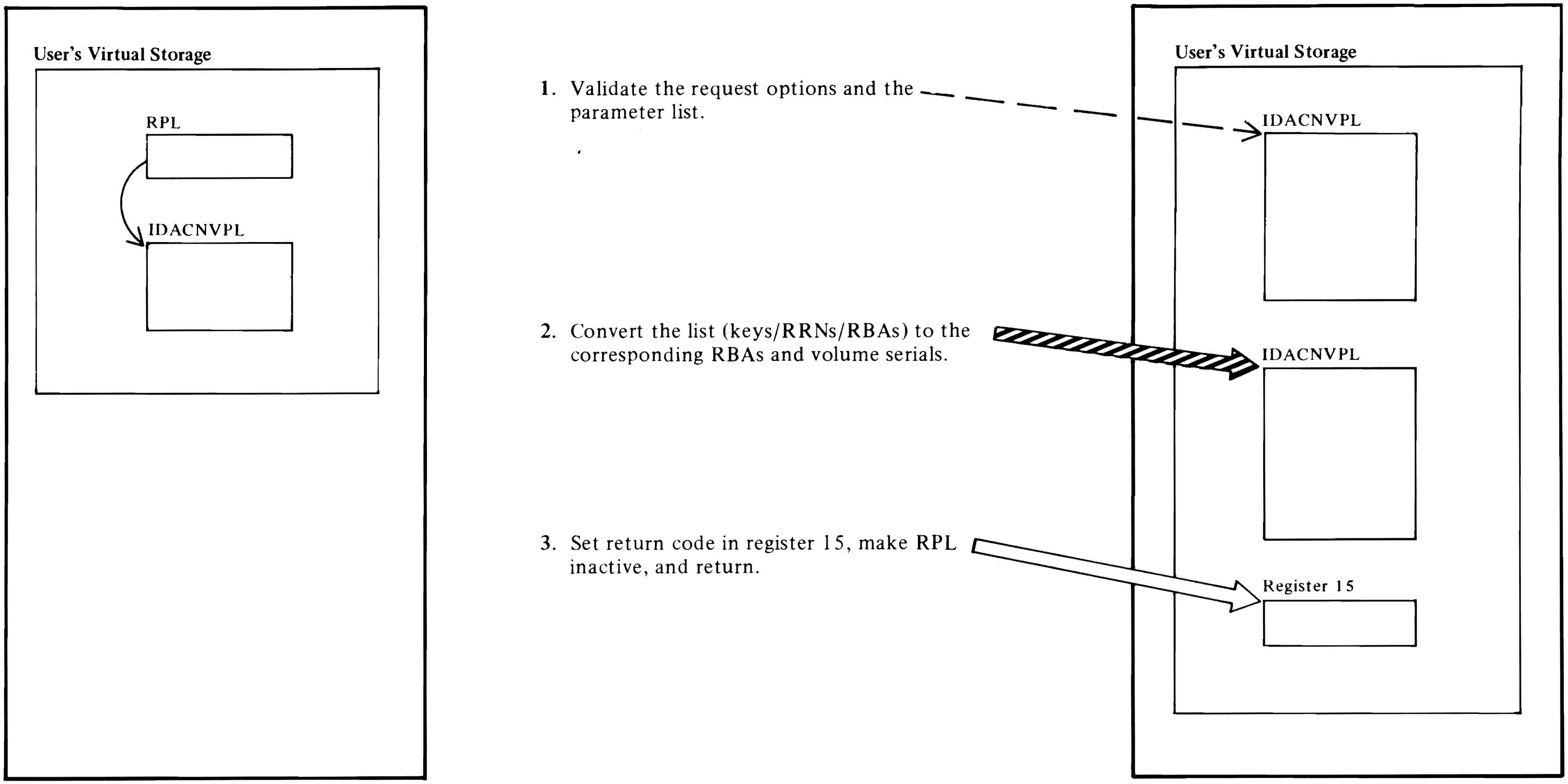
IDA019SN

The placeholder is disconnected and any positioning information associated with this RPL is lost.

3 **IDA019R1: TERMRPL**

Return to the caller with completion code set in Register 15 and RPLFDBK.

Diagram BX1. CNVTAD: Converting Key/RRN/RBA to Volume Serial and RBA



1. Validate the request options and the parameter list.

2. Convert the list (keys/RRNs/RBAs) to the corresponding RBAs and volume serials.

3. Set return code in register 15, make RPL inactive, and return.

Notes for Diagram BX1

1 **IDA019R1** calls **IDA019SG**

If the request is CNVTAD, call IDA019SG.

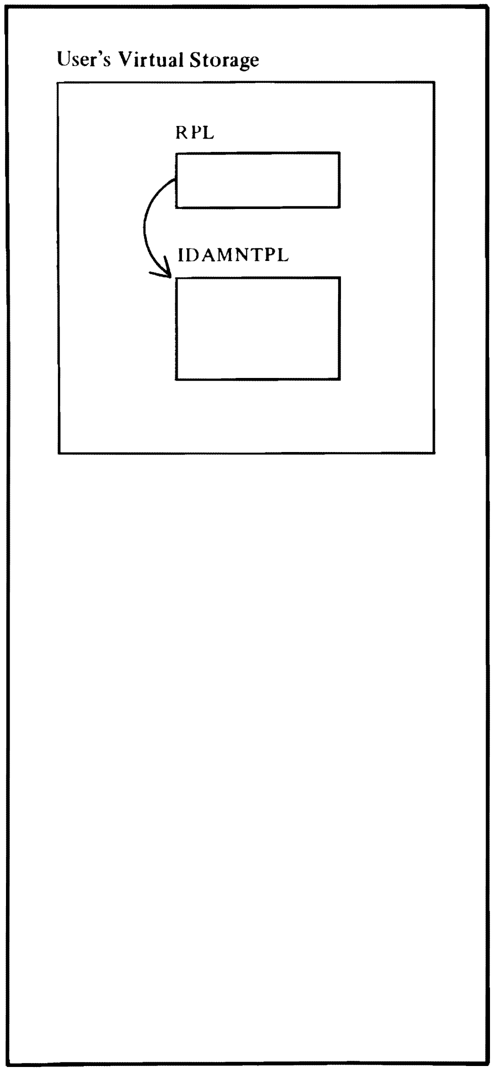
2 **IDA019SG** calls **IDA019RB, IGX00006**

Each IDACNVPL argument is converted to an RBA value.

- For ESDS, the RBA is validity checked.
- For RRDS, the RBA is calculated from the RRN.
- For KSDS, an index search parameter list (IDAIXSPL) is built, and IDA019RB is called to search the index.

A volume serial is obtained for each RBA by calling IGX00006, which issues SVC 26 (CATLG macro) to 'locate' the volume serial. See *OS/VS2 MVS Mass Storage System Extensions: Communicator (MSSC),* LY35-0038, for a description of IGX00006.

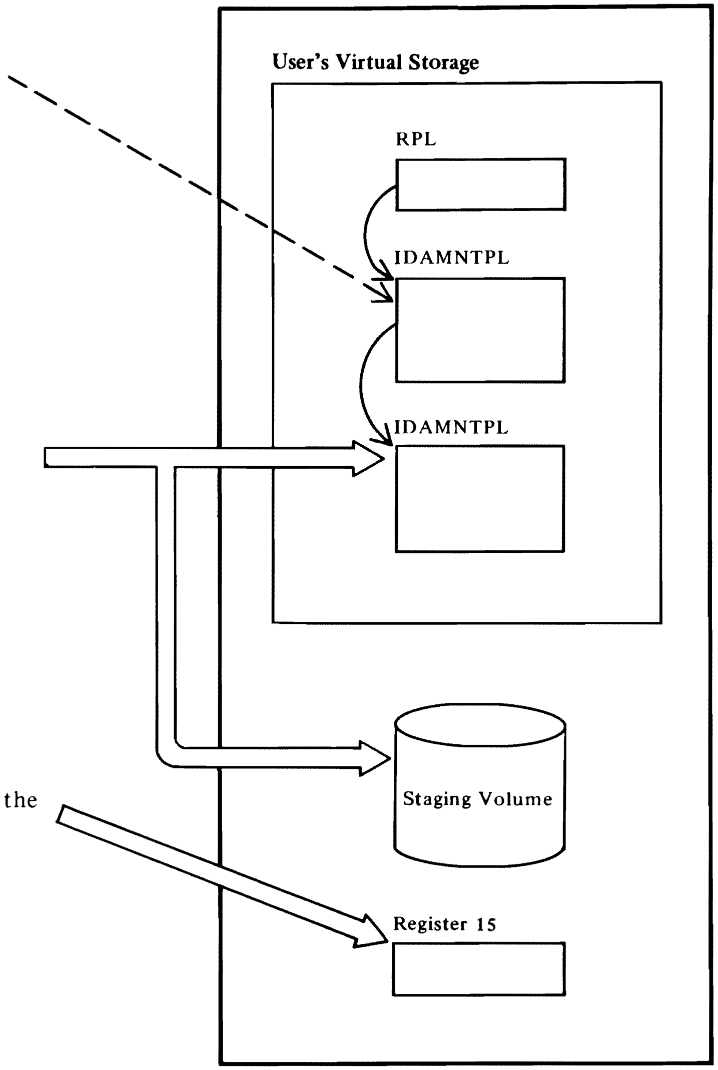
Diagram BX2. MNTACQ: Mounting a Volume and Staging VSAM Records onto It



1. Validate the request options and the parameter list.

2. The volume is mounted and the data cylinders corresponding to the RBAs are acquired.

3. A return code is set in register 15, and the RPL made inactive.



Notes for Diagram BX2

1 **IDA019R1** calls **IDA019SL**

If the request is MNTACQ, IDA019SL is called.

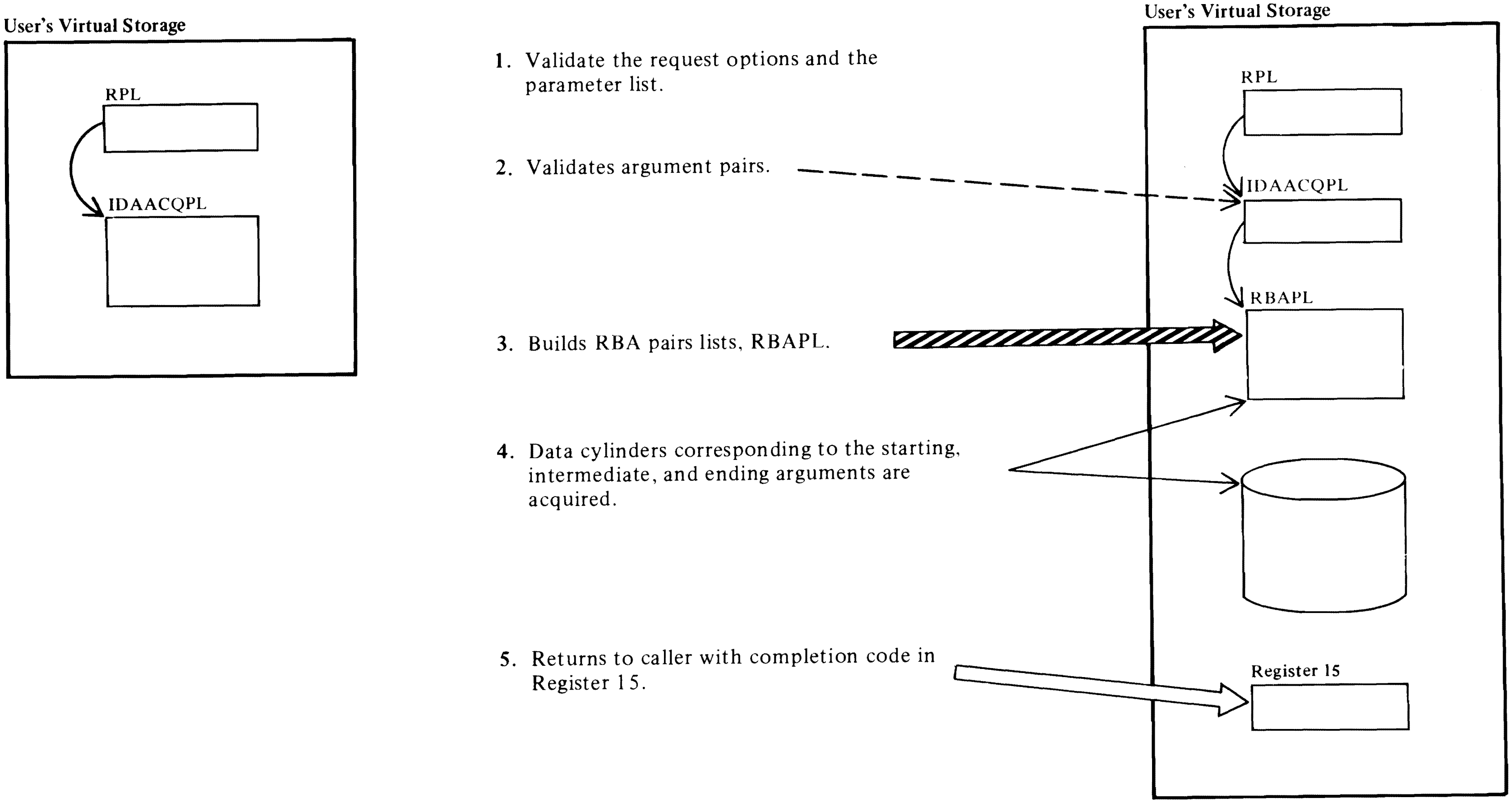
2 **IDA019SL** calls **IDAEOVIF**

IDAEOVIF is called to mount the volume.

IDA019SL calls **IDAMSSIF**

IDAMSSIF is called to issue SVC 109 route code 6 to acquire the data cylinders corresponding to the RBAs.

Diagram BX3. ACQRANGE: Staging a Range of Data from a VSAM Data Set.



1. Validate the request options and the parameter list.

2. Validates argument pairs.

3. Builds RBA pairs lists, RBAPL.

4. Data cylinders corresponding to the starting, intermediate, and ending arguments are acquired.

5. Returns to caller with completion code in Register 15.

Notes for Diagram BX3

1 **IDA019R1** calls **IDA019SH**

If the request is ACQRANGE, IDA019SH is called.

2 **IDA019SH** calls **KSDSPROC** or **BLDRBAPL.**

3 • If it is a KSDS, IDA019SH calls KSDSPROC.

KSDSPROC) calls **IDA019RB**

The starting key index record is retrieved. If it is an IMBED data set, KSDSPROC calls IMBEDDS to retrieve the ending key; otherwise, NONIMBED is called. The RBA pair is built.

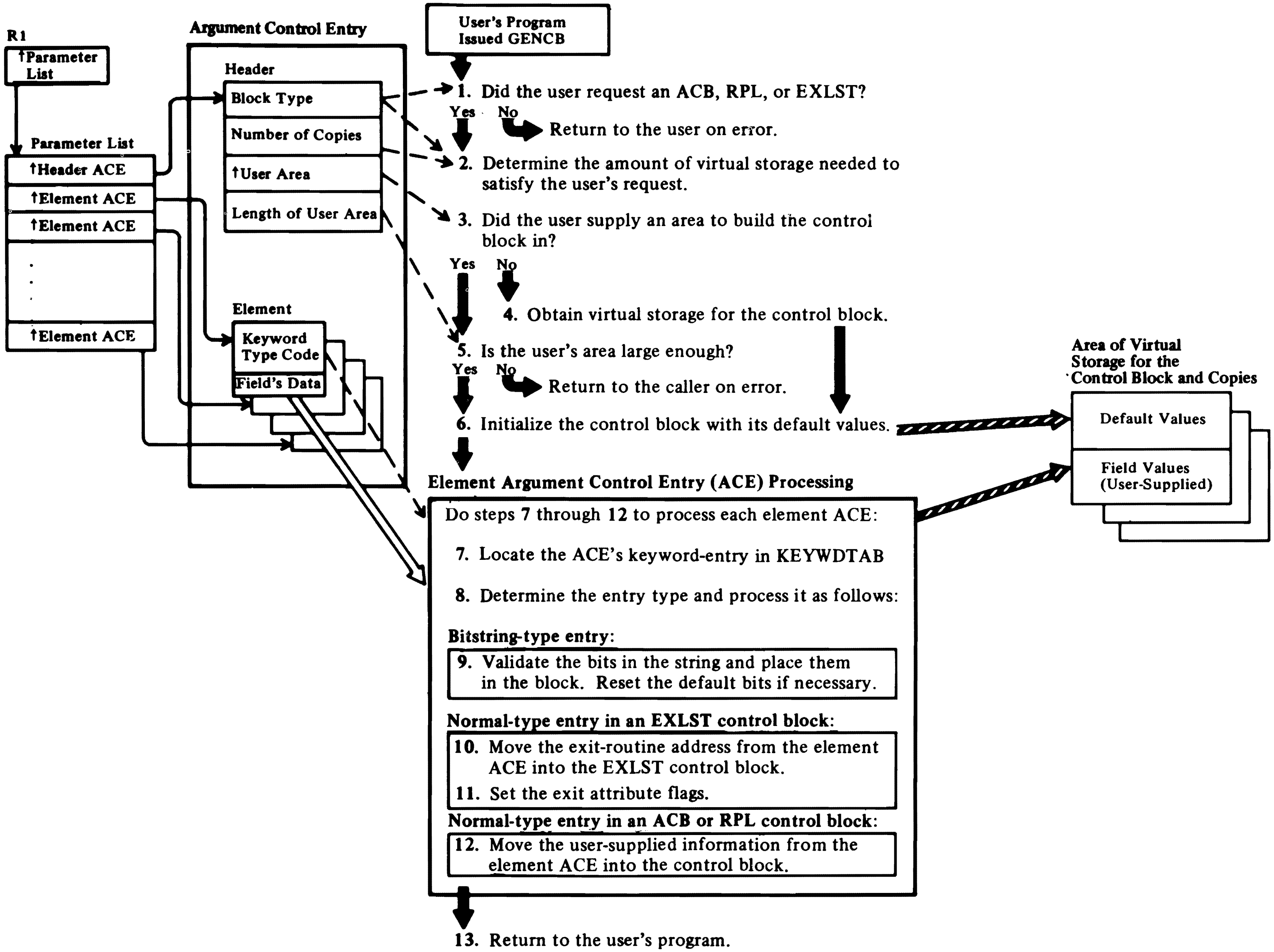
• If it is an RRDS or ESDS, IDA019SH calls BLDRBAPL.

BLDRBAPL converts RRNs to RBAs and builds the RBA pair list for RRDSs and ESDSs.

4 **IDA019SH** calls **IDAMSSIF**

IDAMSSIF is called to issue SVC 109 route code 6 to acquire the data cylinders corresponding to the RBAs.

Diagram CA. GENCB: Build a New Control Block



1. Did the user request an ACB, RPL, or EXLST?
   Yes No — Return to the user on error.
2. Determine the amount of virtual storage needed to satisfy the user's request.
3. Did the user supply an area to build the control block in?
   Yes No
4. Obtain virtual storage for the control block.
5. Is the user's area large enough?
   Yes No — Return to the caller on error.
6. Initialize the control block with its default values.

Element Argument Control Entry (ACE) Processing

Do steps 7 through 12 to process each element ACE:

7. Locate the ACE's keyword-entry in KEYWDTAB
8. Determine the entry type and process it as follows:

Bitstring-type entry:

9. Validate the bits in the string and place them in the block. Reset the default bits if necessary.

Normal-type entry in an EXLST control block:

10. Move the exit-routine address from the element ACE into the EXLST control block.
11. Set the exit attribute flags.

Normal-type entry in an ACB or RPL control block:

12. Move the user-supplied information from the element ACE into the control block.

13. Return to the user's program.

Record-Management Compendiums (Including End of Volume)

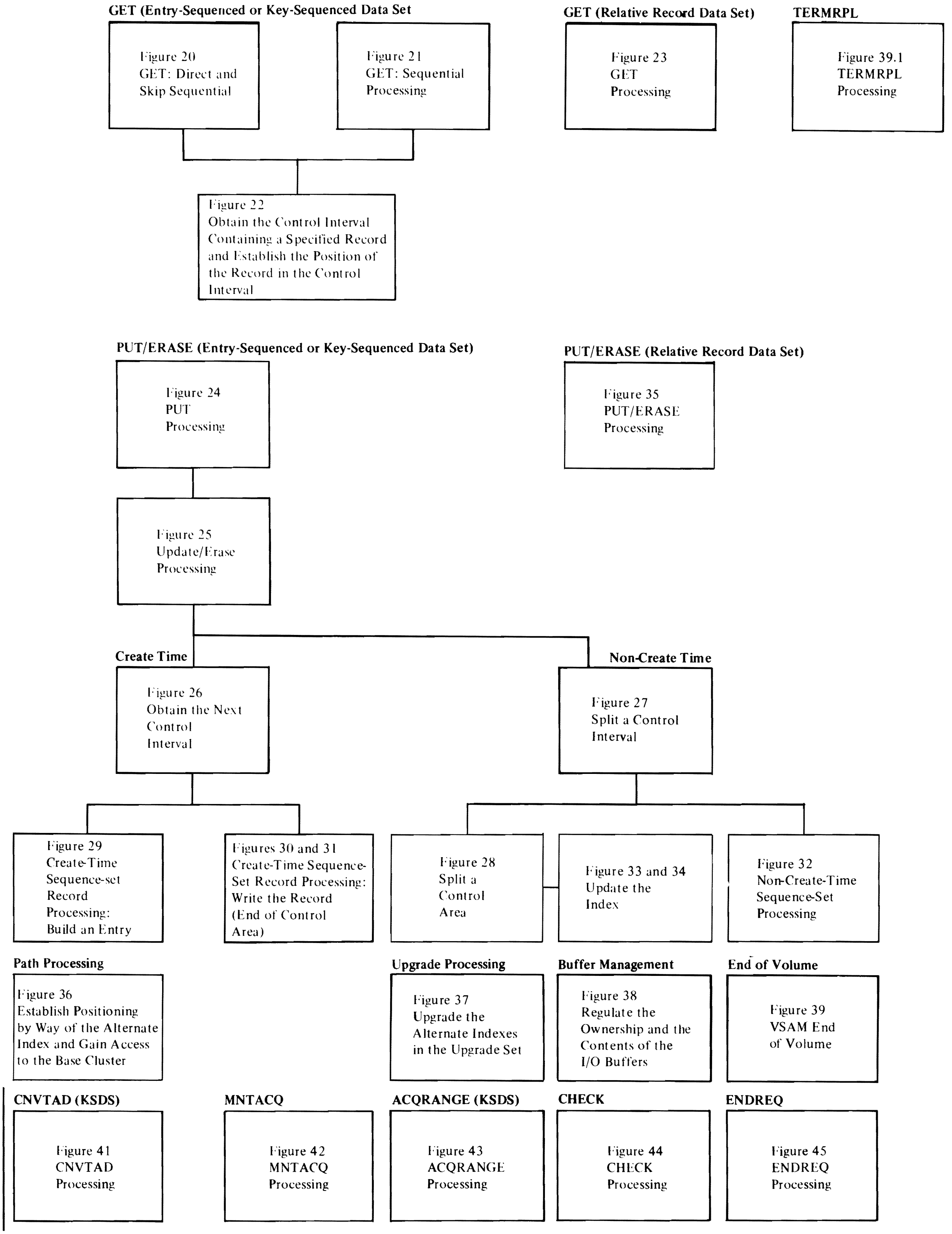

GET (Entry-Sequenced or Key-Sequenced Data Set
Figure 20 GET: Direct and Skip Sequential
Figure 21 GET: Sequential Processing
Figure 22 Obtain the Control Interval Containing a Specified Record and Establish the Position of the Record in the Control Interval
GET (Relative Record Data Set)
Figure 23 GET Processing
TERMRPL
Figure 39.1 TERMRPL Processing
PUT/ERASE (Entry-Sequenced or Key-Sequenced Data Set)
Figure 24 PUT Processing
Figure 25 Update/Erase Processing
PUT/ERASE (Relative Record Data Set)
Figure 35 PUT/ERASE Processing
Create Time
Figure 26 Obtain the Next Control Interval
Non-Create Time
Figure 27 Split a Control Interval
Figure 29 Create-Time Sequence-set Record Processing: Build an Entry
Figures 30 and 31 Create-Time Sequence-Set Record Processing: Write the Record (End of Control Area)
Figure 28 Split a Control Area
Figure 33 and 34 Update the Index
Figure 32 Non-Create-Time Sequence-Set Processing
Path Processing
Figure 36 Establish Positioning by Way of the Alternate Index and Gain Access to the Base Cluster
Upgrade Processing
Figure 37 Upgrade the Alternate Indexes in the Upgrade Set
Buffer Management
Figure 38 Regulate the Ownership and the Contents of the I/O Buffers
End of Volume
Figure 39 VSAM End of Volume
CNVTAD (KSDS)
Figure 41 CNVTAD Processing
MNTACQ
Figure 42 MNTACQ Processing
ACQRANGE (KSDS)
Figure 43 ACQRANGE Processing
CHECK
Figure 44 CHECK Processing
ENDREQ
Figure 45 ENDREQ Processing

Figure 19. Record-Management Program Organization Contents

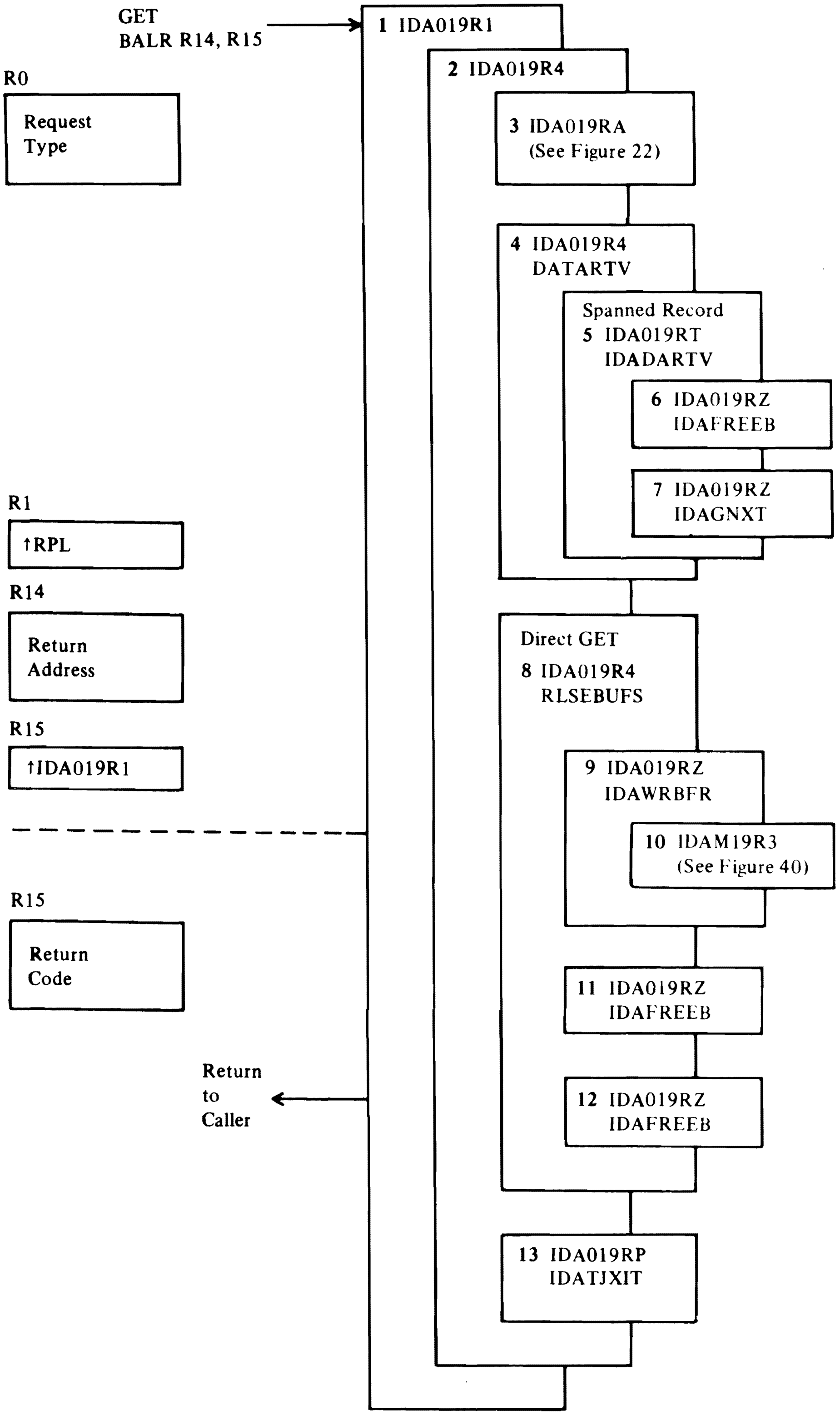


Figure 20. GET: Direct and Skip Sequential Processing (ESDS, KSDS)

Notes for Figure 39.1

1 IDA019R1 receives control from the TERMRPL macro.

2 IDA019SN releases all commonly shared VSAM resources owned by the terminated RPL.

3 IDAFREEB frees the index buffer (if the RPL is for a KSDS).

4 IDASBF subtracts excess data buffers.

5 IDARSTRT sttempts to restart all deferred synchronous requests not in the terminated address space.

6 IDA019SN disconnects the PLH.

7 IDA019R1 sets a condition code in register 15 and returns to caller.

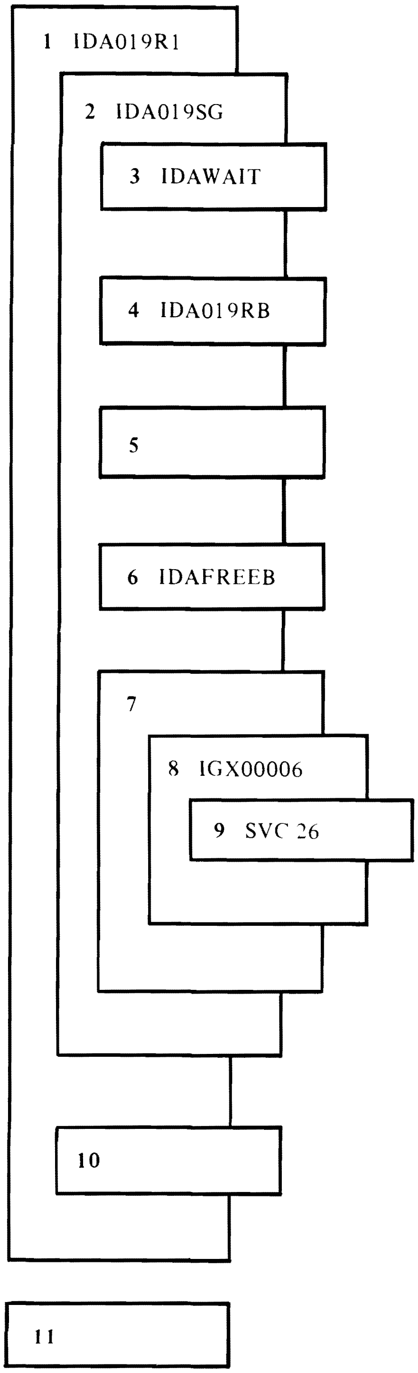


Figure 39.2 CNVTAD Processing—Converts Key/RBA/RRN to Volume Serial and RBA

Notes for Figure 39.2

1 When the CNVTAD macro is invoked, IDA019R1 is called to validate request options and parameter list.

2 IDA019R1 calls IDA019SG to convert the parameter list (IDACNVPL) arguments to RBAs and volume serials.

3 IDA019SG WAITs for previous I/O to be completed.

4 IDA019RB is called for index search on keyed requests.

5 The index record is searched to obtain an RBA for the BASE DATA RECORD.

6 IDA019SG frees the buffer IDA019RB gotten for the index search.

7 For an ESDS, the RBA is returned; for an RRDS, the RBA is calculated.

8 IGX00006 (SVC109) is called to obtain volume serials for each RBA.

9 IGX00006 issues SVC 26 to 'LOCATE' the volume serials.

10 IDA019SG returns to IDA019R1.

11 IDA019R1 returns to the user.

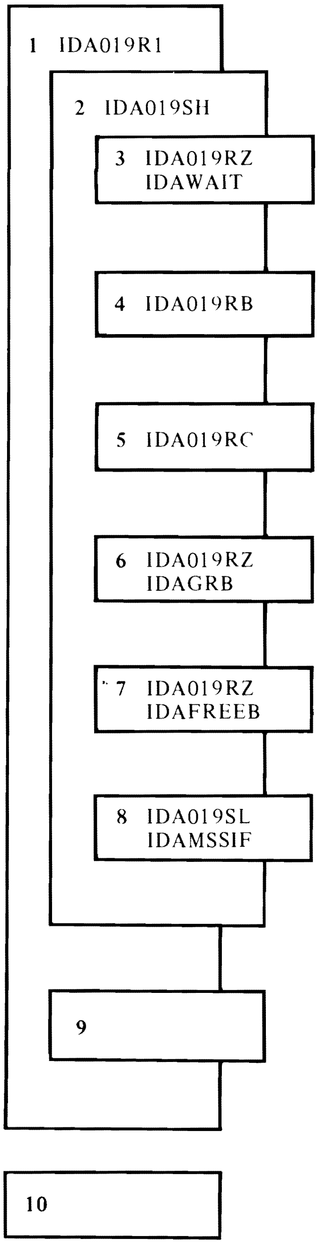


Figure 39.3 ACQRANGE Processing—Stages a Range of Data From a VSAM Data Set

Notes for Figure 39.3

1 When the ACQRANGE macro is invoked, IDA019R1 is called to validate request options and parameter list.

2 IDA019R1 calls IDA019SH to build an RBA pairs list (IDARBAPL) consisting of keys/RBAs/RRNs describing ranges of data.

3 IDA019RZ is called to WAIT for previous I/O completion.

4 IDA019RB is called to retrieve the starting key.

5 IDA019RC is called to retrieve the ending key.

6 IDA019RZ is called to get next index record, if necessary.

7 IDA019RZ is called to free the buffers.

8 IDAMSSIF is called to issue SVC 109 routing code 6 to acquire the data cylinders corresponding to the pairs of RBAs.

9 When IDA019SH regains control after the acquire, IDA019SH returns the IDAACQPL, ECB WAIT list pointer, RPERREG, and RPLERRCD to IDA019R1.

10 IDA019R1 WAITs or returns to user.

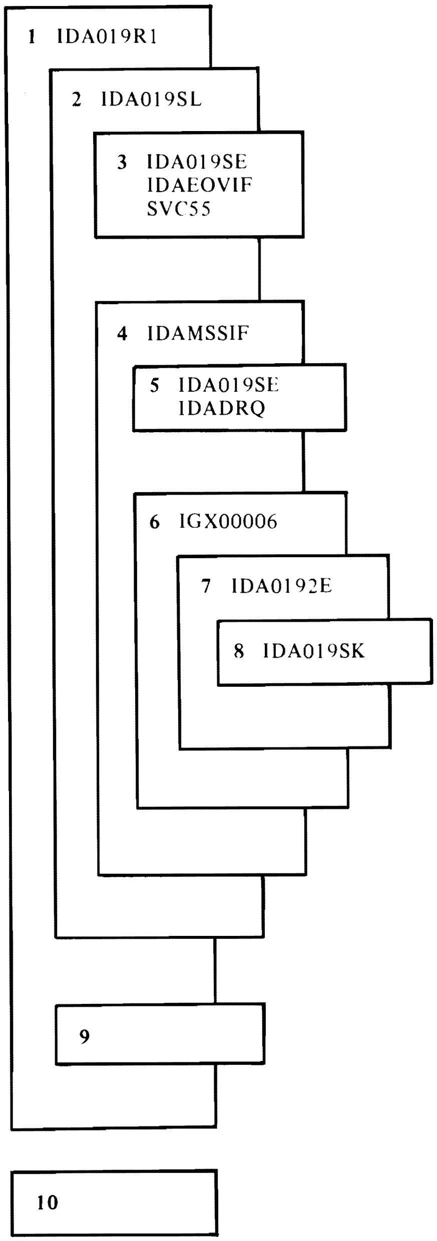


Figure 39.4 MNTACQ Processing—Mounts a Volume and Stages VSAM Records into It

Notes for Figure 39.4

1 When the MNTACQ macro is invoked, IDA019R1 is called to validate request options and parameter list.

2 IDA019R1 calls IDA019SL to mount the volume indicated by the parameter list.

3 IDA019SL calls IDAEOVIF to mount the volumes.

4 IDA019SL calls IDAMSSIF to issue SVC 109 to acquire the data cylinders corresponding to the pairs of RBAs.

IDAMSSIF locks the AMBXs to prevent EOV from running during the acquire.

5 IDA0192E is called to WAIT for previous I/O completion.

6 IGX00006 (SVC 109 routing code 6) is called and retrieves the volume serial numbers of the volumes specified by the RBA pairs.

7 IGX00006 calls IDA0192E to acquire the data cylinders by converting RBAs or RBA pairs to cylinder extents and to build an acquire parameter list and ECB.

8 IDA0192E calls IDA019SK to sort cylinder extents into ascending sequence. If it is not a virtual device, the WAIT list is posted with successful completion and return to caller. If it is a virtual device, SVC 126 is called for acquire with deferred response.

9 IDA0192E returns and passes the ECB WAIT list to IDA019SL.

10 IDA019SL returns to IDA019R1 to WAIT or return to user. See *OS/VS2 MVS Mass Storage System Extensions: Communicator (MSCC),* for a description of IHX00006, IDA0192E, and IDA019SK.

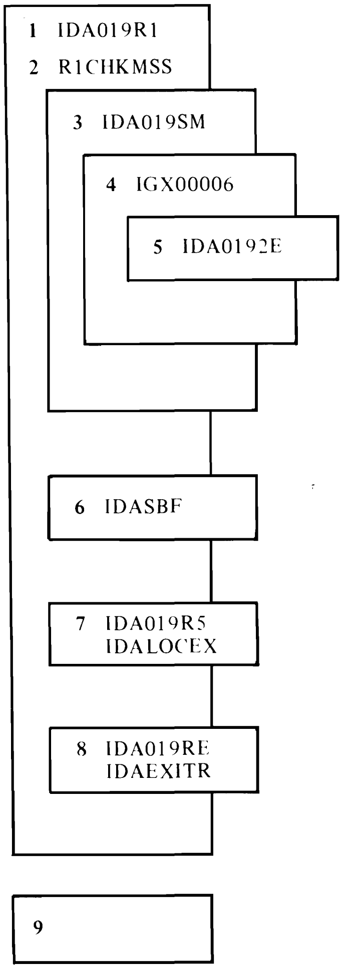


Figure 39.5 CHECK Proecssing —WAITs and POSTs ECBs

Notes for Figure 39.5

1 When the CHECK macro is invoked, IDA019R1 is called to validate request options and parameter list.

If the previous request was CNVTAD, MNTACQ, and ACQRANGE, control passes to R1CHKMSS; otherwise, continues a normal CHECK with R1CHECK.

2 R1CHKMSS provides CHECK function and internal synchronization for requests which specify SYN.

3 R1CHKMSS calls IDA019SM to do WAIT processing. IDA019SM WAITs on ECBs if any are not posted complete, sets RPLERREG/RPLERRCD from the ECB completion code, and releases the ECBs and WAIT list.

4 IDA019SM calls IGX00006 to free ECBs.

5 IGX00006 calls IDA0192E to process FREEMAIN request. FREEMAIN is issued for groups of ECBs if all are posted complete.

6-8 R1CHKMSS provides R1CHECK function for CNVTAD, MNTACQ, and ACQRANGE. It calls IDASBF to subtract excess buffers and IDALOCEX to find error exits and IDAEXITR to take the exit.

9 IDA019R1 returns to user.

See *OS/VS2 MVS MSS Storage System Extensions: Communicator (MSCC),* for a description of IGX00006 and IDA0192E.

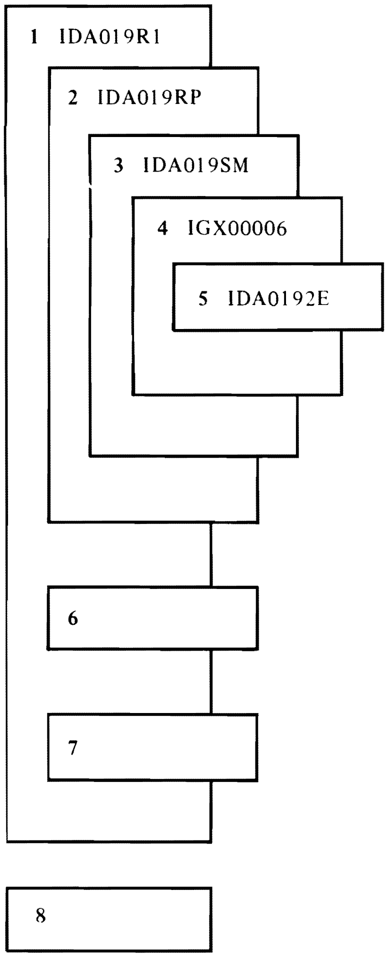


Figure 39.6 ENDREQ Processing—WAITs, POSTs, Frees ECBs

Notes for Figure 39.6

1 When the ENDREQ macro is invoked, IDA019R1 is called to validate request options and parameter list.

2 IDA019R1 calls IDA019RP, the ENDREQ processing module.

If previous request was MNTACQ or ACQRANGE, IDA019RP calls IDA019SM to WAIT for I/O completion.

3 IDA019SM WAITs on the ECBs; if any are not posted, sets RPLERREG/RPLERRCD from the ECB post code.

4 IDA019SM calls IGX00006 to free ECBs.

5 IGY00006 calls IDA0192E to process FREEMAIN request. FREEMAIN is issued for groups of ECBs if all are posted complete.

6 ENDREQ processing for buffer flushing and error exits continues on return to IDA019RP from IDA019SM.

7 IDA019RP returns control to IDA019R1.

8 IDA019RP returns control to user. See *OS/VS2 MVS Mass Storage System Extensions: Communicator (MSSC)*, for a description of IGX00006 and IDA0192E.

I/O-Management Compendiums

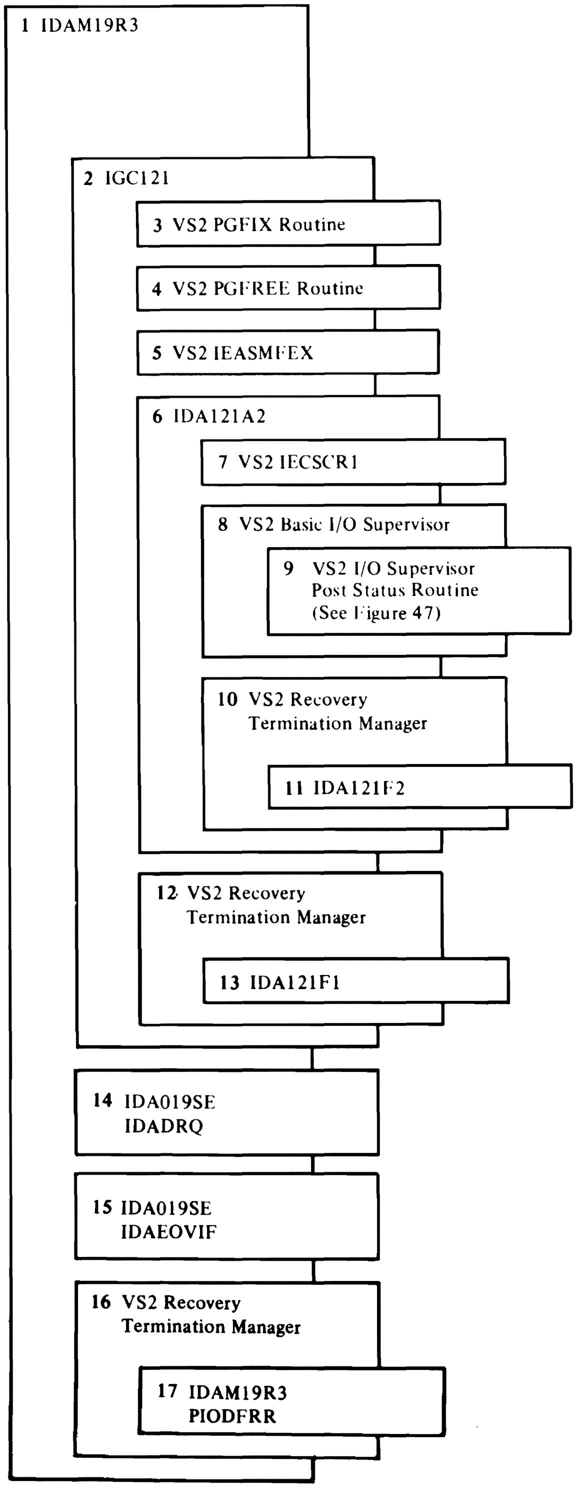

1 IDAM19R3
2 IGC121
3 VS2 PGFIX Routine
4 VS2 PGFREE Routine
5 VS2 IEASMFEX
6 IDA121A2
7 VS2 IECSCR1
8 VS2 Basic I/O Supervisor
9 VS2 I/O Supervisor Post Status Routine (See Figure 47)
10 VS2 Recovery Termination Manager
11 IDA121F2
12 VS2 Recovery Termination Manager
13 IDA121F1
14 IDA019SE IDADRQ
15 IDA019SE IDAEOVIF
16 VS2 Recovery Termination Manager
17 IDAM19R3 PIODFRR

Figure 40. I/O Management: Translating Virtual Addresses to Real Addresses and Completing a Channel Program for I/O

Module Directory

Module Name	Descriptive Name	External Procedure Names	Component	Method of Operation Diagrams	Program Organization Figures
		IDADRQ	RM	BP1, BP3, BS2, DA1	24, 35, 38, 40, 43, 46
		IDAEOVIF	RM	BE, BG2, BN2, BO1, DA1, DA2, DA4	26, 28, 29, 34, 40, 43
		IDARSTRT	RM	BW	39.1
IDA019SF	Control-Area Split-Spanned Records	IDA019SF	RM	BH4	28
IDA019SG	Convert Keys/RRNs to RBAs	IDA019SG	RM	BX1	39.2
IDA019SH	Acquire a Range of Keys/RBAs/RRNs	IDA019SH	RM	BX3	39.3
IDA019SK	Sort Acquire List Extents	IDA019SK	—	—	39.4
IDA019SL	Mount Volume and Acquire	IDA019SL	RM	BX2	39.4
IDA019SM	Check and ENDREQ Support for MNTACQ/ACQRANGE Request	IDA019SM	RM	—	39.5, 39.6
IDA019SN	Terminate RPL	IDA019SN	RM	BW	39.1
IDA019S1	Improved Control-Interval Processing Driver	IDA019S1	RM	BN1,BN3	—
IDA019S3	Improved Control-Interval Processing—I/O Management	IDA019S3	RM	BN1, BN3	—
IDA019S6	Control Interval Rebuild	IDA019S6	RM	BC,BD,BH3	21,22
IDA0192A	VSAM Open String	IDA0192A	O	AC1, AC2, AC7, AG	5, 6, 7, 11
IDA0192B	Open a Cluster	IDA0192B	O	AC4, AL1	8, 18
IDA0192C	Catalog Interface	IDA0192C	O	AC2, AC4, AD6 AD7, AE2, AL1, BT1	5, 6, 8, 12, 14, 15, 18, 39
IDA0192D	Stage/Destage (ACQUIRE/RELINQUISH)	IDA0192D	O/C/EOV	AD6, AE2, BT2	8, 12, 14, 18, 39
IDA0192E	Stage by RBA or RBA Range (ACQUIRE)	IDA0192E	—	—	39.4 - 39.6
IDA0192F	Open Base Cluster, Path, and Upgrade Alternate Index	IDA0192F	O	AC3, AC4, AC5, AC6	8,18
IDA0192G	Data-Space Security Verification	IDA0192G	O	—	15
IDA0192I	ISAM Interface: Open Processing	IDA0192I	II	AC1, AC7	7
IDA0192M	Virtual-Storage Manager	IDA0192M	O/C/EOV	AL2	8, 10, 16
IDA0192P	VSAM Open/Close/EOV: Problem Determination	IDA0192P	O	AD5, AD6, AE2, AH3	8, 12, 14, 18, 39
IDA0192S	VSAM Open/Close/EOV: SMF Record Build	IDA0192S	O	AC7, AD6, AE2	8, 12, 14, 39
IDA0192V	Volume Mount and Verify	IDA0192V	O	AC3, BT1	5, 8,18, 39
IDA0192W	Channel-Program-Area Build	IDA0192W	O	AC1, AC4	8, 10
IDA0192Y	String Build and Shared-Resource Processor	IDA0192Y	O/C	AC1, AC4, AC5, AD5, AF1, AH3, AL1, AL2	8, 10, 12, 14, 16, 18
IDA0192Z	Control-Block Build	IDA0192Z	O	AC4, AL1	8, 18
IDA0195A	VSAM SNAP Format Routine	IDA0195A	O/C/EOV	—	—
IDA0200B	Close a Cluster	IDA0200B	C	AD1, AD3, AD4, AD6	12

Module Directory

Module Name	Descriptive Name	External Procedure Names	Component	Method of Operation Diagrams	Program Organization Figures
IDA0200S	ISAM Interface: Close Processing	IDA0200S	II	AD1, AD7	11
IDA0200T	VSAM Close String	IDA0200T	C	AD1, AD2, AD3, AD4, AD5, AD7	12
IDA0231B	Close (TYPE=T) a Cluster	IDA0231B	C	AE1, AE2	14
IDA0231T	VSAM Close (TYPE=T) String	IDA0231T	C	AE1, AE2	14
IDA0557A	VSAM End of Volume	IDA0557A	EOV	BT1, BT2	—
IDA121A2	Actual Block Processor	IDA121A2	IOM	DA2, DA3, DA4	40
		IDA121F2	IOM	DA3	40
IDA121A3	Normal End Appendage	IDA121A3	IOM	DA4	41
		F3FRR	IOM	DA4	41
		IDA121F3	IOM	DA4	41
IDA121A4	Abnormal End Appendage	IDA121A4	IOM	DA5	41
		IDA121F4	IOM	DA5	41
IDA121A5	Asynchronous Routine	IDA121A5	IOM	DA4, DA5	41
IDA121A6	Purge Routine	IDA121A6	IOM	—	—
IDA121CV	Communication Vector Table (IEZABP)	—	IOM	—	—
IEFVAMP	AMP Parameter Interpreter	IEFNB902	—	—	—
IFG0192A	VSAM Open/Close/EOV String Load (Interface between OS/VS and VSAM O/C/EOV)	IFG0192A	O/C/EOV	AF	5, 7, 8, 11, 12, 14
IFG0192Y	BLDVRP/DLVRP Load Routine	IFG0192Y	O/C/EOV	—	10, 16
IGC121	Supervisor-State I/O Driver	IGC121	IOM	DA1, DA2, DA3	40
		IDA121F1	IOM	DA2	40
IGX00006	VSAM MSS Support SVC	IGX00006	—	—	39.2, 39.4 39.5, 39.6

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
IDAWRBFR	IDA019RZ	Buffer Management: Write the Buffer	BE, BG2, BH3, BH4, BH5, BH8, BH9, BK1, BK2, BN2, BN3, BO1, BO2	20, 24, 25, 26, 27, 28, 32, 35, 38
IDAWRTBF	IDA019RZ	Write a Buffer	—	—
IDAXGPLH	IDA019RU	Get a Placeholder	BB1	—
IDA0A05B	IDA0A05B	Checkpoint/Restart: Restart	AJ, AL1, AL2	18
IDA0C05B	IDA0C05B	Checkpoint/Restart: SSCR and Initial DEB Processing	AK	18
IDA0C06C	IDA0C06C	Checkpoint/Restart: Checkpoint	AI, AJ	17
IDA0I96C	IDA0I96C	Checkpoint/Restart: SSCR Build and Cleanup	AJ	17
IDA019C1	IDA019C1	Control Block Manipulation	CA, CB1, CB2	—
IDA019RA	IDA019RA	Direct Record Locate	BC, BE, BH1, BJ	20, 21, 22, 24
IDA019RB	IDA019RB	Index Search	BC, BH1, BH8, BJ, BM	22, 34
IDA019RC	IDA019RC	Search Compressed Index Block	BC, BH1, BH2, BH6, BI, BJ, BM, BS4	22, 24, 25, 32
IDA019RD	IDA019RD	DD DUMMY Processing	—	—
IDA019RE	IDA019RE	Control-Interval Split	BH1, BH3, BH7	24, 25, 27, 28, 32
IDA019RF	IDA019RF	Control-Area Split	BH1, BH2, BH3, BH4, BH5	24, 25, 26, 27, 33, 34
IDA019RG	IDA019RG	Index Create	BG1, BG2, BG3, BG4, BG5, BK2	26, 29, 30, 31
IDA019RH	IDA019RH	Index Insert	BH3, BH6, BH7, BH8	27, 32, 33, 34
IDA019RI	IDA019RI	Index Upgrade	BH4, BH5, BH7, BH8, BH9	28, 33, 34
IDA019RJ	IDA019RJ	Split Index Record	BH4, BH7, BH8, BH9, BH10	34
IDA019RK	IDA019RK	Preformat	BG2, BH4, BH8, BH9, BK2, BN2, BO1	24, 26, 28, 29
IDA019RL	IDA019RL	Data Modify	BH2, BI	24, 25
IDA019RM	IDA019RM	Data Insert	BE, BF, BH1, BH2, BH3	24, 25, 26, 27, 28, 40
IDA019RN	IDA019RN	Indexing Subroutines	—	—
IDA019RO	IDA019RO	Verify	BM	—
IDA019RP	IDA019RP	ENDREQ and JRNAD	BK1, BK2	—
IDA019RQ	IDA019RQ	Relative Record Subroutines	BO1, BO2	35
IDA019RR	IDA019RR	Relative Record Driver	BC, BD, BJ, BO1	23, 25, 35
IDA019RS	IDA019RS	Spanned Record Data Modify	BH2, BI	24, 25
IDA019RT	IDA019RT	Spanned Record Data Insert	BE, BF, BH1	24

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
IDA019RU	IDA019RU	Alternate-Index Upgrade Driver	BC, BD, BE, BH1, BI, BR	37
IDA019RV	IDA019RV	Locate Previous Sequence-Set Record	—	28
IDA019RW	IDA019RW	Buffer Management, Part 2	—	38
IDA019RX	IDA019RX	Path Processing Driver	BQ	36
IDA019RY	IDA019RY	Shared Resources Buffer Management	BP1, BP2, BP3, BS1, BS2, DA1	38
IDA019RZ	IDA019RZ	Buffer Management Interface	BC, BS1, BS2, BS4	38
IDA019R1	IDA019R1	Record Management: Request Decode and Validate	AD7, BB1, BK1, BK2, BL	12, 14, 20, 21, 23, 24, 35, 36
IDA019R2	IDA019R2	Buffer Management, Part 1	BS1, BS2, BS3, DA1	38
IDA019R3	IDAM19R3	I/O Management: Problem-State I/O Driver	BG1, BS1, BS2, BS3, BS4, DA1, DA2, DA3	20, 22, 38, 40
IDA019R4	IDA019R4	Keyed/Addressed Request Driver	BC, BD, BE, BF, BH1, BH3, BQ, BR	20, 21, 22, 24, 25, 36, 37
IDA019R5	IDA019R5	I/O Error Analyais	BB3, BK1, BL	38
IDA019R8	IDA019R8	Control-Interval Processing	BM, BN1, BN2, BN3	—
IDA019SA	IDA019SA	Control-Interval Initialization-Create Entry-Sequenced Data Set	BE, BF, BG1, BG2, BG3, BK2	26, 29, 30
IDA019SB	IDA019SB	Dynamically Build Channel Program Area for Shared Resources	—	—
IDA019SE	IDA019SE	I/O Error Service Routines	BB3, BE, BO1, BP1, BP3, BS2, BV, DA1, DA2, DA4	25, 26, 28, 29, 34, 35, 38, 39, 39.1, 40
IDA019SF	IDA019SF	Control-Area Split-Spanned Records	BH4	28
IDA019SG	IDA019SG	Convert Keys/RRNs/RBAs to RBAs	BX1	39.2
IDA019SH	IDA019SH	Acquire a Range of Keys/RBAs/RRNs	BX3	39.3
IDA019SK	IDA019SK	Sort ACQUIRE LIST Extents	—	39.4
IDA019SL	IDA019SL	Mount Volume and Acquire	BX2	39.4
IDA019SM	IDA019SM	CHECK and ENDREQ Support for MNTACQ and ACQRANGE Request	—	39.5, 39.6
IDA019SN	IDA019SN	Terminate RPL	BW	39.1
IDA019S1	IDA019S1	Improved Control-Interval Processing Driver	BN1, BN3	—
IDA019S3	IDA019S3	Improved Control-Interval Processing-I/O Management	BN1, BN3	—
IDA019S6	IDA019S6	Control Interval Rebuild	BC,BD,BH321,22	
IDA0192A	IDA0192A	VSAM Open String	AC1, AC2, AC7, AG	5, 6, 7, 11
IDA0192B	IDA0192B	Open a Cluster	AC4, AL1	8, 18

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
IDA0192C	IDA0192C	VSAM Open/Close: Catalog InterfaceAC2, AC4, AD6,	AD7, AE2, AL1, BT1	5, 6, 8, 12, 14, 15, 18, 39
IDA0192D	IDA0192D	Stage/Destage (ACQUIRE/RELINQUISH)	AD6, AE2, BT2	8, 12, 14, 18, 39
IDA0192E	IDA0192E	Stage by RBA or RBA Range (ACQUIRE)	—	39.4 - 39.6
IDA0192F	IDA0192F	Open Base Cluster, Path, and Upgrade Alternate Index	AC3, AC4, AC5, AC6	8, 18
IDA0192G	IDA0192G	Data-Space Security Verification	AL2	15
IDA0192I	IDA0192I	ISAM Interface: Open Processing	AC1, AC7	7
IDA0192M	IDA0192M	Virtual Storage Manager	—	8, 10, 16
IDA0192P	IDA0192P	VSAM Open/Close/EOV: Problem Determination	AD5, AD6, AE2, AH3	8, 12, 14, 18, 39
IDA0192S	IDA0192S	VSAM Open/Close/EOV: SMF Record Build	AC7, AD6, AE2	8, 12, 14, 39
IDA0192V	IDA0192V	Volume Mount and Verify	AC3, BT1	5, 8, 18, 39
IDA0192W	IDA0192W	Channel-Program-Area Build	AC1, AC4	8, 10
IDA0192Y	IDA0192Y	String Build and Shared-Resource Processor	AC1, AC4, AC5, AD5, AF1, AH3, AL1, AL2	8, 10, 12, 14, 16, 18
IDA0192Z	IDA0192Z	Control Block Build	AC4, AL1	8, 18
IDA0200B	IDA0200B	Close a Cluster	AD1, AD3, AD4, AD6	12
IDA0200S	IDA0200S	ISAM Interface: Close Processing	AD1, AD7	11
IDA0200T	IDA0200T	VSAM Close String	AD1, AD2, AD3 AD4, AD5, AD7	12
IDA0231B	IDA0231B	Close (TYPE=T) a Cluster	AE1, AE2	14
IDA0231T	IDA0231T	VSAM Close (TYPE=T) String	AE1, AE2	14
IDA0557A	IDA0557A	VSAM End of Volume	BT1, BT2	—
IDA121A2	IDA121A2	I/O Management: Actual Block Processor	DA2, DA3, DA4	40
IDA121A3	IDA121A3	I/O Management: Normal End Appendage	DA4	41
IDA121A4	IDA121A4	I/O Management: Abnormal End appendage	DA5	41
IDA121A5	IDA121A5	I/O Management: Asynchronous Routine	DA4, DA5	41
IDA121A6	IDA121A6	I/O Management: Purge Routine	—	—
IDA121F1	IGC121	Functional Recovery Routine	DA2	40
IDA121F2	IDA121A2	Functional Recovery Routine	DA3	40
IDA121F3	IDA121A3	Functional Recovery Routine	DA4	41
IDA121F4	IDA121A4	Functional Recovery Routine	DA5	41
IEFNB902	IEFVAMP	AMP Parameter Interpreter	—	—
IFG0192A	IFG0192A	VSAM Open/Close/EOV String Load (Interface between OS/VS and VSAM O/C/EOV)	AF	5, 7, 8, 11, 12, 14
IFG0192Y	IFG0192Y	BLDVRP/DLVRP Load Routine	—	10, 16
IGC121	IGC121	I/O Management: Supervisor State I/O Driver	DA1, DA2, DA3	40
IGX00006	IGX00006	VSAM MSS Support SVC	—	39.2, 39.4 39.5, 39.6

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
IMDUSRF9	AMDUSRF9	IMDPRDMP Format Appendage	—	—
PIODFRR	IDAM19R3	Functional Recovery Routine	DA1	40

Module Packaging

Most VSAM modules reside in pageable virtual storage; some I/O-Management modules reside in the nucleus. The following table lists the VSAM load modules and transients that are resident in the SVCLIB or LPALIB library or in the nucleus. Those in the libraries are loaded into the pageable supervisor or link-pack area by nucleus initialization (NIP)′at initial program load (IPL). Those in the nucleus are link-edited there when the system is generated.

Name	Description	VSAM Modules
Record Management		
IDA019L1	Main Record Management	IDAM19R3, IDA019RA, IDA019RB, IDA019RC, IDA019RD, IDA019RE, IDA019RF, IDA019RG, IDA019RH, IDA019RI, IDA019RJ, IDA019RK, IDA019RL, IDA019RM, IDA019RN, IDA019RO, IDA019RP, IDA019RQ, IDA019RR, IDA019RU, IDA019RV, IDA019RW, IDA019RX, IDA019RY, IDA019RZ, IDA019R1, IDA019R2, IDA019R4, IDA019R5, IDA019R8, IDA019SA, IDA019SB, IDA019SF, IDA019SG, IDA019SH, IDA019SL, IDA019SM, IDA019SE, IDA019SE
IDA019L2	Improved Control-Interval Processing	IDA019S1, IDA019S2, IDA019S6
IDA019RS	Spanned Record Data Modify	IDA019RS
IDA019RT	Spanned Record Data Insert	IDA019RT
Open/Close/End of Volume and Checkpoint/Restart		
IDA0192A	Open/Close/End of Volume	IDACKRA1,IDA0A05B, IDA0C05B, IDA0C06C, IDA0I96C, IDA0192A, IDA0192B, IDA0192C, IDA0192D, IDA0192F, IDA0192G, IDA0192I, IDA0192M, IDA0192P, IDA0192S, IDA0192V, IDA0192W, IDA0192Y, IDA0192Z, IDA0200B, IDA0200S, IDA0200T, IDA0231B, IDA0231T, IDA0557A
ISAM Interface		
IDAIIFBF	FREEDBUF	IDAIIFBF
IDAIIPM1	QISAM Load	IDAIIPM1
IDAIIPM2	QISAM Scan	IDAIIPM2
IDAIIPM3	BISAM	IDAIIPM3
IDAIISM1	SYNAD	IDAIISM1

Name	Description	VSAM Modules
I/O Management		
IDAM19R3	Problem-State I/O Driver	IDAM19R3 (Packaged in Main Record Management IDA019L1)
IDA019SB	Dynamic Channel Program Area Build for Shared Resources	IDA019SB (Nucleus)
IDA121A2	Actual Block Processor	IDA121A2 (Nucleus)
IDA121A3	Normal End Appendage	IDA121A3 (Nucleus)
IDA121A4	Abnormal End Appendage	IDA121A4 (Nucleus)
IDA121A5	Asynchronous Routine	IDA121A5
IDA121A6	Purge Routine	IDA121A6 (Nucleus)
IDA121CV	Communication Vector Table (IEZABP)	IDA121CV (Nucleus)
IGC121	Supervisor-State I/O Driver	IGC121 (Nucleus)
Control Block Manipulation		
IDA019C1	Control Block Manipulation	IDA019C1
VSAM Transient Routine		
IFG0192A	VSAM Open/Close/End of Volume Loader	IDAICIA1, IDAOCEA1, IDAOCEA2, IDAOCEA4, IFG0192A
Miscellaneous Routines		
AMDUSRF9	IMDPRDMP format appendage	AMDUSRF9
IDA0195A	VSAM SNAP format routine	IDA0195A
IEFVAMP	AMP parameter interpreter	IEFVAMP

Control Block Subpool Assignment

Subpool 230 in the high end of the user's private address space contains the DEBs, to be consistent with OS/VS I/O Support and Checkpoint/Restart. Subpool 245 in the system queue area (SQA, in global storage) contains the IOSBs and SRBs because I/O Supervisor, running in any user's private address space, needs them in a place where it can always address them. Subpool 241 contains ECBs for deferred response posting by MSS ACQUIRE. Subpool 252 in the low end of the private area contains the I/O control blocks that must be protected from user alteration (including alteration by Record Management). The AMBXN is *not* in Subpool 252 (which has storage protection key of 0) because it contains fields from the AMB and the IOMB that Record Management needs to update. The AMBXN is in Subpool 250, along with other unprotected control blocks.

All control blocks for a catalog are kept in global storage, in either Subpools 231 and 241 in the common service area (CSA) or Subpool 245 in SQA.

"Virtual-Storage Management" in "Diagnostic Aids" indicates the subpools in which storage for particular control blocks is indicated.

Control Block Formats

This section discusses VSAM control blocks and (except for those adequately covered in *OS/VS2 Data Areas*) gives their format.

OS/VS2 VSAM Cross Reference (microfiche) has a "Symbol Where Used Report" that lists alphabetically all the symbols used in VSAM modules, in particular the labels of the control blocks discussed here. With each symbol are listed all the modules that use it, along with a code that tells how each symbol is used:

D	defined
R	read (that is, referenced without alteration)
W	written (that is, altered)
C	compared

ABP—Actual Block Processor (I/O-Management Communication Vector Table)

The ABP is a communication vector table that contains entry points for I/O Management modules located in the nucleus. It is link-edited in the nucleus as IDA121CV, along with the modules.

The ABP is created by NIP and pointed to by the system CVT (CVTIOBP).

Actual Block Processor (ABP)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0 (0)	1	ABPID	Control block identifier, X'C1'
1 (1)	1	ABPLEN	Length of the ABP
2 (2)	2	ABPBR14	Unconditional branch, register 14
4 (4)	4	ABPSIOD	Address of the Supervisor-State I/O Driver (IGC121)
8 (8)	4	ABPABP	Address of the Actual Block Processor routine (IDA121A2)
12 (C)	4	ABPNE	Address of the Normal End Appendage (IDA121A3)
16 (10)	4	ABPAE	Adddress of the Abnormal End Appendage (IDA121A4)

ACB—Access Method Control Block

The VSAM ACB describes a VSAM cluster. It is built by the user's program with the ACB or GENCB macro. Before the cluster is opened, the ACB can be modified by the user's DD statements and by the MODCB macro. After the cluster is opened, the ACB is pointed to by the RPL (RPLDACB) that describes the user's record processing request.

Access Method Control Block (ACB)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0 (0)	1	ACBID	Control block identifier, X'A0'
1 (1)	1	ACBSTYP	Subtype: X'10' = VSAM X'20' = VTAM
2 (2)	2	ACBLENG	Length of the ACB
4 (4)	4	ACBAMBL ACBIXLST ACBJWA ACBIBCT	Address of the AMBL Address of the index list
8 (8)	4	ACBINRTN	Address of the VSAM Interface routine (IDA019R1)
12 (C)	2	ACBMACRF	MACRF flags:
		ACBMACR1	MACRF flag byte 1:
	1...	ACBKEY	The record is identified by a key—keyed processing
	.1..	ACBADR ACBADD	The record is identified by a RBA (relative byte address)—addressed processing
	..1.	ACBCNV	Control interval processing ACBBLK
	...1	ACBSEQ	Sequential processing
	.... 1...	ACBDIR	Direct processing
	.... .1..	ACBIN	Input (GET, READ) processing
	.... ..1.	ACBOUT	Output (PUT, WRITE) processing
	1	ACBUBF	User-supplied buffer space
13 (D)		ACBMACR2	MACRF flag byte 2:
	...1	ACBSKP	Skip sequential processing
	.... 1...	ACBLOGON	VTAM LOGON indicator
	.... .1..	ACBRST	Set data set to empty state
	.... ..1.	ACBDSN	Basic subtask shared control-block connection on common DSNAMEs
	1	ACBAIX	Object to be processed is the alternate index of the path specified in the given DDNAME
	xxx.		Reserved
14 (E)	1	ACBBSTNO	Number of concurrent strings for alternate-index path
15 (F)	1	ACBSTRNO	Number of RPL strings
16 (10)	2	ACBBUFND	Number of buffers requested for data
18 (12)	2	ACBBUFNI	Number of buffers requested for index
20 (14)	4	ACBBUFPL	Address of the buffer header (BUFC)

AMBL—Access Method Block List

The AMBL describes a VSAM cluster and points to the cluster's data set and index AMBs. When the cluster is opened, an AMBL is built to describe the cluster. If the cluster's data set (and index) is shared with other users, AMBs already exist for the data set (and index). The existing AMB's addresses are put into the AMBL. If the cluster is not shared, AMBs are built to describe the cluster's data set and, if the cluster is key-sequenced, to describe the data set's index. The AMBL is pointed to by the cluster's ACB (ACBAMBL).

Access Method Block List (AMBL)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0 (0)	4	AMBLPCHN	Address of the primary AMBL in the AMBL chain
4 (4)	4	AMBLSCHN	Address of the secondary AMBL in the AMBL chain
8 (8)	4	AMBLACB	Address of the ACB associated with the AMBL
12 (C)	1	AMBLEFLG	End of Volume flags:
	1...	AMBLWAIT	End of volume is waiting
	.1..	AMBLESET	End of Volume encountered an error and restored control blocks to their original condition
	..xx xxxx		Reserved
13 (D)	1	AMBLCOMP	End of Volume lock
14 (D)	2		Reserved
16 (10)	8	AMBLDDNM	The ACB's DDNAME field
16 (10)	8	AMBLIDF	Cluster identifier
16 (10)	4	AMBLCACB	Address of the ACB of the catalog
20 (14)	3	AMBLDCI	Control-interval number of the catalog data record
23 (17)	1	AMBLQ	Qualifier:
	1...	AMBLDDC	DD connect only
	.1..	AMBLGSR	Cluster opened for global shared resources
	..1.	AMBLLSR	Cluster opened for local shared resources
	...1	AMBLFSTP	Cluster opened for fast path (improved control-interval access)
	.... 1...	AMBLUBF	Cluster opened for user buffering
	.... .1..	AMBLKSDS	Cluster opened as a key-sequenced data set
	.... ..1.	AMBLESDS	Cluster opened as an entry-sequenced data set
	1	AMBLDFR	Cluster opened for deferred writes
24 (18)		AMBLXPT	In a base AMBL, address of the path AMBL; in a path AMBL, address of the base AMBL
28(1C)	2	AMBLVC	Identifies the entry in the valid-AMBL table that idnetifies this AMBL:
28(1C)	1	AMBLVRT	Number of the valid-AMBL table in the chain of valid-AMBL tables
29(1D)	1	AMBLENO	Offset within the valid-AMBL table

Access Method Block List (AMBL)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
30(1E)	1	AMBLTYPE	Type of control block structure opened:
	1...	AMBLPATH	Path
	.1..	AMBLUPGR	Upgrade set
	..1.	AMBLAIX	Alternate index
	...1	AMBLBASE	Base cluster
	.... 1...	AMBLFIX	Control blocks are fixed in real storage
	xxx		Reserved
31(1F)	1	AMBLQ2	Qualified extension
	1...	AMBLCBIC	Cluster opened with control blocks in common storage area
	.xxx xxxx		Reserved
32 (20)	1	AMBLID	Control block identifier, X'50'
33 (21)	1	AMBLSHAR	Sharing indicators:
	1...	AMBLPRIM	Identifies the primary AMBL
	.1..	AMBLCATO	The catalog is open
	..1.	AMBLWRIT	The user intends to write or update records in the data set
	...x xxxx		Reserved
34 (22)	1	AMBLLEN	Length of the AMBL
35 (23)	1	AMBLFLG1	Flags:
	1...	AMBLFULL	The user-supplied master password was verified
	.1..	AMBLCINV	The user-supplied control-interval password was verified
	..1.	AMBLUPD	The user-supplied update password was verified
	...1	AMBLVVIC	The AMBL is for the mass storage volume inventory (MSVI) data set
	...1	AMBLSDS	The AMBL is for a system data set
	.... 1...	AMBLSCRA	The AMBL is for a catalog recovery area in system storage
	.... .1..	AMBLUCRA	The AMBL is for a catalog recovery area in user's storage
	.... ..1.	AMBLCAT	The AMBLACB field points to a catalog's ACB
	1	AMBLDUMY	A DD DUMMY statement was specified
	...x x.x.		The combination of these bits indicates the type of data set: 001 Catalog 101 MSVI 011 SCRA
36 (24)	1	AMBLFLG2	Flags:
	...1	AMBLSTAG	Cluster is staged
	.... 1...	AMBLII	This AMBL is for an ISAM interface structure
	xxx. .xxx		Reserved
37 (25)	1	AMBLNST	Number of strings
38 (26)	2	AMBLNUM	Number of AMB pointers in the AMBL
40 (28)	1		Reserved
41 (29)	1	AMBLNIDS	Number of identifiers
42 (2A)	10	AMBLMIDS	Five 2-byte fields, each containing a VSAM module's identifier
52 (34)	4	AMBLDTA	Address of the cluster's data set AMB

Access Method Block List (AMBL)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
56 (38)	4	AMBLIX	Address of the cluster's index AMB
60 (3C)	4	AMBLBIB	Address of the base information block
64 (40)	4	AMBLCMB	Address of the cluster management block

Index Modification Work Area (IMWA)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
	..1.	IMWBSE	Indicates the new index entry should be a section entry
	.x xxxx		Reserved
2 (2)	2	IMWLEN	Length of IMWA in bytes
4 (4)	4	IMWIXSP	Address of index search parameter list
8 (8)	32	IMWISWKA	Index search parameter list (see IXSPL)
40 (28)	4	IMWXKEYP	Address of the next (higher-keyed) index entry
44 (2C)	4	IMWIKEYP	Address of the new index entry's key
48 (30)	4	IMWXPTR	Value of the index pointer field in the next (higher-keyed) index entry
52 (34)	4	IMWIPTR	Value to be inserted in new index entry's pointer field
56 (38)	4	IMWLBUFC	Address of a data BUFC for a data buffer containing the lowest key following a control-area split
60 (3C)	4	IMWBUFP	Address of the index record being processed
64 (40)	1	IMWFGAIN	Front key-compression adjustment to be added to IBFLPF field in next (higher-keyed) index entry
65 (41)	1	IMWIEL	Value of IBFLPL field—that is, compressed length of new index entry's key
66 (42)	1	IMWSVIEL	Save area for IBFLPL value
67 (43)	1		Reserved
68 (44)	2	IMWCIMVN	Readjustment to number of control intervals in old control area following a control area split to enable an index record to be built for the new control area
70 (46)	2	IMWNSOFF	Offset to next section entry in index record
72 (48)	4		Reserved
76 (4C)	(key length)	IMWKEY1	Highest possible key for a mass insertion—that is, last key in a sequence of keys to be inserted which is less than an existing key; also, save area for current insert key under a no-fit condition

IOMB—I/O-Management Block

The IOMB is used by I/O Management to control its processing of a request. It contains the addresses of other control blocks, flags used by I/O Management, and a 16-word register save area. The addresses of the first BUFC and CPA are inserted by I/O Management after it verifies the control blocks.

The IOMB is pointed to by the PLH (PLHIOB). It points to the IOSB, which points to the SRB. These three control blocks take the place of the IOB, which is used by some other drivers of the I/O Supervisor.

I/O Management Block (IOMB)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0 (0)	4	IOMBID	Control block identifier: 'IOMB'
4 (4)	4	IOMBUFC	Address of the first BUFC
8 (8)	4	IOMCPA	Address of the first CPA
12 (C)	4	IOMPLH	Address of the PLH
16 (10)	4	IOMAMB	Address of the AMB
20 (14)	4	IOMIQE	Address of the IQE (interrupt queue element)
24 (18)	4	IOMECBPT	Address of the ECB
28 1C)	4	IOMVSL	Address of the VSL (virtual subarea list, which is the same as the PFL, page fix list)
32 (20)	4	IOMPGAD	Address of the caller to get control when I/O operation is complete (zero for Record Management—used for Auxiliary Storage Management)
36 (24)	4	IOMIOSB	Address of the IOSB
40 (28)	3	IOMFLAGS	Flags:
40 (28)	2	IOMFL	Flags reset after I/O completes:
			Byte 1:
	xx..	IOMAPEND	Appendage flags:
	1...	IOMNE	Normal End Appendage completed
	.1..	IOMAE	Abnormal End Appendage completed
	..1.	IOMPURGE	A purge is in progress
	.... 1...	IOMCBERR	A control block wasn't valid
	.... .1..	IOMADERR	Virtual addresses in the VPL weren't successfully converted to real addresses for the IDAL
	.... ..1.	IOMPGFIX	Pages are fixed in real storage
	1	IOMCSW	The address of the channel status word is incorrect
	...x		Reserved
			Byte 2:
	1...	IOMDDR	Dynamic-device reconfiguration
	.1..	IOMCPRB	Problem state caller
	..xx		Reserved
	.... 1...	IOMEEXIT	Channel end appendage exited
	.... .1..	IOMIRBSW	Asynchronous processing scheduled
	xx		Reserved
42 (2A)	1	IOMSTIND	Status indicators:
	1...	IOMAMUSE	The IOMB is in use
	.1..	IOMEOVW	End of Volume is waiting for an IOMB
	..1.	IOMEOVTS	End of Volume has set the IOMLOCK field
	...1	IOMEOVXC	End-of-Volume indicator
	.... 1...	IOMLLOCK	A local lock is held

I/O Management Block (IOMB)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
	.... .1..	IOMSLOC	SALLOC is held
	.... ..1.	IOMSRBM	The user is processing with SRB (event) dispatching
	1	IOMSR	Suspend/Resume indicator

VGTT—VSAM Global Termination Table

The VGTT identifies global virtual storage that may need to be specially freed if an error prevents it from being freed normally. There are various types of VGTTs for:

- Keeping track of an address space's use of a global VSAM resource pool (for processing with global shared resources)
- Keeping track of certain control blocks (sets of IOSB, SRB, and PFL) that are kept in global storage for processing with local shared resources (the normal pointers to these control blocks are in unprotected local storage—the VGTT is in global storage, adjacent to them)
- Keeping track of control blocks during the opening of a catalog, a catalog recovery area, or the mass storage volume inventory data set (all of whose control blocks are kept in global storage
- Keeping track of certain control blocks (sets of IOSB, SRB, and PFL) that are kept in global storage for processing a user data set and all related data sets (such as alternate indexes)
- Keeping track of control blocks (ECBs and WAITLIST if also GSR) that are kept in global storage for stage by key range processing.

The VGTT is pointed to by the ASCB (address space control block). It is chained to the next VGTT for the address space.

VSAM Global Termination Table (VGTT)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0 (0)	4	VGTTID	Control block identifier: 'VGTT'
4 (4)	1	VGTTTYPE	VGTT type indicator:
	1..	VGTTSDS	VGTT is for system data set
	.1..	VGTTGSR	VGTT is for processing with global shared resources—VGTTVUSE is used
	..1.	VGTTLSR	VGTT is for processing with local shared resources
	...1	VGTTCTLG	VGTT is for opening a catalog, a catalog recovery area, or the mass storage volume inventory data set
	.... 1...	VGTTOPEN	VGTT is for processing a user's data set without shared resources
	.... .1..	VGTTCBIC	Opened with control blocks in common storage area
	xx		Reserved
5 (5)	1		Reserved
6 (6)	1	VGTTGSRK	GSR key type, if VGTTGSR is on
7 (7)	1	VGTTSP	Subpool number of the VGTT and of the global storage it protects
8 (8)	4	VGTTSIZE	Length of the VGTT
12 (C)	4	VGTTNEXT	Address of the next VGTT (zero for the last VGTT in the chain)
16 (10)	4	VGTTBIB	Address of the base information block for the user's data set and all related data sets (such as alternate indexes)
20 (14)	4	VGTTVUSE	For a VGTT for global shared resources, the use count that was contributed by the processing of the user's data set and all related data sets.

VSAM Global Termination Table (VGTT)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
24 (18)	4	VGTTPSB	Address of the protected sphere block (which contains HEBs for use by Virtual-Storage Management)
28 (1C)	4		Reserved
32 (20)	VL	VGTTCORE	For a VGTT for local shared resources, the virtual-storage area the VGTT protects

VIOT—Valid-IOMB Table

The VIOT contains the address of each valid IOMB within a VSAM resource pool (for processing with shared resources). It is pointed to by the VSRT (VSRTVIOT) and by each AMB associated with the resource pool.

Valid-IOMB Table (VIOT)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0(0)	4	VIOHDR	Header
0(0)	1	VIOID	Control block identifier, X'16'
1(1)	1		Reserved
2(2)	2	VIOLEN	Length of the valid-IOMB table
4(4)	4 x *n*	VIOPTR	Address of a valid IOMB; this field is repeated *n* times

VMT—Volume Mount Table

The VMT identifies and describes volumes that are mounted for a base cluster and all clusters associated with it for processing. There is a VMT for each device type. The first VMT is pointed to by the BIB (BIBVMT).

Volume Mount Table (VMT)—Description and Format

Offset	Bytes and Bit Pattern	Field Name	Description
0 (0)	4	VMTHDR	Header:
0 (0)	1	VMTID	Control block identifier, X'12'
1 (1)	1		Reserved
2 (2)	2	VMTLEN	Length of the VMT
4 (4)	4	VMTNXT	Address of the next VMT
8 (8)	2	VMTNOVOL	Number of volume entries (*n*) in the VMT
10 (A)	3		Reserved
13 (D)	3	VMTDEV	Device information:
13 (D)	1	VMTDVOPT	Device options
14 (E)	2	VMTDVTYP	Device class and type
16 (10)	16 x *n*	VMTVOL	Volume entry for a volume to be mounted:
16 (10)	4	VMTUSECT	Use count
20 (14)	1	VMTVFLG1	Volume flags:
	1...	VMTOPEN	The volume is being processed by Open
	.xxx xxxx		Reserved
21 (15)	1		Reserved
22 (16)	6	VMTVLSER	The volume's serial number
28 (1C)	4	VMTUCB	Address of the UCB for the volume

Function Code	Module that Detected Error	Operation Being Performed When Error Was Detected
Open		
1	IDA0192C	Initialize for catalog interface processing.
2	IDA0192C	Determine which data sets are associated with data-set name on DD statement, determine catalog, and check password.
3	IDA0192C	Determine data-set attributes.
4	IDA0192C	Get volume information.
5	IDA0192C	Update "open" indicator in catalog.
6	IDA0192C	Update catalog when data set is being closed.
7	IDA0192C	Retrieve volume timestamp.
8	IDA0192C	Record-Management catalog update.
9	IDA0192C	Update preformat indicator in catalog.
10	IDA0192C	Retrieve 44-byte cluster name.
11	IDA0192C	Retrieve 44-byte component name.
20	IDA0192V	Initialize for mounting and verify volume.
21	IDA0192V	Check volume timestamp.
22	IDA0192V	Handle messages.
23	IDA0192V	Mount volume.
30	IDA0192S	Initialize for SMF processing.
31	IDA0192S	Build SMF record.
40	IDA0192D	Initialize for staging.
41	IDA0192D	Build UCB list.
42	IDA0192D	Build list for ACQUIRE/RELINQUISH (stage/destage).
43	IDA0192D	Issue ACQUIRE or RELINQUISH.
50	IDA0192Z	Initialize for building control blocks.
51	IDA0192Z	Determine number of buffers needed.
52	IDA0192Z	Build buffers.
53	IDA0192Z	Build control blocks.
54	IDA0192Y	Build string blocks.
60	IDA0192B	Module initialization.
61	IDA0192B	Locate data-set attributes and check them for validity.
62	IDA0192B	Volume processing.
63	IDA0192B	Preformat extent.
70	IDA0192W	Initialize for building channel program.
71	IDA0192W	Build channel program area.
80	IFG0193A	Check return codes from IFG0191X or IFG0191Y.
81	IDA0192A	Initialize for VSAM Open processing.
82	IDA0192A	Verify ACB.
83	IDA0192F	Fix control blocks in real storage.
84	IDA0192B	Allow subtasks to share data set.
85	IDA0192F	Mount and verify volumes.
87	IDA0192A	Determine whether to connect base cluster to an existing structure or generate a new structure.

Function Code	Module that Detected Error	Operation Being Performed When Error Was Detected
88	IDA0192F	Open base cluster.
89	IDA0192F	Open alternate index in upgrade set.
90	IDA0192F	Open alternate index in path.
93	IDA0192A	Build a dummy DEB.
95	IDA0192A	Terminate VSAM Open processing.
96	IDA0192A	Clean up after an error in Open processing.
99	IFG0192B	Error processing for ACB being processed on a system not generated for VSAM.
Close		
100	IFG0200V	Read JFCB.
101	IDA0200T	Initialize for VSAM Close processing.
102	IDA0200T	Check validity of AMBL and DEB.
103	IDA0200T	Complete deferred write requests.
104	IDA0200T	Close path.
105	IDA0200T	Close base cluster.
106	IDA0200T	Close sphere (close upgrade alternate indexes and free storage).
107	IDA0200T	Close upgrade set.
108	IDA0200T	Process volume mount table.
109	IDA0200T	Close dummy data set.
110	IDA0200B	Module initialization.
111	IDA0200B	Check validity of AMBLs and DEBs.
112	IDA0200B	SMF processing.
113	IDA0200B	Update statistics and RBA information in the catalog.
114	IDA0200B	Free storage for control blocks.
115	IDA0200B	Write a buffer.
148	IDA0200T	Force deletion of VSAM global resource pool.
149	IDAOCEA2	Force deletion of VSAM global resource pool.
150	IGC0002C	Read JFCB.
151	IDA0231T	Initialize for VSAM Close (TYPE=T) processing.
152	IDA0231T	Check validity of AMBL and DEB.
153	IDA0231T	Complete deferred write requests.
154	IDA0231T	Close (TYPE=T) path.
155	IDA0231T	Close (TYPE=T) base cluster.
156	IDA0231T	Close (TYPE=T) upgrade set.
157	IDA0231B	Module initialization.
158	IDA0231B	Check validity of AMBLs and DEBs.
159	IDA0231B	Update statistics and RBA information.
160	IDA0231B	SMF processing.
161	IDA0231B	Write a buffer.
End of Volume		
200	IFG0551F	Read JFCB.

Function Code	Module that Detected Error	Operation Being Performed When Error Was Detected
201	IDA0557A	Initialize for VSAM End-of-Volume processing.
202	IDA0557A	Locate and mount volume.
203	IDA0557A	Allocate space.
204	IDA0557A	Switch volumes.
205	IDA0557A	Build control blocks.
206	IDA0557A	Update SMF record.
207	IDA0557A	Preformat extent.
208	IDA0557A	Record Management, catalog update.
209	IDA0557A	Reset control blocks.

Macros

The following tables list VSAM and OS/VS macros and explain what they do. The macros are divided into those that define control blocks and data areas (mapping macros) and those that issue executable code (action macros).

OS/VS2 VSAM Cross Reference (microfiche) has a table ("Macro Where Used Report") of all the macros issued by VSAM with a listing of the modules (and other macros) that issue them.

Mapping Macros

The following table lists macros that define the format of control blocks and data areas used by VSAM modules.

Macros That Define Data Areas

Macro	Description
ACB	Builds an access-method control block (ACB) at assembly time
CVT	Maps the communication vector table (CVT)
ECB	Maps the event control block
IDAACQPL	Maps the acquire range parameter list (ACQPL)
IDAAIR	Maps the alternate-index record
IDAAMB	Maps the access method block (AMB)
IDAAMBL	Maps the access method block list (AMBL)
IDAAMBXN	Maps the access method block extension (AMBXN)
IDAAMDSB	Maps the access method data set statistics control block (AMDSB)
IDAARDB	Maps the address range definition block (ARDB)
IDAARWA	Maps a recovery work area for the Restart modules
IDABIB	Maps the base information block (BIB)
IDABFR	Maps the buffer control set
IDABLPRM	Maps the resource pool parameter list (BLPRM)
IDABSPH	Maps the buffer subpool header (BSPH) for shared resources
IDABUFC	Maps the buffer control block (BUFC)
IDACBTAB	Maps the tables used by the Control Block Manipulation routine
IDACIDF	Maps the control-interval descriptor field (CIDF)
IDACLWRK	Maps the close work area

Macros That Define Data Areas (continued)

Macro	Description
IDACMB	Maps the cluster management block (CMB)
IDACNVPL	Maps the convert keys/RRNs/RBAs parameter list (CNVPL)
IDACPA	Maps the channel program area (CPA)
IDACSL	Maps the core save list (CSL)
IDACTREC	Maps the work area built when the VS2 Catalog-Management routines use Open, Close, or End of Volume
IDADIWA	Maps the data insert work area (DIWA)
IDADSECT	Maps miscellaneous data areas for Checkpoint/Restart
IDADSL	Maps the DEB save list (DSL)
IDAEDB	Maps the extent definition block (EDB)
IDAELEM	Maps the Control Block Manipulation routine's element argument control entry
IDAEQUS	Defines the equates for the ISAM Interface: SYNAD—Message-Build routine
IDAERMSG	Maps the ISAM Interface: SYNAD Message format
IDAERRCD	Lists the VSAM Open and Close ACB Error Codes
IDAESL	Maps the enqueue save list (ESL)
IDAFOREC	Maps the work area for VSAM Open, Close, and End of Volume (the work area is called "FORCORE" in program comments)
	IDAFOREC issues IDAPDPRM, IEFJFCBN, and IEFJFCBX.
IDAGENC	Maps the GENCB header argument control entry
IDAHEB	Maps the header element block (HEB)
IDAICWA	Maps the index create work area (ICWA)
IDAIDXCB	Lists the VSAM control-block-identifier codes
IDAIICB	Maps the ISAM-Interface control block (IICB)
IDAIIREG	Defines the ISAM-Interface register usage
	IDAIIREG issues IDAIICB, IDARPLE, IFGRPL, IHADCB, and IHADCBDF.
IDAIMWA	Maps the index modification work area (IMWA)
IDAIOB	Maps the VSAM IOB extension
IDAIOMB	Maps the I/O-Management control block (IOMB)
IDAIOSCN	Maps the VSAM Open, Close, and End-of-Volume commonly-used declarations
IDAIRD	Defines the index record
IDAIXSPL	Maps the index search parameter list (IXSPL)
IDALPMB	Maps the logical-to-physical mapping block (LPMB)
IDAMNTPL	Maps the mount volume and acquire parameter list (MNTPL)
IDAMODC	Maps the MODCB header argument control entry
IDAOPWRK	Maps the ACB work area for Open (OPW or OPWRK)
IDAPDPRM	Maps the VSAM Open, Close, and End-of-Volume problem determination parameter list
IDAPLH	Maps the placeholder (PLH)
IDAPSL	Maps the page save list (PSL)
IDARBAPL	Maps the RBA paris list (RBAPL)

Macros That Define Data Areas (continued)

Macro	Description
IDARDF	Maps the record definition field (RDF)
IDAREGS	Defines register usage for all Record-Management modules
IDARMRCD	Lists the Record-Management return codes
IDARPLE	Maps the ISAM-Interface request parameter list extension (RPLE)
IDARTMAC	Maps data structures for recovery routines
IDASHOW	Maps the SHOWCB header argument control entry
IDASSL	Maps the swap save list (SSL)
IDATEST	Maps the TESTCB header argument control entry
IDAUPT	Maps the upgrade table (UPT) for upgrading alternate indexes
IDAVAT	Maps the valid-AMBL table (VAT)
IDAVCRT	Maps the VSAM checkpoint/restart table (VCRT), the VSAM checkpoint/restart storage blocks (VCRCORE), and the HEB save area (VCRHEBSA).
IDAVGTT	Maps the VSAM global termination table (VGTT)
IDAVIOT	Maps the valid-IOMB table (VIOT)
IDAVMT	Maps the volume mount table (VMT)
IDAVSRT	Maps the VSAM shared resource table (VSRT)
IDAVUCBL	Maps the VSAM Open and End of Volume: Volume Mount and Verify UCB list
IDAVVOLL	Maps the VSAM Open and End of Volume: Volume Mount and Verify volume serial number list
IDAWAX	Maps the work area for path processing (WAX)
IDAWSHD	Maps the working storage header (WSHD)
IECDIOCM	Maps the communication area of the VS2 I/O Supervisor
IECDIOSB	Maps the I/O-Supervisor control block (IOSB)
IECDIPIB	Maps the I/O Supervisor—Purge interface block (IPIB)
IECDSECS	Maps the DSECS
IECDSECT	Maps the common open/close work area
IECRRPL	Maps the common O/C/EOV Recovery Routine parameter list
IECSDSL1	Maps the SDSL1
IEESMCA	Maps the SMCA
IEEVCHWA	Maps a work area that VS2 Checkpoint passes to VSAM Checkpoint
IEEVRSWA	Maps a work area that VS2 Restart passes to VSAM Restart
IEFJFCBN	Maps the job file control block (JFCB)
IEFJFCBX	Maps the job file control block (JFCB)
IEFJMR	Maps the JMR
IEFTCT	Maps the TCT
IEFTIOT1	Maps the task input/output table (TIOT)
IEFUCBOB	Maps the VS2 unit control block (UCB)
IEZABP	Maps the ABP—I/O-Management communication vector table (module IDA121CV)
IEZCTGFL	Maps the VS2 catalog field parameter list (CTGFL)
IEZCTGPL	Maps the VS2 catalog parameter list (CTGPL)
IEZDEB	Maps the VS2 data extent block (DEB)

Macros That Define Data Areas (continued)

Macro	Description
IEZIOB	Maps the VS2 input/output block (IOB)
IEZJSCB	Maps the VS2 job step control block (JSCB)
IFGACB	Maps the access-method control block (ACB)
IFGEXLST	Maps the exit list (EXLST)
IFGRPL	Maps the request parameter list (RPL)
IGGCAXWA	Maps the VS2 catalog auxiliary work area (CAXWA)
IHAASCB	Maps the VS2 address space control block (ASCB)
IHAASXB	Maps the VS2 address space extension control block (ASXB)
IHADCB	Maps the VS2 data set control block (DCB)
IHADCBDF	Maps the VS2 data set control block (DCB)
IHADECB	Maps the VS2 data extent control block (DECB)
IHADSAB	Maps the VS2 data set association block (DSAB)
IHAFRRS	Maps VS2 Recovery Termination Manager Dsects for function recovery routines
IHAIQE	Maps the VS2 interrupt queue element (IQE)
IHAPSA	Maps the VS2 prefixed save area (PSA)
IHAPVT	Maps the VS2 page vector table (PVT)
IHARB	Maps the VS2 request block (RB)
IHARMPL	Maps the VS2 Resource Manager's parameter list for interfacing with VSAM Task Close Executor (IDA0CEA2)
IHASDWA	Maps the STAE diagnostic work area (SDWA, also called the recovery termination communication area—RTCA)
IHASRB	Maps the VS2 service request block (SRB)
IHJSSCR	Maps the subsystem control record of areas saved at checkpoint time
IKJRB	Maps request blocks
IKJTCB	Maps the task control block (TCB)
XCTLTABL	Maps the VS2 XCTL table

Action Macros

This table lists the macros issued by VSAM that generate executable code.

Macros That Generate Executable Code

Macro	Description
ABEND	Abnormal termination (VS2 macro)
ACQRANGE	Stage a range of data from a VSAM data set
BLDVRP	Builds a VSAM resource pool for shared resources
CATLG	Loads the address of the catalog parameter list (CTGPL) into register 1 and issues SVC 26
CLOSE	VSAM CLOSE: Disconnects a user from a VSAM data set
CNVTAD	Convert key/RRN/RBA to volume serial and RBA
DEBCHK	Checks the validity of the DEB
DELETE	(Same as VS2 DELETE macro)
DEQ	(Same as VS2 DEQ macro)
DLVRP	Deletes a VSAM resource pool for shared resources

Macros That Generate Executable Code (continued)

Macro	Description
DOM	Deletes operator message (VS2 DOM macro)
ENDREQ	Terminates a VSAM record processing request (such as GET or PUT)
ENQ	(Same as VS2 ENQ macro)
ERASE	Deletes a VSAM record
ESTAE	Specifies task asynchronous exit (VS2 macro)
EXCP	(Same as VS2 EXCP macro)
FREEMAIN	Releases virtual storage obtained by a GETMAIN
GENCB	Generates a VSAM control block (ACB, EXLST, or RPL)
GET	Retrieves a record from a data set on a direct-access device
GETIX	Retrieves a control interval from the index of a key-sequenced data set
GETMAIN	Obtains virtual storage for a temporary work area
GTRACE	Calls the Generalized Trace Facility (GTF) to copy VSAM control blocks
IDACALL	Transfers control from procedure A to procedure B and allows procedure B to return control to procedure A at the instruction following the IDACALL instruction-expansion
IDACB1	Transforms operands for Control Block Manipulation macros (GENCB, MODCB, SHOWCB, and TESTCB)
IDACB2	Scans keywords and generates code for Control Block Manipulation macros
IDAERMAC	Prints MNOTEs for Control Block Manipulation macro user-programmer errors
IDAEXITR	Transfers control from VSAM modules to a user's exit routine and allows the user exit routine to return control to the VSAM module at the instruction following the IDAEXITR instruction-expansion IDAEXITR issues DELETE, IDARST14, IDASVR14, and LOAD.
IDAGMAIN	Gets virtual storage for VSAM Open, Close, and End of Volume
IDAPATCH	Generates maintenance space
IDAPFMT	Gives control to End of Volume to preformat a control area
IDARST14	Puts the return address in register 14
IDASVR14	Saves register 14 in the placeholder (PLH) push-down list
IECRES	Transfers control to the VS2Resident routine
LOAD	(Same as VS2 LOAD macro)
MNTACQ	Mounts a volume and stages VSAM records into it
MODCB	Modifies a VSAM control block (ACB, EXLST, or RPL)
MODESET	(Same as VS2 MODESET macro)
MRKBFR	Marks a buffer in a VSAM resource pool
OBTAIN	(Same as VS2 OBTAIN macro)
OPEN	Connects a user's program to a VSAM data set
PGFIX	"Fixes" a page of virtual storage so that it remains in real storage for a duration
PGFREE	"Frees" a "fixed" page of virtual storage.
POINT	Identifies a starting point in a VSAM data set
POST	(Same as VS2 POST macro)
PUT	Writes a record into a VSAM data set
PUTIX	Writes a control interval in the index of a key-sequenced data set

Macros That Generate Executable Code (continued)

Macro	Description
RESERVE	(Same as VS2 RESERVE macro)
RETURN	(Same as VS 2 RETURN macro)
SCHBFR	Searches for a control interval in a VSAM resource pool
SDUMP	Schedules SVC dump routine (VS2 macro)
SETFRR	Sets up functional recovery routine (VS2 macro)
SETLOCK	Obtains or releases a lock (VS2 macro)
SETRP	Records recovery information (VS2 macro)
SHOWCAT	Displays information from a VSAM catalog
SHOWCB	Displays information from a VSAM control block (ACB, EXLST, or RPL)
SMFWTM	Writes the SMF message into the SMF data set
STARTIO	Gives control to the VS2 I/O Supervisor to start an I/O operation
SYNCH	(Same as ISAM SYNCH macro)
TERMRPL	Releases owned resources of a terminated RPL and restarts deferred synchronous requests
TESTAUTH	Checks authorization of a calling module to perform certain functions
TESTCB	Tests information in a VSAM control block (ACB, EXLST, or RPL)
TIME	Obtains the correct time from the VS2 system time-of-day clock
VERIFY	Gives control to Record Management to check the end-of-data indicators for Checkpoint/Restart or for Access Method Services VERIFY command
WAIT	(Same as VS2 WAIT macro)
WRTBFR	Writes buffers from a VSAM resource pool
WTO	Writes a message to the operator (no reply)
XCTL	Transfers control (VS2 XCTL macro)

Note: The use of these user macros is described in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide:*

CLOSE
ENDREQ
ERASE
GENCB
GET
MODCB
OPEN
POINT
PUT
SHOWCB
TESTCB

The use of these user macros is described in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications:*

| ACQRANGE
BLDVRP
| CNVTAD
DLVRP
GETIX
| MNTACQ
MRKBFR
PUTIX
SCHBFR
SHOWCAT
WRTBFR

After an asynchronous request for access to a data set, VSAM indicates in register 15 whether the request was accepted, as follows:

Reg 15	Condition
0(0)	Request was accepted.
4(4)	Request was not accepted because the request parameter list indicated by the request (RPL=address) was active for another request.

After a synchronous request, or a CHECK or ENDREQ macro, register 15 indicates whether the request was completed successfully, as follows:

Reg 15	Condition
0(0)	Request completed successfully.
4(4)	Request was not accepted because the request parameter list indicated by the request (RPL=address) was active for another request.
8(8)	Logical error; specific error is indicated in the feedback field in the RPL.
12(C)	Physical error; specific error is indicated in the feedback field in the RPL.

Paired with the 0, 8, and 12 indicators in register 15 are return codes in the feedback field of the request parameter list.

The feedback return codes for the 0 indicator in register 15, which doesn't cause VSAM to exit to an exit routine, are:

RPLFDBK Code	Condition
0(0)	Request completed successfully.
4(4)	Request completed successfully. For retrieval, VSAM mounted another volume to locate the record; for storage, VSAM allocated additional space or mounted another volume.
8(8)	For GET requests, indicates that a duplicate key follows; for PUT requests, indicates that a duplicate key was created in an alternate index with the nonunique attribute.
12(C)	(Shared resources only.) A buffer needs to be written.
16(10)	Control area split was required because a sequence set control interval had free space insufficient to contain the key to be inserted.
20(14)	Data set is not on a virtual DASD for MNTACQ/ACQRANGE request. Detected by IDA0192E.
28(1C)	A CI split for the CI with the RBA acquired from RPLDDDD which was interrupted. The CI was read as nonupdate with address access. This warning condition indicates that duplicate data records may exist.
32(20)	Request deferred for a resource held by the terminated RPL is asynchronous and cannot be restarted by TERMRPL.
36(24)	Possible data set error condition was detected by TERMRPL: 1. The request was abnormally terminated in the middle of its I/O operation. 2. One of the data/index BUFCs of the string contains data that needs to be written (BUFCMW=ON) but it was invalidated by TERMRPL.
40(28)	Error in PLH data BUFC pointer was detected by TERMRPL.

See the discussions below for the logical-error return codes and for the physical-error return codes.

Function Codes for Logical and Physical Errors

When a logical or physical error occurs during processing that involves alternate indexes, VSAM provides a code in the RPLCMPON field that indicates whether the base cluster, its alternate index, or its upgrade set was being processed and whether upgrading was okay or might have been incorrect because of the error:

Code	What Was Being Processed	Status of Upgrading
0(0)	Base cluster	Okay
1(1)	Base cluster	Might be incorrect
2(2)	Alternate index	Okay
3(3)	Alternate index	Might be incorrect
4(4)	Upgrade set	Okay
5(5)	Upgrade set	Might be incorrect

Logical-Error Return Codes

When a logical-error-analysis exit routine (LERAD) is provided, it gets control for logical errors, and register 15 doesn't contain 8, but contains the entry address of the LERAD routine.

Figure 57 gives the contents of the registers when VSAM exits to the LERAD routine.

If a logical error occurs and a LERAD exit routine isn't provided (or the LERAD exit is inactive), VSAM returns control to the processing program following the last executed instruction. Register 15 indicates a logical error (8), and the feedback field in the request parameter list contains a code identifying the error. Register 1 points to the request parameter list.

Reg	Contents
0	Unpredictable.
1	Address of the request parameter list that contains the feedback field the routine should examine. The register must contain this address if the exit routine returns to VSAM.
2-13	Same as when the request macro was issued. Register 13, by convention, contains the address of the processing program's 72-byte save area, which may not be used as a save area by the LERAD routine if the routine returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the LERAD routine. The register doesn't contain the logical-error indicator.

Figure 57.Contents of Registers When a LERAD Routine Gets Control

Figure 58 gives the logical-error return codes in the feedback field and explains what each one means.

RPLFDBK
Code **Condition**

4(4) End of data set encountered (during sequential retrieval). Either no EODAD routine is provided, or one is provided and it returned to VSAM and the processing program issued another GET.

Detected by: IDA019RA, IDA019RD, IDA019RR, IDA019RY
IDA019R2, IDA019R4, IDA019R8

8(8) Attempt was made to store a record with a duplicate key.

Detected by: IDA019RA, IDA019RQ, IDA019RX, IDA019R4, IDA019SE

12(C) Attempt was made to store a record out of ascending key sequence; record may also have a duplicate key.

Detected by: IDA019RA, IDA019RR, IDA019RX, IDA019R4, IDA019SE

16(10) Record not found.

Detected by: IDA019RA, IDA019RR, IDA019RY

20(14) Record already held in exclusive control by another requester.

Detected by: IDA019RF, IDA019RY, IDA019R2, IDA019R8

24(18) Record resides on a volume that can't be mounted.

Detected by: IDA019RW, IDA019RY, IDA019R2, IDA019R5, IDA019SE,
IDA019SL

28(1C) Data set cannot be extended because VSAM can't allocate additional direct-access storage space. Either there isn't enough space left in the data space for the secondary-allocation request or an attempt was made to increase the size of a data set by splitting the control area (high used RBA change) during processing with SHROPT=4 and DISP=SHR.

Detected by: IDA019RE, IDA019SE

32(20) An RBA was specified that doesn't give the address of any data record in the data set.

Detected by: IDA019RA, IDA019R8, IDA019SG

36(24) Key ranges were specified for the data set when it was defined, but no range was specified that includes the record to be inserted.

Detected by: IDA019RM

40(28) Insufficient virtual storage in the address space to complete the request.

Detected by: IDA019RG, IDA019RU, IDA019RX, IDA019SH, IGX00006

44(2C) Work area not large enough for the data record (GET with OPTCD=MVE).

Detected by: IDA019RR, IDA019RT, IDA019RY,
IDA019R4, IDA019R8

48(30) Invalid options, data set attributes, or processing conditions specified for TERMRPL request:

- CNV processing
- The specified RPL is asynchronous
- Chained RPL's
- PATH processing
- Shared resources (LSR/GSR)
- Create mode
- RRDS
- Data set contains spanned records
- User not in Key 0 and Supervisor state
- EOV in process (Secondary allocation)

Detected by: IDA019SN

Figure 58 (Part 1 of 4). Logical-Error Return Codes in the RPL Feedback Field from a Request Macro

RPLFDBK
Code **Condition**

52(34) The previous request was TERMRPL.

Detected by IDA019SN

64(40) As many requests are active as the number specified in the STRNO parameyer of the ACB macro; therefore, another request cannot be activated.

Detected by: IDA019RU, IDA019RX, IDA019R1

68(44) Attempt was made to use a type of processing (output or control-interval processing) that was not specified when the data set was opened.

Detected by: IDA019RQ, IDA019R4, IDA019R8

72(48) A keyed request for access was made to an entry-sequenced data set or a GETIX or PUTIX was issued to an entry-sequenced or relative record data set.

Detected by: IDA019R1, IDA019R8

76(4C) An addressed or control-interval PUT was issued to add a record to a key-sequenced data set, or a control-interval PUT was issued to a relative record data set.

Detected by: IDA019R1, IDA019R8

80(50) An ERASE request was issued for access to an entry-sequenced data set.

Detected by: IDA019RL, IDA019RX, IDA019R8

84(54) OPTCD=LOC was specified for a PUT request or in a request parameter list in a chain of request parameter lists.

Detected by: IDA019RQ, IDA019R1, IDA019R4, IDA019R8

88(58) A sequential GET or PUT request was issued without VSAM having been positioned for it, or a change was made from addressed access to keyed access without VSAM having been positioned for keyed sequential retrieval, or an illegal switch between forward and backward processing was attempted.

Detected by: IDA019RQ, IDA019RR, IDA019R4, IDA019R8

92(5C) A PUT for update or an ERASE was issued without a previous GET for update, or a PUTIX was issued without a previous GETIX.

Detected by: IDA019RQ, IDA019RX, IDA019R4, IDA019R8

96(60) Attempt was made to change a key during an update.

Detected by: IDA019RL, IDA019RX

100(64) Attempt was made to change the length of a record during an addressed update.

Detected by: IDA019RL, IDA019RQ

104(68) The RPL options are either invalid or conflicting in one of the following ways:

- SKP was specified and either KEY wasn't specified or BWD was specified
- BWD was specified for CNV processing
- FWD and LRD were specified
- Neither ADR, CNV, nor KEY was specified in the RPL
- WRTBFR, MRKBFR, or SCHBFR was issued, but either TRANSID was greater than 31 or a shared-resources option wasn't specified
- ICI processing was specified, but a request other than a GET or a PUT was issued

Detected by: IDA019RA, IDA019RR, IDA019RY, IDA019RX,
IDA019R1, IDA019R4, IDA0198

Figure 58 (Part 2 of 4). Logical-Error Return Codes in the RPL Feedback Field from a Request Macro

RPLFDBK Code	Condition
108(6C)	RECLEN specified was larger than the maximum allowed, equal to 0, smaller than the sum of the length and the displacement of the key field, or not equal to record (slot) length specified for a relative record data set. Detected by: IDA019RL, IDA019RQ, IDA019RU, IDA019R4, IDA019R8
112(70)	KEYLEN specified was too large or equal to 0. Detected by: IDA019R1
116(74)	A GET, POINT, ERASE, direct PUT, skip sequential PUT, or PUT with OPTCD=UPD not permitted during initial data-set loading (that is, for storing records in the data set the first time it's opened). Detected by: IDA019RR, IDA019R4, IDA019R8
132(84)	An attempt was made in locate mode to retrieve a spanned record. Detected by: IDA019RT
136(88)	An addressed GET was issued for a spanned record in a key-sequenced data set. Detected by: IDA019RT
140(8C)	Inconsistent spanned-record segments. Detected by: IDA019R4
144(90)	Invalid pointer in an alternate index (no associated base record). Detected by: IDA019RX
148(94)	The maximum number of pointers in the alternate index has been exceeded. Detected by: IDA019RU
152(98)	(Shared resources only.) Not enough buffers are available to process the request. Detected by: IDA019RY
156(9C)	An invalid Control Interval was detected during keyed processing. The invalid conditions possible are: 1. A key is not greater than the previous key. 2. A key is not in the current control interval. 3. A spanned record RDF is encountered. 4. A freespace pointer is invalid. 5. The number of records does not match a group RDF record count. Detected by: IDA019RA, IDA019RV, IDA019R4, IDA019S6

Figure 58 (Part 3 of 4). Logical-Error Return Codes in the RPL Feedback Field from a Request Macro

RPLFDBK
Code **Condition**

164(A4) Invalid options specified for CNVTAD/MNTACQ/ACQRANGE request:

- Generic key (GEN)
- Create mode
- Path processing
- User buffers (UBF) with LSR/GSR
- KSDS but not key processing (KEY)
- ESDS but not address processing (ADR)
- RRDS but not key processing (KEY)
- IMBED data set with only one level of index.

Detected by: IDA019R1

168(A8) User parameter list errors detected for CNVTAD/MNTACQ/ACQRANGE request:

- No user parameter list is specified (RPLARG=0)
- Argument count = zero for CNVTAD/MNTACQ request
- Ending argument is less than starting argument for ACQRANGE request
- Parameter list is not on word boundary.

Detected by: IDA019R1, IDA019SH

172(AC) ACQUIRE immediate errors returned by SVC 126 for MNTACQ/ACQRANGE request.

Detected by: IDA0192E

176(B0) Staging failure for MNTACQ/ACQRANGE request. (MSS hardware errors.)

Detected by: IDA019SM

180(B4) RBA/Volume error for MNTACQ/ACQRANGE request. (Required volume not mounted or specified RBA(s) not on mounted volume.)

Detected by: IDA019SL, IDA0192E

184(B8) Catalog errors returned from SVC 26 for CNVTAD request.

Detected by: IGX00006

188(BC) Storage in subpool 241 is not available for MNTACQ or ACQRANGE request.

Detected by: IDA0192E

192(C0) Invalid relative record number.

Detected by: IDA019RQ, IDA019RR

196(C4) An addressed request was issued to a relative record data set.

Detected by: IDA019R1

200(C8) Addressed or control-interval access was attempted by way of a path.

Detected by: IDA019RX

204(CC) PUT-insert requests are not allowed in backward mode.

Detected by: IDA019RQ, IDA019R4

208(D0) Invalid ENDREQ request.

Detected by: IDA019SM, IDA019RP

Figure 58 (Part 4 of 4). Logical-Error Return Codes in the RPL Feedback Field from a Request Macro

Field	Bytes		Length	Discussion
Message	91	105	15	Messages are divided according to ECB condition codes: X'41'— "UNIT EXCEPTION" "PROGRAM CHECK" "PROTECTION CHK" "CHAN DATA CHK" "CHAN CTRL CHK" "INTFCE CTRL CHK" "CHAINING CHK" "UNIT CHECK" *If the type of unit check can be determined, this message is replaced by one of the following:* "CMD REJECT" "INT REQ" "BUS OUT CK" "EQP CHECK" "DATA CHECK" "OVER RUN" "TRACK COND CK" "SEEK CHECK" "COUNT DATA CHK" "TRACK OVERRUN" "CYLINDER END" "INVALID SEQ" "NO RECORD FOUND" "FILE PROTECT" "MISSING A.M." "OVERFL INCP" X'48'—"PURGED REQUEST" X'4F'—"R.HA.RO. ERROR" For any other ECB completion code—"UNKNOWN COND."
	106		1	Comma (,)
Physical Direct-Access Address	107	120	14	BBCCHHR (bin, cylinder, head, and record)
	121		1	Comma (,)
Access Method	122	127	6	"VSAM"

Figure 61 (Part 2 of 2). Format of Physical-Error Messages

Open and Close Return Codes

When a processing program receives control after it has issued an OPEN or CLOSE macro, register 15 indicates whether all of the data sets were opened or closed successfully:

Reg 15	Condition
0(0)	All data sets were opened or closed successfully.
4(4)	Open: all data sets were opened successfully, but one or more warning messages were issued (ACBERFLG codes less than X'80'). Close: at least one data set (VSAM or nonVSAM) was not closed successfully.
8(8)	Open: at least one data set (VSAM or nonVSAM) was not opened successfully; if there was an error for an ACB, it was restored to the contents it had before OPEN was issued.
12(C)	Open: at least one data set (VSAM or nonVSAM) was not opened successfully; if there was an error for an ACB, it was not restored to the contents it had before

OPEN was issued (and the data set cannot be opened without the ACB's being restored).

ACBERFLG Code	Condition
0(0)	When register 15 contains 0: All data sets were opened or closed successfully When register 15 contains 8: Either VSAM is processing the ACB for some other request, Or DDNAME was not specified in the ACB
4 (4)	Warning message: the ACB is already opened (and the user issued OPEN), or the ACB is already closed (and the user issued CLOSE or temporary CLOSE).
96 (60)	Warning message: an unusable data set was opened for input. Detected by: IDA0192B
100 (64)	Warning message: Open encountered an empty alternate index that is part of an upgrade set. Detected by: IDA0192B
104 (68)	Warning message: the timestamp for the volume does not match the timestamp in the catalog record for the data set. (This may mean the cluster existing on the volume(s) is not accurately described by its catalog record.) Detected by: IDA0192A
108 (6C)	Warning message: the timestamp for the index is less than the timestamp for the data set. (This could occur if the data set was updated without the index being open.) Detected by: IDA0192B
116 (74)	Warning message: the last request to close this data set was not completed successfully. Detected by: IDA0192B
128 (80)	DDNAME not found in TIOT.
132 (84)	An I/O error was detected while the system was reading the JFCB. Detected by: IDA0192F, IDA0200V
136 (88)	Not enough storage was available for work areas, buffers, or control blocks. Detected by: IDA0192A, IDA0192B, IDA0192C, IDA0192F, IDA0192W, IDA0192Y, IDA0192Z, IDA0200B, IDA0200T, IDA0231B, IDA0231T
144 (90)	An I/O error occurred while a catalog record was being read or written. A return code was set by a VS2 Catalog-Management routine. Detected by: IDA0192C
148 (94)	The catalog entry for the data set being opened or closed was not found or an unidentified error occurred while VSAM was searching the catalog. Detected by: IDA0192C
152 (98)	The data set being opened is protected by a password, and the VSAM Open routine was unable to validate the password or an unauthorized program is attempting to open a catalog as a data set. Detected by: IDA0192C
160 (A0)	The buffer space specified was not consistent with the buffer requirements of the data set; or the ACB indicated keyed access,

ACBERFLG Code	Condition
	but the data set is not a key-sequenced data set; or the device type specified in the DD statement is not consistent with the device type indicated in the catalog entry for the data set; or user buffering is specified in the ACB's MACRF field and control-interval processing should be specified, but is not. Detected by: IDA0192A, IDA0192B, IDA0192C, IDA0192Z

ABEND	Error	Error detected by	ABEND issued by	Error indication set in DCB or DECB by
1(001)	The user did not specify a SYNAD exit routine.			
(a)	I/O error	VSAM initially and BISAM during CHECK	BISAM (IDAIIPM3)	SYNAD (DECB) (IDAIISM1)
(b)	Invalid request	BISAM	BISAM	BISAM (DECB)
49(031)	The user did not specify a SYNAD exit routine.			
(a)	VSAM physical or logical error	VSAM	SYNAD	SYNAD
(b)	Invalid request	VSAM	SYNAD	GET and SETL routines of SCAN (IDAIIPM2)
(c)	Sequence check	LOAD (IDAIIPM1)	LOAD	RESUME routine of LOAD
(d)	Length error (RDW greater than LRECL)	LOAD	LOAD	LOAD
57(039)	End of data without EODAD routine	VSAM	SCAN (IDAIIPM2)	EODAD routine of SCAN
58(03A)	VSAM ACB closed with warning messages	CLOSE (IDA0200S)	CLOSE	CLOSE
59(03B)	Validity check	OPEN (IDA0192I)	OPEN	Validity-check routine of OPEN

Catalog values and DCB values for LRECL, KEYLE, or RKP don't correspond, or, with QISAM, DISP is specified OLD when the data set is being opened for output, and there are already records in the data set (implying RELOAD).

Figure 67. ISAM-Interface ABENDs

Exception codes may be set in the DCB (for QISAM processing) or the DECB (for BISAM processing) in connection with ISAM-Interface ABENDs. Figures 68 and 69 give the exception codes. Except where indicated, register 15 contains 8, for logical errors.

DECB exception code	**Explanation**	**Corresponding RPL feedback code(s)**	**Explanation**	**Error detected By**
DECBEXC1				
1...	Record not found	16(10)	Record not found	VSAM
		24(18)	Record on unmountable volume	VSAM
.1..	Record-length check	108(6C)	Record-length check	VSAM
..1.	Space not found	28(1C)	Data set not extendable	VSAM
...1	Invalid request	none	No RPL available	ISAM Interface
		20(14)	Exclusive-control conflict	VSAM
		36(24)	No key range defined for insertion	VSAM
		64(40)	Placeholder not available	VSAM
		96(60)	Key-change attempted	VSAM
.... 1...	Uncorrectable I/O error	4-24 (04-18)	A physical error (Register 15 contains 12(0C))	VSAM
.... .1..	Unreachable block	—	A logical error not covered by another exception code	VSAM
.... ..1.	Overflow record (indicated for all READ requests)	none		ISAM Interface
....1	Duplicate record	8(08)	Duplicate record	VSAM
DECBEXC2				
xxxx xx..	Reserved (always 0)	none		
.... ..1.	Channel program initiated by an asynchronous routine (never indicated, always 0)	none		
....1	Previous macro was READ KU	none		ISAM Interface

Figure 68. BISAM Exception Codes in Relation to VSAM Return Codes