IBM Technical Newsletter

This Newsletter No. SN26-0909
Date April 28, 1978

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

Prerequisite Newsletters None

OS/VS2 Virtual Storage Access Method (VSAM) Logic

This technical newsletter, a part of Release 3.7 of OS/VS2, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and removed are:

cover-7
13 - 16.1 (16.1 added)
89-92
111,112
221-226
275-288.1 (288.1 added)
379-384
419-424

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

Summary of Amendments

This technical newsletter updates the publication to document technical changes for Control Interval Split.

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 Virtual Storage Access Method (VSAM) Logic

Release 3.7

IBM

Second Edition (January 1976)

This edition, as amended by technical newsletter SN26-0895 and SN26-0909, applies to Release 3.7 of OS/VS2 and to any subsequent releases of that system unless otherwise indicated in new editions or technical newsletters.

Significant system changes are summarized under "Summary of Amendments" following the list of illustrations. In addition, miscellaneous editorial and technical changes have been made throughout the publication.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/370 Bibliography*, GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose California 95150. All comments and suggestions become the property of IBM.

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/VS Data Management Macro Instructions,* GC26-3793
- *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 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."

CONTENTS

SUMMARY OF AMENDMENTS

Technical 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 duplicated 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

Enhanced VSAM

VSAM has several new functions and data structues for the independent component release of VS2 Release 3: alternate indexes, Checkpoint/Restart processing, spanned records, relative record data sets, processing the index of a key-sequenced data set, shared resources among data sets, improved control-interval processing, backward sequential processing, catalog recovery, and virtual-storage management. These additions to VSAM change this logic manual in all its sections: method of operation diagrams (HIPOs), program organization figures (compendiums), directories, data areas, and diagnostic aids. The directories identify all the new modules and external procedures and indicate which HIPOs and compendiums refer to them.

Method of operation diagrams have been added for Data-Set Management to document recovery-termination processing.

The detailed descriptions of some control blocks, for which you were previously referred to *OS/VS2 Data Areas*, are included in this book.

The index has been expanded to include more proper names:

- Modules and external procedures are included with a page reference to the "Directory."
- Internal procedures and program instruction labels are included with page references to the pages where may appear.

Alternate Indexes

Alternate indexes for key-sequenced and entry-sequenced data sets add control blocks and complicate control block interrelationships. Opening and closing a path (a base cluster and the alternate index through which access is gained to it) more than double the number of HIPOs for Data-Set-Management. Access by way of a path and alternate-index upgrading change and add HIPOs to Record Management.

Checkpoint/Restart Processing

Checkpoint/Restart processing changes four HIPOs, and adds five HIPOs, two program organization figures, and four control blocks.

Spanned Records

Having data records longer than one control interval changes a number of HIPOs in Record Management. It changes the contents of control information in the RDFs in a control interval.

Relative Record Data Set

The relative record data set brings to three the number of types of VSAM data sets. It changes the contents of control information in the RDFs in a control interval. It changes HIPOs and adds a HIPO to Record Management.

Processing the Index of a Key-Sequenced Data Set

User access to the control intervals of a prime index changes HIPOs in Record Management to include the GETIX and PUTIX macros.

Shared Resources among Data Sets

Shared buffers, I/O-related control blocks, and channel programs among data sets for processing add control blocks and change control block interrelationships. Building and deleting a VSAM resource pool add a HIPO to Data-Set Management for the BLDVRP and DLVRP macros add a section to "Diagnostic Aids" to describe recovery with global shared resources. Managing I/O buffers adds HIPOs to Record Management for the MRKBFR, WRTBFR, and SCHBFR macros.

Improved Control-Interval Processing

Improved control-interval processing changes HIPOs in Record Management to show the bypassing of certain functions for faster processing.

Backward Sequential Processing

Backward sequential processing changes HIPOs in Record Management to include processing data records in descending sequence by RBA or key.

Catalog Recovery

The optional recovery function that enables users to recover or restore data sets changes HIPOs slightly in Data-Set Management and Record Management.

Virtual-Storage Management

The management of virtual storage has been centralized in Virtual-Storage Management, which controls most requests for storage. It adds control blocks, which are described in "Virtual-Storage Management" in "Diagnostic Aids."

Release 3

Staging and destaging of data between mass storage and direct-access storage is added for the IBM 3850 Mass Storage System.

Release 2

I/O Management

For communication with the VS2 I/O Supervisor, the IOB control block is replaced by a set of three control blocks: IOMB, IOSM, and SRB.

Interface Between VSAM Record Management and VS2 I/O Supervisor. The function of putting together a channel program for issuing STARTIO has been separated from Record Management to stand logically as an interface between Record Management and the VS2 I/O Supervisor.

Interface Between VS2 Auxiliary Storage Management and I/O Supervisor. VSAM I/O Management serves the same function for paging I/O between real storage and external page storage. It is the programming interface between the VS2 Auxiliary Storage Manager and I/O Supervisor.

Security and Integrity

OS/VS2 multiprocessing requires changes in the way serially reusable resources are shared. A scheme of software locks that programs must obtain and free in order to use certain resources replaces hardware disabling. The CS instruction (compare and swap) is also used to ensure the integrity of serially reusable resources. ENQ/DEQ and the TS instruction (test and set) are still used in OS/VS2, Release 2.

I/O Management, Data-Set Management (Open and Close, for both VSAM and the ISAM Interface), and End of Volume (considered logically as part of Record Management) use the local memory lock to protect VSAM control blocks when chaining them. They obtain or release the local memory lock with the SETLOCK macro. Data-Set Management also uses the CS instruction when chaining DEB control blocks or modifying UCB information.

Most of the processing of I/O Management is in supervisor mode. It uses the MODESET macro to swap storage-protection keys. Data-Set Management runs primarily in storage-protection key 0 (as does End of Volume). ISAM-Interface Open and Close run primarily in the user's key. These modules use the MODESET macro to swap keys when transferring control to and from OS/VS Open and Close. Record Management (with the exception of End of Volume) continues to run in the user's key, as in OS/VS1 and Release 1 of OS/VS2.

VSAM protects crucial I/O control blocks by placing them in protected subpools. A user of VSAM cannot modify these control blocks and cannot, therefore, interfere with the operation of the system. VSAM Open and Close work with copies of ACB control blocks to prevent a user from interfering with the system. ISAM-Interface Open and Close don't work with copies since they run in the user's storage-protection key and thus provide no system processing to be interfered with.

Recovery and Termination

OS/VS2's philosophy of recovery is to avoid relPLing the system and to free up resources claimed by a failing task in order to be able to continue processing without contending with fragmented resources. The system must be able to reclaim actual resources and slots in control tables.

The modules of I/O Management have functional recovery routines that get control from VS2 Recovery Termination Manager when an error occurs in I/O Management. A recovery routine releases various resources (such as the local memory lock, if it has been obtained) and may cause information to be recorded in SYS1.DUMP (with the SDUMP macro) or in SYS1.LOGREC (with the SETRP macro).

Data-Set Management and End of Volume share a recovery routine that gets control from the VS2 I/O Support Recovery Routine (which is an ESTAE routine) when an error occurs during the processing of these modules, and they share a Task Close Executor that gets control from VS2 Task Close for task or memory termination.

The recovery routine causes information to be recorded in SYS1.DUMP and SYS1.LOGREC concerning the processing that preceded an error. Open, Close, and End of Volume have been altered to leave various audit information in the Open/Close/End-of-Volume Work Area for this purpose.

The Task Close Executor frees up storage in the system area.

ISAM-Interface Open and Close also have a recovery routine that runs under control of the ESTAE routine mentioned above. This routine frees up ISAM Interface work areas following errors from which recovery cannot be made.

Page-Space Preformatting

VSAM in Release 2 of OS/VS2 recognizes the special case of a page-space data set being opened for output. When it occurs, VSAM makes the calculations required for preformatting, then transfers control to the Control-Area Preformat routine of Record Management to preformat all of the control areas that comprise the page-space data set.

Notes for Diagram BC

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 based on the following considerations:

- For skip-sequential processing, the index search starts at the sequence set—normally 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 into the RBA of the control interval that contains the record, plus the offset of the record in the control interval.

IDA019RR calls **IDA019RZ (IDAGRB)**

The control interval is read in by RBA.

Addressed Processing

IDA019RA

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

IDA019R4 calls **IDA019S6**

If a CI is encountered that indicates a CI split was interrupted, IDA019S6 is called to correct the problem.

2 This step doesn't apply to a relative record data set.

3 **IDA019R4** calls **IDA019RT (IDADARTV)**

A spanned record is retrieved.

IDADARTV calls **IDA019RZ (IDAFREEB)**

A segment is moved to the user's area. The buffer is freed.

IDADARTV calls **IDA019RZ (IDAGNXT)**

The next segment is obtained.

4 **IDA019R4**

If the user is performing locate processing, the address of the record is moved into the user area.

If the request is for update and an upgrade set exists, IDA019RU is called to save the LLOR (least length of the record that contains all key fields). (See Diagram BR.)

IDA019RA calls **IDA019S6**

If a CI is encountered that indicates a CI split was interrupted, IDA019S6 is called to correct the problem.

Relative Record Processing

IDA019RR

If the user is performing locate processing, the address of the record is moved into the user's area.

5 **IDA019R4: RLSEBUFS** (calls **IDA019RZ**)

If the request is direct and neither update, note-string-position, nor locate mode is specified, the contents of the buffer are not logically needed by the next request and the buffer is released. If the user is processing with shared resources, any index buffer is freed.

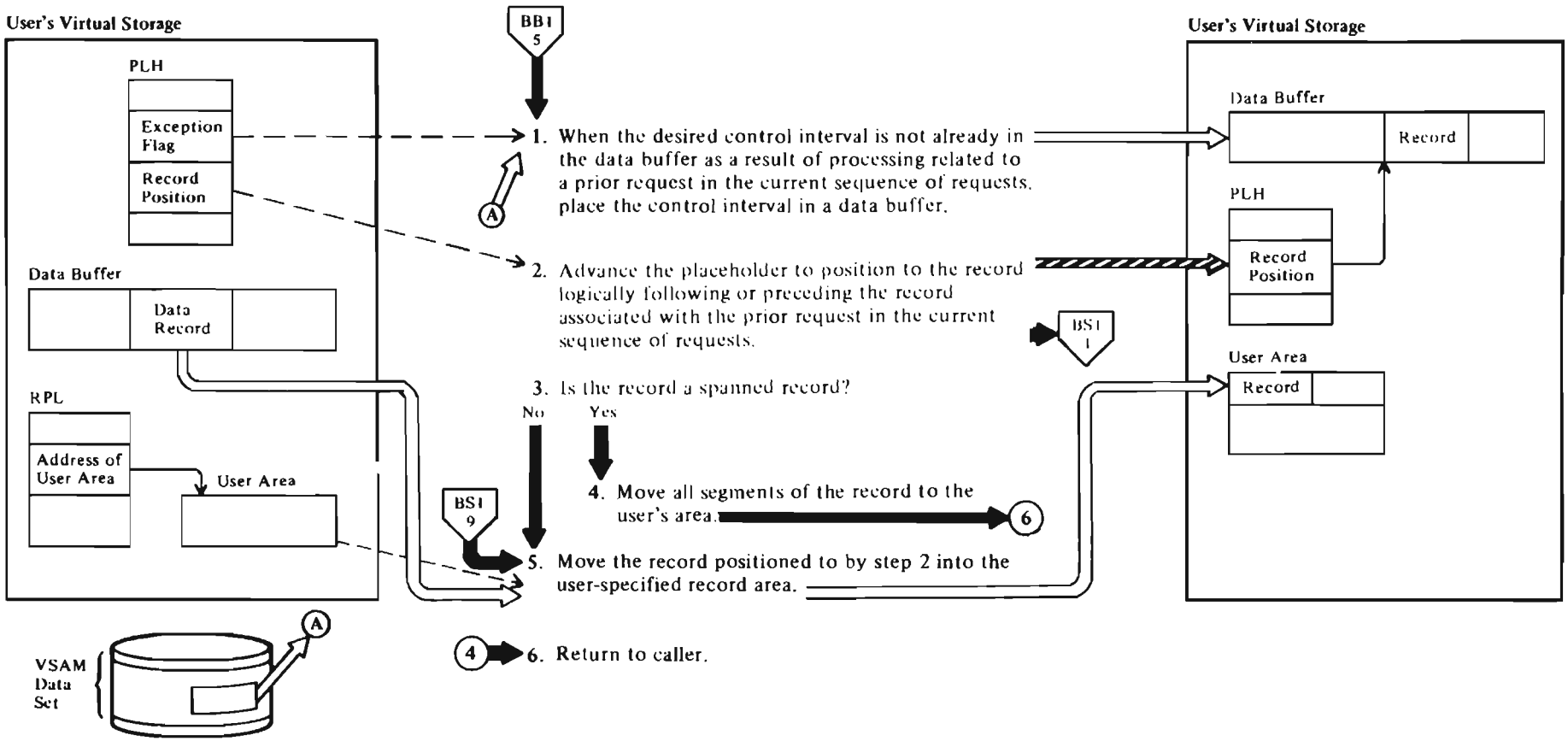
Relative Record Processing

IDA019RR calls **IDA019RZ (IDAFREEB)**

If the request is direct and neither update, note-string-position, nor locate mode is specified, the buffer is released.

Page of SY26-3825-1
Revised April 28, 1978
By TNL SN26-0909

Diagram BD. GET-Sequential Processing: Sequential Retrieval



Notes for Diagram BD

Key-Sequenced or Entry-Sequenced Data Set

1 IDA019R4

2 Forward Processing

IDA019R4: ADVPLH

Normal GET-sequential processing advances the record pointer to the next record in physical sequence in the data buffer.

If the advance positions the record pointer to the end of a control interval, the following processing is performed:

IDA019R4 calls **IDA019RZ (IDAFREEB)**

The current buffer is released.

IDA019R4 calls **IDA019RZ (IDAGNXT)**

The next control interval is retrieved. If the next control interval contains all free space, the retrieval process continues until a control interval containing data is acquired.

IDA019RA calls **IDA019S6**

If a CI is encountered that indicates a CI split was interrupted, IDA019S6 is called to correct the problem.

Backward Processing

IDA019R4 calls **IDA019RV (IDAADVPH)**

Normal processing advances the record pointer to the preceding record in RBA sequence in the data buffer.

IDA019RV calls **IDA019S6**

If a CI is encountered that indicates a CI split was interrupted, IDA019S6 is called to correct the problem.

If the record pointer points to the beginning of a control interval, the following processing takes place:

IDAADVPH calls **IDA019RZ (IDAFREEB)**

The current control interval is released.

IDAADVPH calls **IDA019RZ (IDAGNXT)**

The preceding control interval is retrieved.

4 IDA019R4 calls **IDA019RT (IDADARTV)**

A spanned record is retrieved.

IDADARTV calls **IDA019RZ (IDAFREEB)**

A segment is moved to the user's area. The buffer is freed.

IDADARTV calls **IDA019RZ (IDAGNXT)**

The next segment is obtained.

5 IDA019R4: DATARTV

If the request is for update and an upgrade set exists, IDA019RU is called to save the LLOR (least length of the record that contains all key fields). (See Diagram BR.)

Relative Record Data Set

1 IDA019RR

The data buffer contains the current control interval.

2 IDA019RR: ADVPLH

The record pointer is advanced for normal sequential processing or backed up for backward sequential processing. If the record pointer points to the end of the control interval for normal processing or the beginning of the control interval for backward processing, the following processing takes place:

IDA019RR calls **IDA019RZ (IDAFREEB)**

The current buffer is released.

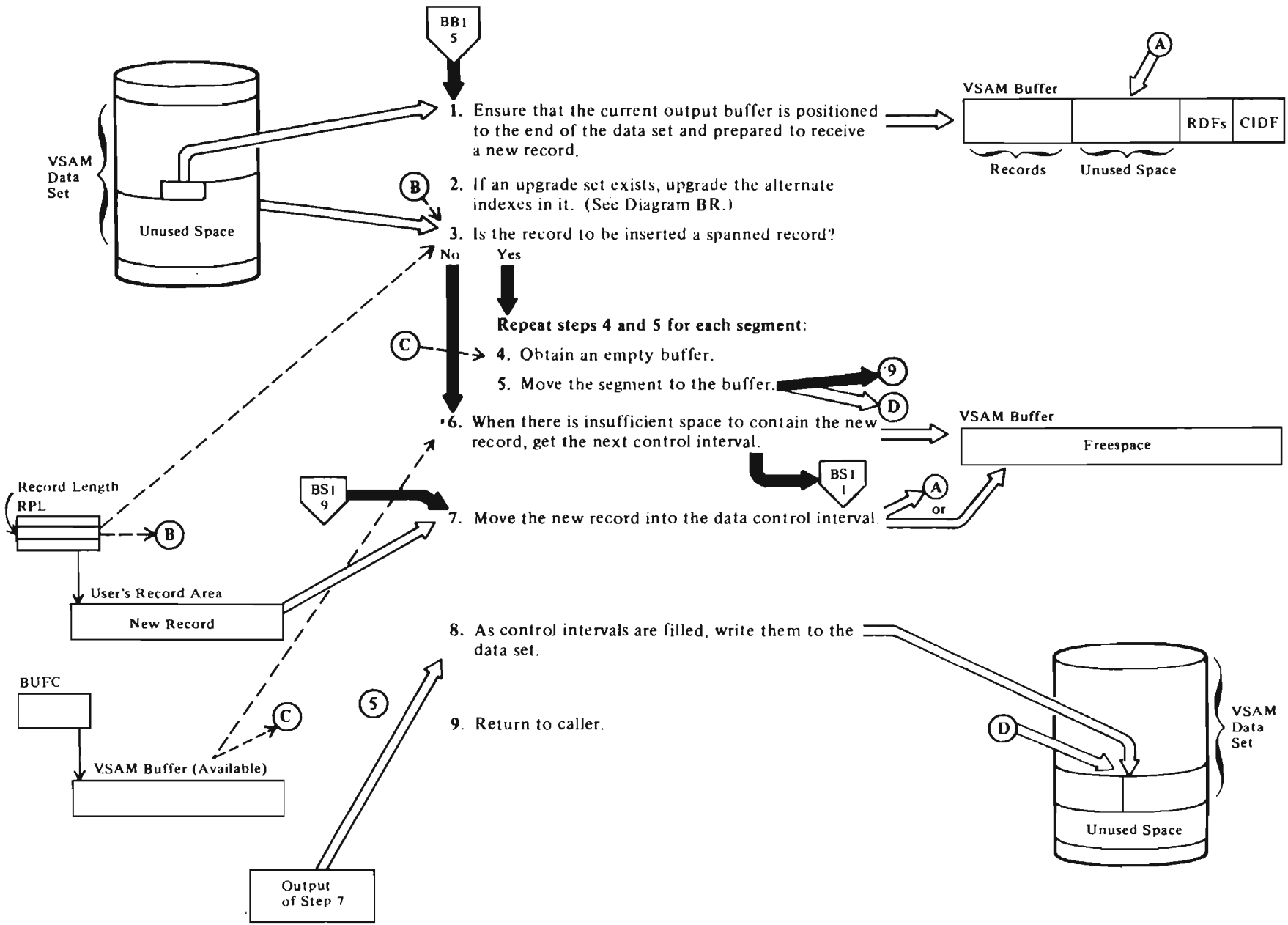
IDA019RR calls **IDA019RZ (IDAGNXT)**

For normal processing, the next sequential control interval is retrieved, and the record pointer is set to the first record. For backward processing, the preceding sequential control interval is retrieved, and the record pointer is set to the last record.

5 IDA019RR

Page of SY26-3825-1
Revised April 28, 1978
By TNL SN26-0909

Diagram BE. PUT-Entry-Sequenced Processing: Create or Mass Insert



Notes for Diagram BH3

24 IDA019RE calls **IDA019RZ (IDAGRB)**

When the current sequence-set index record has been updated by another request since it was last read, it must be reread.

IDA019RE calls **IDA019RZ (IDAFREEB)**

When the current data control interval has been updated by another request since it was last written, it must be rewritten to preserve those updates from possible loss.

25

If the record is to be inserted at the end of a control interval or if it is one of a sequence of records to be inserted at the beginning of a control interval, the control interval is not split and the record is placed in the next control interval currently containing freespace.

If the request is a direct request to insert a record at the beginning of the control interval or if it is either a direct or sequential request to insert a record at some point other than the beginning or end of the control interval, the control interval must be split.

If the request is a sequential request, the control interval is split at the point where the data record is to be inserted.

If the request is a direct request, the record boundary nearest to the midpoint of the control interval is used as the split point.

The RDFs are divided among the control intervals so that they remain associated with their respective records.

IDA019RE calls **IDA019RZ (IDAGNFL)** and **IDA019RE (BUILDFS)**

A work buffer is obtained, converted to freespace, and attached to the data insert work area (DIWA). The work buffer is used to perform the record insertion processing.

IDA019RE

Records to the right of the split point in the old control interval are moved into the new freespace control interval. Then the moved records are zeroed-out in the old control interval and the freespace pointers in each control interval's CIDF are adjusted.

26 IDA019RE calls **IDA019RF**

27 IDA019RE calls **IDA019RM**

28 IDA019RE calls **IDA019RH**

The new index entry reflects the high key of the data records within the new data control interval. If the new index entry fits in the index record, the buffer that contains the record is not written to the index until the new data control interval is written to the data set.

29 IDA019RE calls **IDA019RZ (IDAWRBFR)**

When a control interval split occurs, the old data control interval is written with the CIDFBUSY bit on.

IDA019RE calls **IDA019RZ (IDAWRBFR)**

The new data control interval residing in the work buffer associated with the DIWA is written.

IDA019RE calls **IDA019RH (IXIDAWR)**

The updated index record residing in the index buffer associated with the current placeholder is written.

IDA019RE calls **IDA019RZ (IDAWRBFR)**

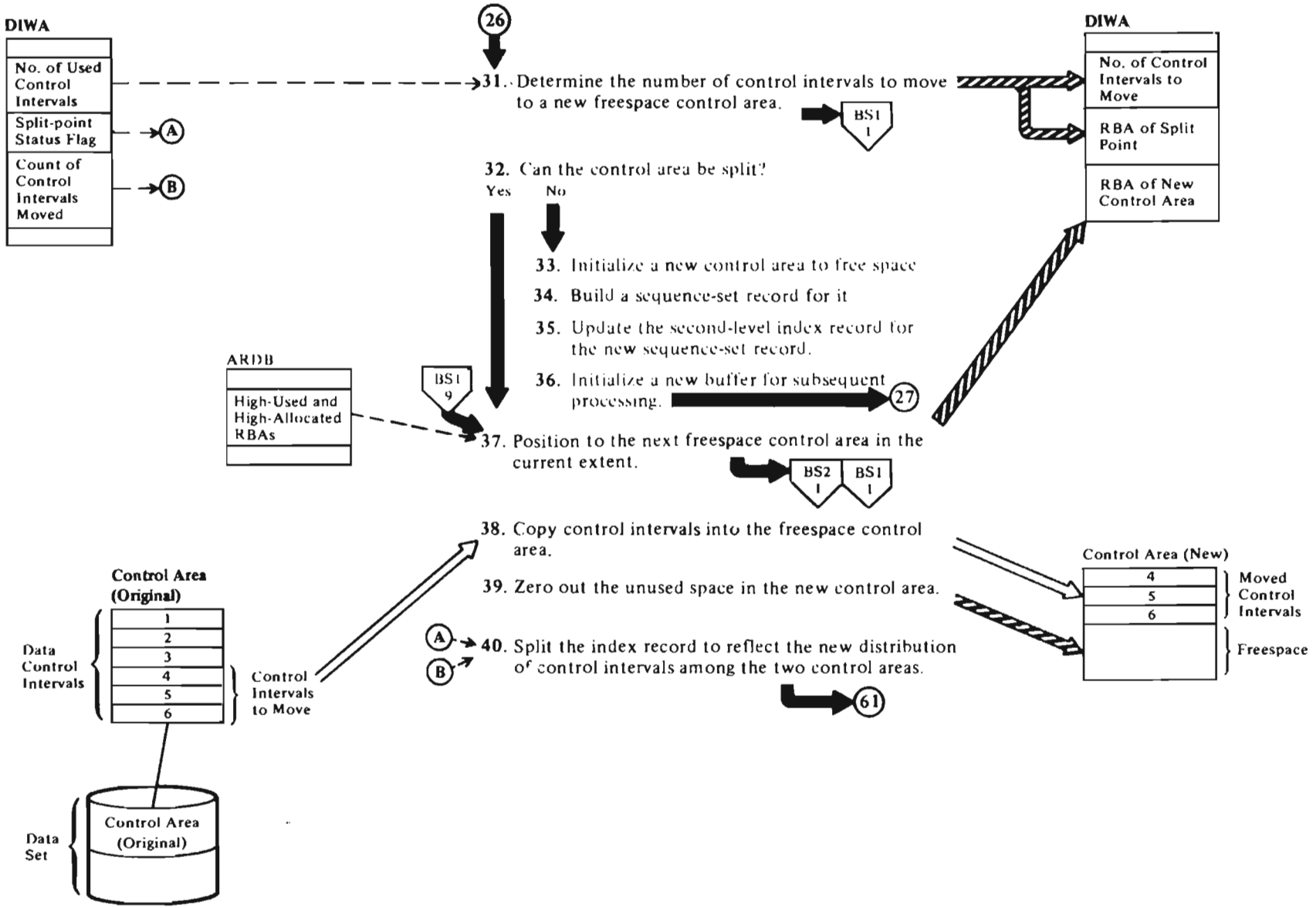
When a control interval split occurs (see note 25), the old data control interval associated with the current placeholder is written with the CIDFBUSY bit off.

30 If the record insertion is unsuccessful after the control interval has been split, a second pass results in a successful insertion—IDA019R4 has verified that the record fits in a control interval.

Page of SY26-3825-1
Revised April 28, 1978
By TNL SN26-0909

Diagram BH4. Modifying a Key-Sequenced Data Set

Split a Control Area to Create Freespace and to Generate a New Index Record



Notes for Figure 20

1 IDA019R1 is the common Record Management request module. It verifies that the request is a valid Record-Management macro, then tests the RPL for keyed or addressed processing.

2 When the request requires either keyed or addressed processing (not a control-interval-processing request), IDA019R4 selects the correct processing path for either GET, PUT, or POINT, and for sequential, skip sequential, or direct processing.

3 When the request is either direct GET or skip sequential GET, IDA019RA locates the position of the desired data record in its control interval.

4 DATARTV makes an unspanned data record available to the caller. It sets the RBA of the data record into the RPL. If the caller's request is in locate mode, DATARTV returns a pointer to the record to the caller. If the request is in move mode, DATARTV moves the data record into the caller's record area.

5 IDADARTV moves all of the segments of a spanned record into the user's area.

6 IDAFREEB frees the buffer.

7 IDAGNXT moves the next segment into a buffer.

8 If the request is direct GET and the caller doesn't want to retain the record's position for subsequent record processing requests, RLSEBUFS releases the data record's buffer.

9 If the buffer was changed by a previous update request, IDAWRBFR rewrites the buffer's control interval into the data set.

10 See Figure 40.

11 IDAFREEB frees the data buffer.

12 If the request is keyed, IDAFREEB frees the buffer containing the sequence set control interval associated with the data buffer.

13 If the user's EXLST contains an active journal exit address, IDATJXIT provides the necessary journaling information for the user's journal exit routine.

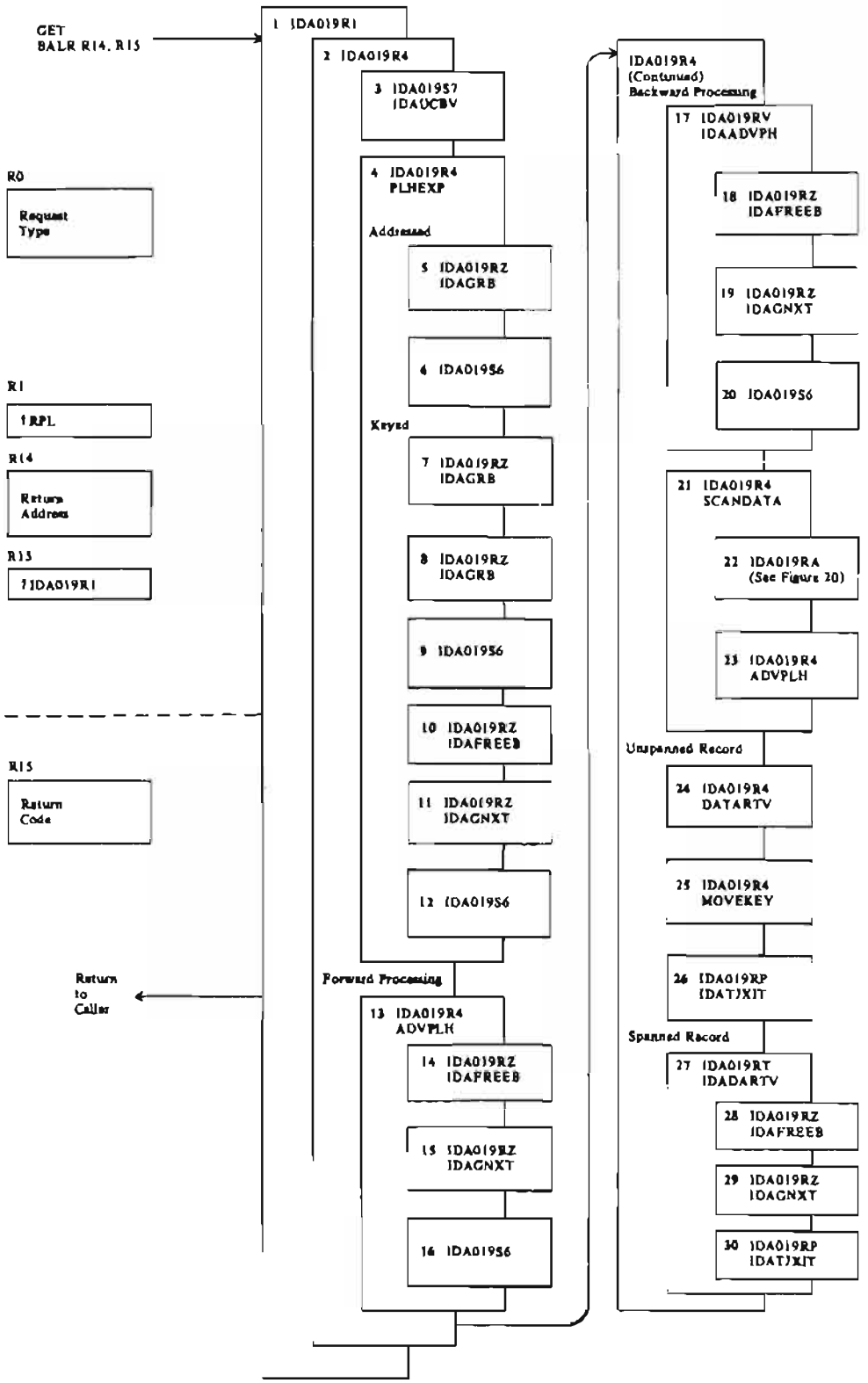


Figure 21. GET: Sequential Processing (ESDS, KSDS)

Notes for Figure 21

1 IDA019R1 is the common Record-Management request module. It verifies that the request is a valid Record-Management macro, then tests the RPL for keyed or addressed processing.

2 When the request requires either keyed or addressed processing (not a control-interval-processing request), IDA019R4 selects the correct processing path for either GET, PUT, or POINT, and for sequential, skip sequential, or direct processing.

3 When the request is sequential GET, PLHEXP tests the status indicators in the placeholder (PLH) to determine if an exceptional condition occurred:

- If the request is the first request after the data set is opened, and isn't preceded by a POINT to position to a starting record:

4 If the request is addressed, IDAGRB reads in the first data control interval of the data set.

5 IDA019S6 is called if the CIDFBUSY bit has been set in the CIDF indicating a CI split was interrupted or is still in progress for this CI. IDA019S6 will set a return code and reason code in the RPL if the CI cannot be repaired.

6 If the request is keyed, IDAGRB reads in the first sequence set control interval.

7 The sequence set control interval is used to determine the RBA of the first data control interval. IDAGRB retrieves the first data control interval of the key-sequenced data set.

8 See step 5.

9 If the first control interval of the key-sequenced data set is empty, IDAFREEB frees its buffer.

10 IDAGNXT obtains the next control interval of the key-sequenced data set. Steps **10** and **11** are repeated as often as necessary to obtain a nonempty control interval of the key-sequenced data set.

11 See step 5.

- If the end of data condition occurs, PLHEXP sets a return code and returns to the caller.
- If a read error occurs, ADVPLH skips over the bad data, resets the PLH so that it points to the next good data control interval's RBA, and returns to the caller with a return code set.
- If the previous request encountered a read-exclusive error (not allowed to read the record because another user has exclusive control over it), SCANDATA searches the index to locate the requested record.

12 If no exceptional conditions have been detected, the PLH now points to the record most recently processed by the user. ADVPLH adjusts the PLH so that it points to the next record (desired by this request) in the buffer.

13 If there are no more records in the buffer (that is, the record most recently processed by the user is the control interval's last record), IDAFREEB frees the buffer.

14 IDAGNXT retrieves the next sequential control interval, unless another buffer already contains the control interval. The PLH is set to point to the first data record in the control interval.

15 See step 5.

16 If no exceptional conditions have been detected, the PLH now points to the record most recently processed by the user. IDAADVPH adjusts the PLH to point to the previous record (desired by this request) in the buffer.

17 If there are no more records in the buffer (that is, the record last processed by the user is the control interval's first record), IDAFREEB frees the buffer.

18 IDAGNXT retrieves the next control interval in descending sequence, unless another buffer already contains the control interval. The PLH is set to point to the last record in the control interval.

19 See step 5.

20 If the current request is GET-for-update, but the record's buffer is not under the caller's exclusive control, SCANDATA locates the record again to ensure that the PLH now points to it, even though updates might have occurred against it. The buffer is now under the caller's exclusive control.

21 IDA019RA searches the index, if the data set is key-sequenced, or uses the caller-supplied RBA, if the data set is entry-sequenced, to determine the record's location in the buffer.

22 If the placeholder needs to be updated, ADVPLH updates it after the record has been located.

23 DATARTV makes an unspanned data record available to the caller. It sets the RBA of the data record into the RPL. If the caller's request is in locate mode, DATARTV returns a pointer to the record in the caller's RPL. If the request is in move mode, DATARTV moves the data record into the caller's record area.

24 MOVEKEY saves the record's key in the placeholder.

25 If the user's EXLST contains an active journal exit address, IDATJXIT provides the necessary journaling information for the user's journal exit routine.

26 IDADARTV moves all the segments of a spanned record into the user's area.

27 IDAFREEB frees the buffer.

28 IDAGNXT moves the next segment into a buffer.

29 See the note for step 20.

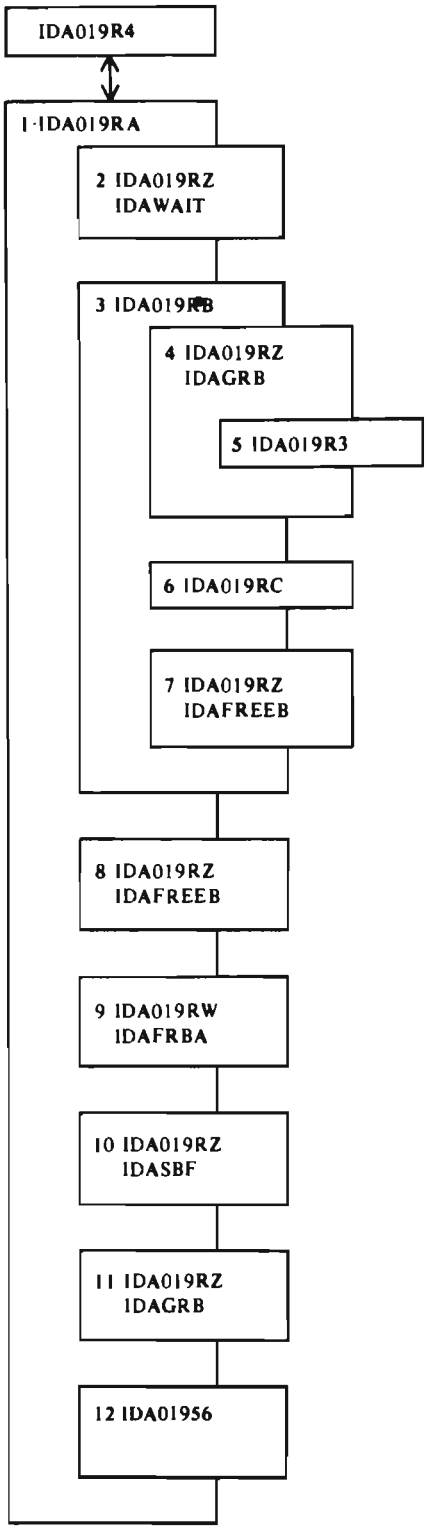


Figure 22. Obtain the Control Interval Containing a Specified Record and Find the Position of the Record in the Control Interval (ESDS, KSDS)

Notes for Figure 22

1 IDA019RA locates the position of a desired data record in a control interval that is in a VSAM Record-Management buffer.

2 If the control interval is being updated by another user, IDAWAIT waits until the updating is complete.

2 If the data set is key-sequenced, IDA019RB searches the index to find the RBA of the desired data record's control interval.

4 IDAGRB obtains an index record to search.

5 See Figure 40.

6 IDA019RC searches the index control interval to locate an index entry containing a key value equal to or greater than the search argument passed by IDA019RB. IDA019RC sets a return code to indicate the status of the search, and a pointer to the requested entry, if found.

7 If IDA019RC hasn't found the termination point for the search (determined by IDA019RB), IDAFREEB releases the buffer containing the just-searched index control interval. **4** through 7 repeat until the termination point for the search is reached.

8 If the placeholder doesn't point to the buffer containing the desired data record, IDAFREEB frees the buffer currently pointed to by the PLH.

9 IDAFRBA determine the RBA of the next sequential (or, if the request is keyed, the next higher keyed) control interval.

10 IDASBF releases all buffers (except one) pointed to by the placeholder—buffers that have been assigned to the placeholder and available for its use, but are not currently in use.

11 IDAGRB retrieves the data record's control interval, located by the previous index search if the data set is key-sequenced or by the caller-specified RBA value if the data set is entry-sequenced.

12 IDA019S6 is called if the CIDFBUSY bit is set on in the CIDF (indicating a CI split was interrupted or is still in progress) for this CI. IDA019S6 will set a return code and reason code in the RPL if the CI cannot be repaired.

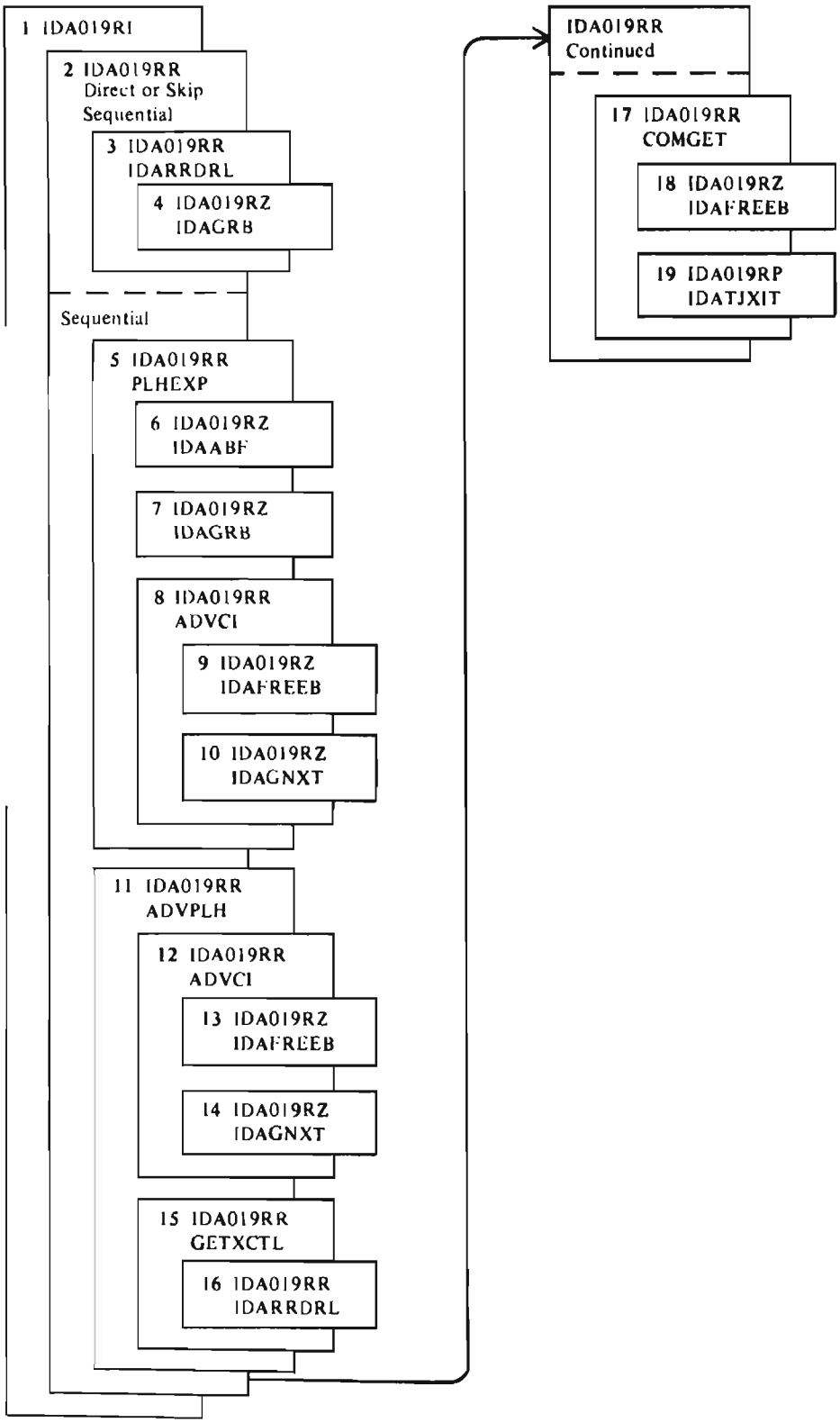


Figure 23. GET Processing (RRDS)

Module Directory

Module Name	Descriptive Name	External Procedure Names	Component	Method of Operation Diagrams	Program Organization Figures
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, BT28,	12, 14, 18, 39
IDA0192F	Open Base Cluster, Path, and Upgrade Alternate Index	IDA0192F	O	AC3, AC4, AC5,8,18 AC6	
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, AE2,8, AH3	12, 14, 18, 39
IDA0192S	VSAM Open/Close/EOV: SMF Record Build	IDA0192S	O	AC7, AD6, AE28,	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, AC5,8, AD5, AF1, AH3, AL1, AL2	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
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

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
F3FRR	IDA121A3	Functional Recovery Routine (of IDA121F3)	DA4	41
IDAABF	IDA019RW	Buffer Management: Add Buffer to Placeholder (PLH)	BH4	28, 35, 38
IDAADSEG	IDA019RS	Insert a Spanned-Record-Segment Entry into a Sequence-Set Record	BH1, BH2	24, 25
IDAADVPH	IDA021RV	Advance Placeholder Backwards	BD	21
IDAAIBF	IDA019RW	Add Insert Buffer to Chain	—	38
IDAAQR	IDA019RN	Split Index Record: Assign RBA to the Index Record	BG3, BG4, BG5, BG8, BH9	29, 30, 33, 34
IDACHKKR	IDA019RM	Check Key for Proper Key Range	—	—
IDACI96C	IDA0I96C	Free VSAM Checkpoint/Restart Storage	AJ, AL2	17, 18
IDACKRA1	IDACKRA1	Checkpoint/Restart: ESTAE	AJ, AK, AL1, AM	17, 18
IDADARTV	IDA019RT	Retrieve a Spanned Record	BC, BD	20, 21
IDADRQ	IDA019R5	Data Insert: Defer the Request until the Device is Available	BP1, BP3, BS2, DA1	24, 35, 38, 40
IDAENDRQ	IDA019RP	ENDREQ Request	BK1, BK2	31
IDAEOVIF	IDA019R5	Data Insert: Interface to VSAM End of Volume	BE, BG2, BN2, BO1, DA1, DA2, DA4	26, 28, 29, 34, 40
IDAER	IDA019RN	Index Create: Erase Dummy Entry from the Index Record	BG5	30
IDAERROR	IDA019R5	Determine Which Exit to Take	—	—
IDAEXCL	IDA019RZ	Exclusive Control	—	38
IDAEXEX	IDA019R5	Exit to User Exception Routine	—	38
IDAEXITR	IDA019R5	Exit to User Routine	BK1, BL	—
IDAFRBA	IDA019RW	Buffer Management: Determine next RBA for Sequential Processing	BN3, BS1, BS2, BS4	22, 38
IDAFREEB	IDA019RZ	Free a Buffer	BC, BD, BE, BG1, BG5, BH3, BH4, BH5, BM, BN1, BN2, BN3, BO1, BO2, BS1	20, 21, 22, 23, 24, 25, 27, 28, 30, 33, 34, 35, 38
IDAGETWS	IDA019RX	Get Working Storage	—	—
IDAGNFL	IDA019RZ	Buffer Management: Obtain an Empty Buffer	BG3, BH3, BH8	29, 30, 33, 34, 38
IDAGNNFL	IDA019RZ	Buffer Management: Obtain Next Empty Buffer for the Placeholder	BE, BF, BG1, BG4, BH4, BH5, BN2, BO1	24, 26, 27, 28, 35, 38
IDAGNXT	IDA019RZ	Buffer Management: Obtain Next Buffer in Sequence	BC, BD, BH4, BN1, BO1	20, 21, 23, 28, 35, 38
IDAGRB	IDA019RZ	Buffer Management: Obtain the Buffer That Contains the Specified RBA	BC, BE, BH1, BH3, BH4, BH5, BH9, BH10, BJ, BM, BN1, BO1, BS2, BS3	21, 22, 23, 27, 28, 33, 34, 38
IDAGWSEG	IDA019RZ	Get a Work Segment (for Shared Resources)	—	38
IDAGWSGW	IDA019RW	Get a Work Segment	—	38

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
IDAHLINS	IDA019RI	Insert Entry into Index-Set Record	BH4	28
IDAICIA1	IDAICIA1	ISAM Interface: Data-Set Management Recovery Routine	AG	11
IDAIIFBF	IDAIIFBF	ISAM Interface: FREEDBUF Processing	BU2	—
IDAIIPM1	IDAIIPM1	ISAM Interface: Load-Mode Processing	BU1	—
IDAIIPM2	IDAIIPM2	ISAM Interface: QISAM Scan-Mode Processing	BU1	—
IDAIIPM3	IDAIIPM3	ISAM Interface: BISAM Processing	BU2	—
IDAIISM1	IDAIISM1	ISAM Interface: SYNAD Processing	BU2	—
IDAIST	IDA019RG	Index Create: Insert Entry into Index Record	BG3, BG4, BG5, BH9	29, 30
IDAIVIXB	IDA019RH	Index Insert: Invalidate Buffers Containing a Copy of the Modified Index Record	— —	
IDAJRNSR	IDA019RT	Journal a Spanned-Record Segment	—	—
IDAMRKBF	IDA019RZ	Mark a Buffer	—	38
IDAMVSEG	IDA019RS	Move a Segment	BH1, BH2	24, 25
IDANEWRD	IDA019RI	Initialize a New Sequence-Set Record	BH4	28
IDAOCEA1	IDAOCEA1	Data-Set Management Recovery Routine (Force Close Executor)	AF, AG	8, 10, 11, 12, 16, 39
IDAOCEA2	IDAOCEA2	Task Close Executor	AH1, AH2, AH3	8, 39
IDAOCEA4	IDAOCEA4	BLDVRP/DLVRP ESTAE Routine	AF, AG	10, 16
IDAR	IDA019RJ	Split Index Record: Read the Record	BG5, BH9, BH10	30
IDARELWS	IDA019RX	Release Working Storage	—	—
IDAREPOS	IDA019RE	Reposition Placeholder	—	—
IDARRDRL	IDA019RR	Direct Record Locate for Relative Record	BJ, BO1	23, 35
IDARSTRT	IDA019R5	Restart	—	—
IDARVRS1	IDA019RV	Order Buffers	—	38
IDARXBD	IDA019RX	Increase Working Buffer Length	—	—
IDASBF	IDA019RZ	Buffer Management: Remove Buffers from Placeholder	BE, BH5, BK1, BN1, BN2	22, 23, 27, 28, 35, 38
IDASCHBF	IDA019RZ	Share a Buffer	—	—
IDASPACE	IDA019RH	Check an Index Record to Ensure It Can Be Split	—	—
IDASPNPT	IDA019RT	Make an INDEX Entry for a Spanned-Record Segment	—	—
IDATJXIT	IDA019RP	Control-Interval Request: Take the Journal Exit	BN1, BN3	20, 21, 24, 25, 26, 27, 35, 38
IDAWAIT	IDA019RZ	Buffer Management: Wait for Completion of I/O Operations	BS2, BS3, BS4	22, 38
IDAWR	IDA019RJ	Split Index Record: Write the Index Record	BG4, BG5, BH10	30, 31

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 Analysis	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	—	—
IDA019SF	IDA019SF	Control-Area Split-Spanned Records	BH4	28
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,BH3	21,22
IDA0192A	IDA0192A	VSAM Open String	AC1, AC2, AC7, AG	5, 6, 7, 11
IDA0192B	IDA0192B	Open a Cluster	AC4, AL1	8, 18
IDA0192C	IDA0192C	VSAM Open/Close: Catalog Interface	AC2, AC4, AD6, AD7, AE2, AL1, BT1	5, 6, 8, 12, 14, 15, 18, 39
IDA0192D	IDA0192D	Stage/Destage (ACQUIRE/RELINQUISH)	AD6, AE2, BT28,	12, 14, 18, 39
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

External Procedure Directory

Procedure Name	Module Name	Descriptive Name	Method of Operation Diagrams	Program Organization Figures
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, DA440	
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, DA340	
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, IDA019S6
IDA019L2	Improved Control-Interval Processing	IDA019S1, IDA019S3
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, IDA0196C, 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
I/O Management		
IDAM19R3	Problem-State I/O Driver	IDAM19R3 (Packaged in Main Record Management IDA019L1)
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)

RDF—Record Definition Field

The record definition field (RDF) describes a record, record slot, or record segment within the control interval. RDFs are put into the control interval right to left so that the rightmost RDF describes the leftmost data record. The format of the RDF is:

Offset	Bytes and Bit Pattern	Description
0(0)	1	Control field:
	.x..	Indicates whether there is (1) or isn't (0) a paired RDF to the left of this RDF.
	..xx	Indicates whether the record spans control intervals: 00 No. 01 Yes—this is the first segment. 10 Yes—this is the last segment. 11 Yes—this is an intermediate segment.
	.... x...	Indicates what the 2-byte binary number that follows this control field gives: 0 The length of the record, segment, or slot described by this RDF. 1 The number of consecutive unspanned records of the same length, or the update number of the segment of a spanned record.
	.... .x..	For a relative record data set, indicates whether the slot described by this RDF does (0) or doesn't (1) contain a record.
	x... ..xx	Reserved.
1(1)	2	Binary number: • When bit 4 in the control field is 0, gives the length of the record, segment, or slot described by this RDF. • When bit 4 in the control field is 1 and bits 2 and 3 are 0, gives the number of consecutive records of the same length. • When bit 4 in the control field is 1 and bits 2 and 3 aren't 0, gives the update number of the segment described by this RDF.

CIDF—Control Interval Definition Field

In an entry-sequenced data set, when there are unused control intervals beyond the last one that contains data, the first of the unused control intervals contains a CIDF filled with 0s. In a key-sequenced or relative record data set, or a key-range portion of a key-sequenced data set, the first control interval in the first unused control area (if any) contains a CIDF filled with 0s. A control interval with such a CIDF contains no data or unused space, and is referred to as the "Software End of File (SEOF)."

The control interval definition field (CIDF) describes the control interval. The format of the CIDF is:

Offset	Bytes and Bit Pattern	Field Name	Description
0(0)	2	CIDFOSET	The displacement from the beginning of the control interval to the beginning of the unused space, or, if there is no unused space, to the beginning of the control information. This number is equal to the length of the data (records, record slots, or record segment). In a control interval without data, the number is 0.
2(2)	2	CIDF11	The length of the unused space. This number is equal to the length of the control interval minus the length of the control information, minus the 2-byte number in the preceding field. In a control interval without data (records, record slots, or record segment), the number is the length of the control interval, minus 4 (the length of the CIDF—there are no RDFs). In a control interval without unused space, the number is 0.
2(2)	1...	CIDFBUSY	Set on for CI split interruption and set off when interruption complete.

Control Area

A control area consists of control intervals; the number of control intervals in a control area is determined by VSAM. The control area is the amount of space that VSAM preformats so that data integrity is ensured for records added to a data set.

Control areas are also used to simplify and localize the movement of records when records are inserted in a key-sequenced data set. If an insertion requires a free control interval and there isn't one, a control-area split results. VSAM establishes a new control area and moves the contents of approximately half of the full control area to free control intervals in the new control area. The new records, as their keys dictate, are then inserted into one of the two control areas.

Index Format

There are two types of indexes in VSAM: the *prime index* of a key-sequenced data set, and *alternate indexes* of either a key-sequenced or an entry-sequenced data set.

A key-sequenced data set is a cluster composed of a data component, which contains the control intervals that contain data records, and an index component, which contains the control intervals that contain the records of the prime index.

An alternate index is itself a key-sequenced data set. Its data component contains index records that give the location of data records within its *base cluster* (the key-sequenced or entry-sequenced data set for which it is the alternate index).

Generalized Trace Facility

The Generalized Trace Facility (GTF) can be used to record information about VSAM processing at the time of an error. If GTF is active in the VS2 system, GTF is used to trace VSAM control blocks when there is an error.

GTF is used to record the contents of the ACB, AMBL, AMBs, AMDSBs, and TIOT entry for the data set being processed when the error occurred.

To format and print GTF records, use the IMDPRDMP service aid with 'USR=(FFF,FF5)' specified in the EDIT statement.

Two types of traces are available to help in debugging VSAM Open/Close/End-of-Volume problems:

- The error trace routine traces VSAM control blocks when an error is detected. The optional work area trace traces the Open/Close/End-of-Volume work area WTG table prefix and the current entry in the WTG table at entry to and exit from the VSAM Open/Close/End-of-Volume modules.
- The work area trace is requested by specifying AMP='TRACE'. This is the same trace that is obtained for nonVSAM Open/Close/End-of-Volume processing when DCB=DIAGNS=TRACE is specified in the JCL. (For details on the AMP and DCB JCL parameters and options, see *OS/VS2 JCL*.

Both traces require that GTF be operating in external mode while the job to be traced is running. In addition, the operator must respond with "TRACE=USR" when the GTF trace message"SPECIFY TRACE OPTIONS" appears at the operator's console.

Additional information on GTF and IMDPRDMP is contained in the *OS/VS2 System Programming Library: Service Aids*.

Return Codes

VSAM sets return codes in the RPL and the ACB. These codes are paired with codes in register 15. Codes set in the RPL are listed and explained under "Return Codes from the Record-Management (Request) Macros." Those set in the ACB, which indicate open or close errors, are listed and explained under "Open and Close Return Codes."

VSAM sets a pair of codes in registers 15 and 0 for the control block manipulation macros. These are listed and explained under "Control Block Manipulation Return Codes."

Return Codes from the Record-Management (Request) Macros

After a request macro or a CHECK or ENDREQ macro is issued, register 15 contains a return code.

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)	Duplicate alternate key follows.
12(C)	(Shared resources only.) A buffer needs to be written.
16(10)	Control area split was required because an index record was too small.
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.)

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.

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

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.

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
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
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
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 during processing with SHROPT=4 and DISP=SHR. Detected by: IDA019R5
32(20)	An RBA was specified that doesn't give the address of any data record in the data set. Detected by: IDA019RA, IDA019R8
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
44(2C)	Work area not large enough for the data record (GET with OPTCD=MVE). Detected by: IDA019RR, IDA019RT, IDA019RY, IDA019R4, IDA019R8
64(40)	As many requests are active as the number specified in the STRNO parameter 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

Figure 58 (Part 1 of 3). Logical-Error Return Codes in the RPL Feedback Field from a Request Macro

RPLFDBK

Code	Condition
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
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

Figure 58 (Part 2 of 3). Logical-Error Return Codes in the RPL Feedback Field from a Request Macro

RPLFDBK Code	Condition
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. Detected by: IDA019RA, IDA019RV, IDA019R4, IDA019S6
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

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

D

E

F

G

H

I