

Burroughs

*Reference
Manual*

**B 20
Systems
Indexed Sequential
Access Method
(ISAM)**

Distribution Code SA

*Priced Item
Printed in U.S.A
July 1985*

5022247

*Reference
Manual*

**B 20
Systems
Indexed Sequential
Access Method
(ISAM)**

(Relative to Release Level 5.0)
Copyright © 1985, Burroughs Corporation, Detroit, Michigan 48232

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the CLASS specified as 2 (S.SW: System Software), the Type specified as 1 (F.T.R.), and the product specified as the 7-digit form number of the manual (for example, 5022247).

LIST OF EFFECTIVE PAGES

Page	Issue
Title	Original
ii	Original
iii	Original
iv	Blank
v thru ix	Original
x	Blank
xi thru xii	Original
1-1 thru 1-5	Original
1-6	Blank
2-1 thru 2-13	Original
2-14	Blank
3-1 thru 3-6	Original
4-1 thru 4-14	Original
5-1 thru 5-13	Original
5-14	Blank
6-1 thru 6-11	Original
6-12	Blank
7-1 thru 7-48	Original
A-1 thru A-13	Original
A-14	Blank
B-1 thru B-3	Original
B-4	Blank
C-1 thru C-3	Original
C-4	Blank
D-1 thru D-4	Original
1 thru 9	Original
10	Blank

TABLE OF CONTENTS

Section	Title	Page
	INTRODUCTION.....	xi
	Examples in This Manual.....	xii
	Reference Material.....	xii
1	OVERVIEW.....	1-1
	Features.....	1-1
	Memory and Disk Requirements.....	1-2
	Related Software.....	1-3
	Installation Instructions.....	1-3
	Hard Disk Systems.....	1-3
	Dual Floppy Standalone Systems.....	1-3
	XE520 Systems.....	1-4
2	CONCEPTS.....	2-1
	File Types.....	2-1
	Key Types.....	2-2
	Unique Record Identifier.....	2-3
	ISAM Operations.....	2-3
	Storing Records.....	2-3
	Reading Records.....	2-4
	Modifying Records.....	2-4
	Deleting Records.....	2-4
	Distributed ISAM.....	2-4
	Data Set Access Modes.....	2-5
	Transactions.....	2-6
	Locking a Record or Data Set.....	2-7
	Deadlocks.....	2-10
	ISAM Installation.....	2-10
	ISAM Configuration File.....	2-12
	ISAM Commands.....	2-12
	Data Security.....	2-12
	Data Integrity.....	2-13
3	ISAM DATA SETS.....	3-1
	Indexes and Keys.....	3-1
	Key Types.....	3-3
4	ISAM UTILITIES.....	4-1
	Default Index File Name.....	4-2
	ISAM COPY.....	4-2
	ISAM CREATE.....	4-4
	ISAM DELETE.....	4-8
	ISAM RENAME.....	4-8
	ISAM SET PROTECTION.....	4-10
	ISAM STATUS.....	4-12
	ISAM TERMINATE.....	4-14

TABLE OF CONTENTS (Cont)

Section	Title	Page
5	ISAM REORGANIZATION.....	5-1
	Loading a Data Set.....	5-7
	Changing Indexes and Other ISAM CREATE Parameters.....	5-8
	Recovering Records, Reclaiming Space, and Merging Data.....	5-10
	Sorting Data Set Records.....	5-11
6	ISAM SERVER INSTALLATION.....	6-1
	Multiuser Installation.....	6-2
	Single-Partition BTOS.....	6-2
	Multipartition BTOS.....	6-3
	ISAM INSTALL.....	6-3
	Memory Allocation.....	6-3
	Resident Code and Data.....	6-4
	Swap Zone.....	6-5
	Heap.....	6-5
	Data Buffers.....	6-5
	Index Buffers.....	6-6
	ISAM CONFIGURE.....	6-6
	ISAM Configure Display.....	6-7
	Cursor Movement.....	6-7
	Display.....	6-8
	Memory Allocation Calculation.....	6-10
	Buffer Size Guidelines.....	6-11
7	ISAM OPERATIONS.....	7-1
	Status Block.....	7-1
	Data Set Management.....	7-3
	ISAM Description Block.....	7-3
	Data Set Access.....	7-8
	Record Management and Access.....	7-8
	Locking.....	7-10
	Transactions.....	7-10
	Transaction-Related Constraints.....	7-11
	Transaction Parameters Block.....	7-11
	ISAM Service Access.....	7-12
	Memory Usage.....	7-13
	Using Either Multiuser or Single-User ISAM.....	7-13
	Asynchronous Requests.....	7-14
	Procedure Definitions.....	7-14
	BeginTransaction Procedure.....	7-14
	CloseISAM Procedure.....	7-15
	CommitTransaction Procedure.....	7-15

TABLE OF CONTENTS (Cont)

Section	Title	Page
	CreateISAM Procedure.....	7-16
	DeleteISAM Procedure.....	7-18
	DeleteISAMRecord Procedure.....	7-18
	DeleteISAMRecordByKey Procedure.....	7-19
	GetISAMRecords and GetISAMRecordsHold Procedures.....	7-20
	HoldISAMDataSet Procedure.....	7-24
	HoldISAMRecord Procedure.....	7-26
	ISAMRequest Procedure.....	7-27
	LoadSingleUserISAM Procedure.....	7-27
	ModifyISAMRecord Procedure.....	7-28
	ModifyISAMRecordByKey Procedure.....	7-29
	NormalizeISAMStatus Procedure.....	7-31
	OpenISAM Procedure.....	7-32
	QueryTransactionParams Procedure....	7-33
	ReadISAMRecordByUri and ReadIsam- RecordByUriHold Procedures.....	7-33
	ReadNextISAMRecord and ReadNext- ISAMRecordHold Procedures.....	7-35
	ReadUniqueISAMRecord and Read- UniqueISAMRecordHold Procedures...	7-35
	ReleaseISAMDataSet Procedure.....	7-37
	ReleaseISAMRecord Procedure.....	7-38
	RenameISAM Procedure.....	7-39
	RollBackTransaction Procedure.....	7-41
	SetISAMProtection Procedure.....	7-42
	SetTransactionParams Procedure.....	7-42
	SetUpISAMIterationLimits Procedure..	7-43
	SetUpISAMIterationPrefix Procedure..	7-45
	StoreISAMRecord Procedure.....	7-47
	VerifyMultiUserISAM Procedure.....	7-48
A	STATUS CODES.....	A-1
B	UPWARD COMPATIBILITY SUPPORT.....	B-1
C	ESTIMATING INDEX FILE SIZES.....	C-1
D	GLOSSARY.....	D-1
	INDEX.....	1

LIST OF ILLUSTRATIONS

Figure	Title	Page
2-1	Series of Transactions.....	2-8
2-2	Deadlock (Samples).....	2-11
3-1	Data Set Index (Sample).....	3-2
4-1	ISAM COPY Form.....	4-2
4-2	ISAM CREATE Form.....	4-4
4-3	ISAM RENAME Form.....	4-9
4-4	ISAM SET PROTECTION Form.....	4-11
4-5	ISAM STATUS Form.....	4-13
4-6	ISAM STATUS Reports (Samples).....	4-14
5-1	ISAM REORGANIZE Form.....	5-2
6-1	ISAM INSTALL Form.....	6-3
6-2	ISAM CONFIGURE Form.....	6-7
6-3	ISAM CONFIGURE Display (Sample).....	6-9

LIST OF TABLES

Table	Title	Page
2-1	Description of Data Set Access Modes.	2-5
2-2	Multiuser Access to Data Set Access Modes.....	2-7
2-3	Operations and Transactions.....	2-9
3-1	Description of Key Types.....	3-3
3-2	ISAM Key Types and Programming Language Representations.....	3-5
4-1	ISAM Commands.....	4-1
4-2	ISAM COPY Form Parameters.....	4-3
4-3	ISAM CREATE Form Parameters.....	4-5
4-4	ISAM RENAME Form Parameters.....	4-10
4-5	ISAM SET PROTECTION Form Parameters..	4-11
4-6	ISAM STATUS Form Parameters.....	4-13
5-1	ISAM REORGANIZE Form Parameters.....	5-3
6-1	Differences Between Multiuser and Single-User Access.....	6-2
6-2	ISAM INSTALL Form Parameters.....	6-4
7-1	ISAM Operations by Function.....	7-2
7-2	Status Block Format (pStatusBlockRet Parameter).....	7-2
7-3	Data Set Management Operations.....	7-3
7-4	ISAM Description Block.....	7-4
7-5	ISAM Index Specification Block.....	7-5
7-6	Type of Key Component.....	7-7

LIST OF TABLES (Cont)

Table	Title	Page
7-7	Record Management Operations.....	7-8
7-8	Single Record Access Operations.....	7-9
7-9	Multiple Record Access (Iteration) Operations.....	7-9
7-10	Locking Operations.....	7-10
7-11	Transaction Operations.....	7-10
7-12	Transaction-Related Constraints.....	7-11
7-13	Transaction Parameters Block Format..	7-12
7-14	ISAM Service Access.....	7-13
7-15	Asynchronous Requests.....	7-14
7-16	CreateISAM Request Block.....	7-17
7-17	DeleteISAM Request Block.....	7-18
7-18	DeleteISAMRecord Request Block.....	7-19
7-19	DeleteISAMRecordByKey Request Block..	7-21
7-20	GetISAMRecords and GetISAMRecords- Hold Request Block.....	7-23
7-21	Buffer Structure for GetISAMRecords and GetISAMRecordsHold when Records are Read (46-Byte Records).....	7-24
7-22	HoldISAMDataSet Request Block.....	7-25
7-23	HoldISAMRecord Request Block.....	7-26
7-24	ModifyISAMRecord Request Block.....	7-29
7-25	ModifyISAMRecordByKey Request Block..	7-31
7-26	Data Set Modes.....	7-33
7-27	ReadISAMRecordByUri and ReadISAM- RecordByUriHold Request Block.....	7-34
7-28	ReadNextISAMRecord and ReadNextISAM- RecordHold Request Block.....	7-36
7-29	ReadUniqueISAMRecord and ReadUnique- ISAMRecordHold Request Block.....	7-37
7-30	ReleaseISAMDataSet Request Block.....	7-38
7-31	ReleaseISAMRecord Request Block.....	7-39
7-32	RenameISAM Request Block.....	7-41
7-33	SetISAMProtection Request Block.....	7-43
7-34	SetUpISAMIterationLimits Request Block.....	7-45
7-35	SetUpISAMIterationPrefix Request Block.....	7-46
7-36	StoreISAMRecord Request Block.....	7-48

INTRODUCTION

This manual provides descriptive and operational information for the B 20 Indexed Sequential Access Method (ISAM) data management facility used by the B 20 family of workstations. ISAM provides efficient and flexible random access to data identified by multiple keys. System designers and applications programmers should use this manual to write ISAM applications used under BTOS.

This manual consists of seven sections, four appendixes and an index:

- Section 1 Provides you with an overview of ISAM, including its features and installation instructions.
- Section 2 Introduces basic ISAM concepts which are explained in more detail in later sections.
- Section 3 Describes ISAM data sets and key types.
- Section 4 Describes the ISAM commands: **ISAM COPY**, **ISAM CREATE**, **ISAM DELETE**, **ISAM RENAME**, **ISAM SET PROTECTION**, **ISAM STATUS**, and **ISAM TERMINATE**.
- Section 5 Documents the **ISAM REORGANIZE** command.
- Section 6 Explains how to install and configure the ISAM server.
- Section 7 Describes ISAM operations by functional categories and presents all the operations in detail.
- Appendix A Contains the status codes and messages you could receive while using ISAM.
- Appendix B Describes the compatibility of earlier applications with this version of ISAM.
- Appendix C Explains how to estimate index file sizes.
- Appendix D Contains the glossary.

EXAMPLES IN THIS MANUAL

This manual uses personnel data sets for examples and illustrations because they are familiar and provide continuity as the manual presents the various ISAM functional levels.

Three data sets contain personnel information: an employee data set, a department data set, and a dependent data set. The parameters included in each data set are:

Employee data set, with one record per employee

- department number
- employee number
- employee name
- salary

Department data set, with one record per department

- department number
- department name
- employee number of department manager

Dependent data set, with one record per dependent

- employee number
- number of dependents
- dependent name
- dependent date of birth

REFERENCE MATERIAL

The following manuals are referenced for additional information:

B 20 Systems Operating System (BTOS) Reference Manual
B 20 Systems Programmer's Guide, Part 1
B 20 Systems Linker/Librarian Reference Manual
B 20 Systems Sort/Merge Reference Manual

SECTION 1

OVERVIEW

The B 20 Indexed Sequential Access Method (ISAM) is a software product that provides efficient, flexible random access to fixed-length records. These fixed-length records are identified by keys contained in the records.

You can use 8086 Assembly, BASIC, COBOL, FORTRAN, and Pascal to write applications accessing ISAM. You can use ISAM on standalone, cluster, and master workstations.

This section contains:

- an overview of ISAM features
- software installation procedures
- memory and disk requirements
- files necessary to run ISAM.

FEATURES

New ISAM features include:

- the ability of applications running on a cluster workstation to simultaneously access data sets located on the local and master workstations
- application or Executive access to remote ISAM data sets across a network (using B-NET)
- the ability to run one or more ISAM applications on each workstation (using the B 20 Context Manager)
- resident or swapping ISAM reorganization
- a utility to remove the ISAM server from a secondary partition of a multipartition system

Standard ISAM features include:

- random access to data identified by multiple keys

- up to 100 keys in each ISAM data set
- key types to support numerous character and numeric data representations
- shared or exclusive access to ISAM data sets, with transactions and record-level or data set-level locking
- resident or swapping ISAM server
- ISAM data sets created from any fixed-length record Sequential Access Method file using ISAMReorganize
- commands to perform maintenance and modifications to data sets and to provide status reports for data sets

MEMORY AND DISK REQUIREMENTS

ISAM requires 57KB of memory for the swapping version of the server and 90KB for the resident version. The minimum installation of ISAM software on hard disk requires 776 disk sectors. The full installation, including optional utilities, requires 1561 disk sectors.

The following files constitute the minimum configuration of ISAM:

```
ISAM.Config
ISAMConfigure.Run
ISAMCreate.Run
ISAMDelete.Run
ISAMRename.Run
ISAMSetPro.Run
ISAMServer.Run
```

The following files comprise the optional ISAM utilities:

```
ISAMCopy.Run
ISAMReorganize.Run
ISAMStatus.Run
ISAMTerminate.Run
```

RELATED SOFTWARE

ISAM requires Release 5.0 of BTOS. For networking applications, ISAM requires B-NET, Release 1.0. Section 6 discusses configuration requirements.

INSTALLATION INSTRUCTIONS

You install the ISAM software from the distribution diskettes.

Hard Disk Systems

To install ISAM on a hard disk system, use the following procedure:

1. Sign on to a B 20 workstation and set your path to the system directory.
2. Insert the first ISAM distribution diskette in floppy drive [f0].
3. Type **SOFTWARE INSTALLATION** at the Executive prompt; then press **GO**.
4. Follow all instructions shown on the display, and insert each of the other ISAM distribution diskettes when prompted.
5. When installation is complete, remove the last diskette and store all of them in a safe place.

Dual Floppy Standalone Systems

To use ISAM on a dual floppy standalone system, prepare working copies (write-enabled) of the BTOS and ISAM diskettes. (For further information on making working copies, refer to the B 20 Systems Standard Software Operations Guide.) Then proceed as follows:

1. Boot your system using a working copy of the BTOS boot diskette.
2. Set your path to the system directory.

3. Insert a working copy of the second BTOS diskette in floppy drive [f0].
4. Insert the first ISAM diskette in floppy drive [f1].
5. Type **SUBMIT** at the Executive prompt, and press **RETURN**.
6. Specify [f1]<Create>**ISAMFloppy.Sub** in the SUBMIT form File list parameter.
7. Press **GO**.
8. Follow the instructions shown on the display.
9. When installation is complete, remove the distribution diskette and write-protect the ISAM diskettes.

XE520 Systems

To install ISAM on an XE520 system, use the following procedure:

1. Sign on to a B 20 workstation clustered to an XE520 master, using the Administrator user name.
2. Insert the first ISAM distribution diskette in floppy drive [f0].
3. Type **SUBMIT** at the Executive prompt and press **RETURN**.
4. Specify [f0]<Sys>**XEInstall.Sub** in the SUBMIT form File list parameter.
5. Press **GO**.
6. Follow the instructions shown on the display.
7. When the installation is complete, remove the last diskette and store all of them in a safe place.

The following information applies to the installation of ISAM on an XE520 system.

- You should add the following statement to the INITFP00.JCL file:
\$Run [Sys]<Sys>ISAMServer.Run,<number of users>,
<configuration file>

This statement must follow any entry for the queue manager and precede any entry for MfAdminAgent.

- Only one copy of the ISAM server can run on an XE520 system. You can install it on a board other than the master FP by inserting the statement \$Run[Sys]<Sys>ISAMServer.Run,<number of users>,<configuration file> into the appropriate JCL file. It must precede any entry for a spooler.
- You should make the statement NoWatchDog the first line in the Master.Cnf file.
- The memory required (in kilobytes) is $64 + 4(\text{number of users} - 4)$.

SECTION 2

CONCEPTS

ISAM allows you to access fixed-length data records contained in ISAM data sets. Each ISAM data set holds one type of data record. When you create an ISAM data set, you specify the record length and key fields, then access the records of a data set through fixed-length keys. You can define multiple key fields to support different access patterns.

FILES TYPES

ISAM stores each ISAM data set as two physical files: a data store file and an index file. You can place these two files on different physical volumes, provided they are on the same workstation.

The data store file holds data records. Disk space management is efficient because all the records in a data set have the same length. Whenever you delete a record, the system marks it deleted, and adds it to a list of free records. The system can reuse these deleted records later to create a new record. The data store file is a Direct Access Method (DAM) file.

The name of a data set is the same as the name of its data store file. You must supply a file specification for the data store file to access or create a data set.

The index file holds indexes for all the keys of a data set. ISAM implements indexes by using a B-tree structure. The B-tree structure has several advantages:

- support of both direct (by key) and sequential access
- efficiency:
 - always takes the same number of I/O operations to reach a record
 - never has to follow long overflow chains

- uses two or three reads to locate a record and one to actually read the record
- fast sequential access: maximum of three reads to locate and read the first record--usually a single read for each subsequent record
- self-reorganization: automatic expansion to accommodate new keys
- automatic compression: removal of keys to improve disk utilization and access efficiency

ISAM separates data and indexes to allow the use of Record Sequential Access Method (RSAM) and DAM for read-only access to ISAM data set files. (For more information on RSAM, refer to the B 20 Systems Operating System (BTOS) Reference Manual.)

ISAM maintains logical dependencies between the data store file and index file of a data set; no access method other than ISAM can open or modify either file and a consistency check detects any invalid access.

The separate data store and index files enable you to use the **MAINTAIN FILE** command to recover data from the data store file after hardware or software failure.

Key Types

A record can have up to 100 keys. The key position in the record describes the key. For example, the key position can show:

- offset from the first byte of the record
- key length
- key type

The ISAM key types support character and numeric representations used by the B 20 programming languages and processors. Byte string and character string key types support character data. Numeric key types support integer, binary, packed decimal, display, and several forms of real numbers.

To increase flexibility, you can specify the following parameters for each key when you create an ISAM data set:

- duplicate keys
- index order (ascending or descending)
- index size reduction (suppress null value key indexing)

Each key uses one index for retrieving a record. You can retrieve records in key-order sequence by any key field and starting with any key value. The system automatically updates the index when you store or modify records.

UNIQUE RECORD IDENTIFIER

A 4-byte unsigned integer uniquely identifies each record in a data set and is called the unique record identifier (URI). Store and Read operations return the record URI. Modify and Delete operations use URIs to identify the target records processed.

URIs are valid only while the data set is open. You cannot save them to identify records once you close and reopen the data set.

ISAM OPERATIONS

ISAM supports four main operations on records:

- storing
- reading
- modifying
- deleting

Storing Records

When an application stores a new record, ISAM places the record in the data set. It then automatically indexes the record according to the values in all the key fields.

Reading Records

When an application reads a record, ISAM retrieves any of the following:

- a single record with a given unique key or unique record identifier
- records with keys of a specific value (an exact match)
- records with key values residing in a specified range (a range match)
- records in which the beginning of a byte or character string key matches a particular value (a prefix match)

The retrieval result can be either the specified records or a sequence of 4-byte unique record identifiers. If ISAM retrieves unique record identifiers, the application can obtain the corresponding records later by using a special form of the Read operation that does not reaccess the index.

Modifying Records

When an application modifies an existing record, ISAM automatically removes the record from each index for a changed key field. ISAM then indexes it under the new key.

Deleting Records

When an application deletes an existing record, ISAM removes the record from the data set and from each individual index.

DISTRIBUTED ISAM

You can share ISAM files with users working at other workstations.

For example, within a cluster system all cluster workstations can access ISAM files located at the master. In addition, they can maintain local files at the cluster workstations if they have local file storage. You can

access ISAM files at the master and cluster workstations simultaneously. In this case, an application system could read a record from a data store file at the master and write a record to a data store file at the cluster. You must install the ISAM server at both workstations.

In addition, you can use B-NET to access ISAM files located on other network workstations. You must specify the node name as part of the complete file specification.

ISAM does not provide transactional monitoring.

DATA SET ACCESS MODES

You can use one of three modes to open a data set in ISAM:

- administrator
- batch
- transaction

Table 2-1 describes data set access modes.

The use of read or modify affects the extent to which applications share a data set.

You can share a data set opened in batch read mode if other users open it for read-only purposes. If you open a data set in batch read mode, the system denies any subsequent request to open the data set in administrator, batch modify, or transaction modify mode.

Table 2-1. Description of Data Set Access Modes

Mode	Activity
Administrator	data set-level activities (deleting ISAM files, renaming ISAM files, and setting protection on the file)
Batch	open a data set in applications requiring exclusive use of the data set (use with read or modify)
Transaction	allows other applications to concurrently access and modify the data set

You open a data set in batch modify mode exclusively; no other user has access to it. A request to open the data set in batch modify mode cannot execute if another user opened it in any mode.

Using batch read, transaction read, or transaction modify modes, other users can open a data set you opened in transaction read mode. A request to open the data set in transaction read mode cannot execute if another user has the data set open in administrator or batch modify mode.

Other users in either transaction read or transaction modify modes can open a data set you opened in transaction modify mode. You cannot execute a request to open the data set in transaction modify mode if another user has the data set open in administrator, batch read, or batch modify mode.

A data set you open in administrator mode is open exclusively. A request to open the data set in administrator mode cannot be executed if another user has already opened it in any mode.

Table 2-2 summarizes these rules.

TRANSACTIONS

An application can open a data set in batch mode to read or modify records. In batch mode, an application has exclusive access to the data set. Since other applications do not have access to the data set at the same time, you cannot make simultaneous updates. Modification of records and data sets does not affect any other application.

If you open a data set in transaction mode for shared access, many applications can access the same records and data sets. Any modification of a record or data set by one application affects the other applications. Errors can occur if you do not coordinate simultaneous access. In ISAM, transactions are the coordination mechanism.

Table 2-2. Multiuser Access to Data Set Access Modes

Initial Mode	Valid Modes for New Open
Administrator	None
Batch read	Batch read Transaction read
Batch modify	None
Transaction read	Batch read Transaction read Transaction modify
Transaction modify	Transaction read Transaction modify

In transaction mode, applications designed to allow multiuser access to a data set divide their processing into a series of transactions. Each transaction is a unit of work, as shown in figure 2-1. You must make changes to a data set within a transaction. Only after a completed transaction can other applications access the changed data.

The beginning of a transaction is the BeginTransaction operation; the end of a transaction is either a CommitTransaction or a RollBackTransaction operation.

You can perform some operations whether or not the application is in a transaction, but you must perform any operation that locks or modifies a record (or locks a data set) while the application system is in a transaction. Table 2-3 illustrates this relationship.

Locking a Record or Data Set

During a transaction, an application can make certain changes to a data set. When an application system is in the midst of modifying data, the data set is not in a consistent state; ISAM prevents other applications from accessing the changed data.

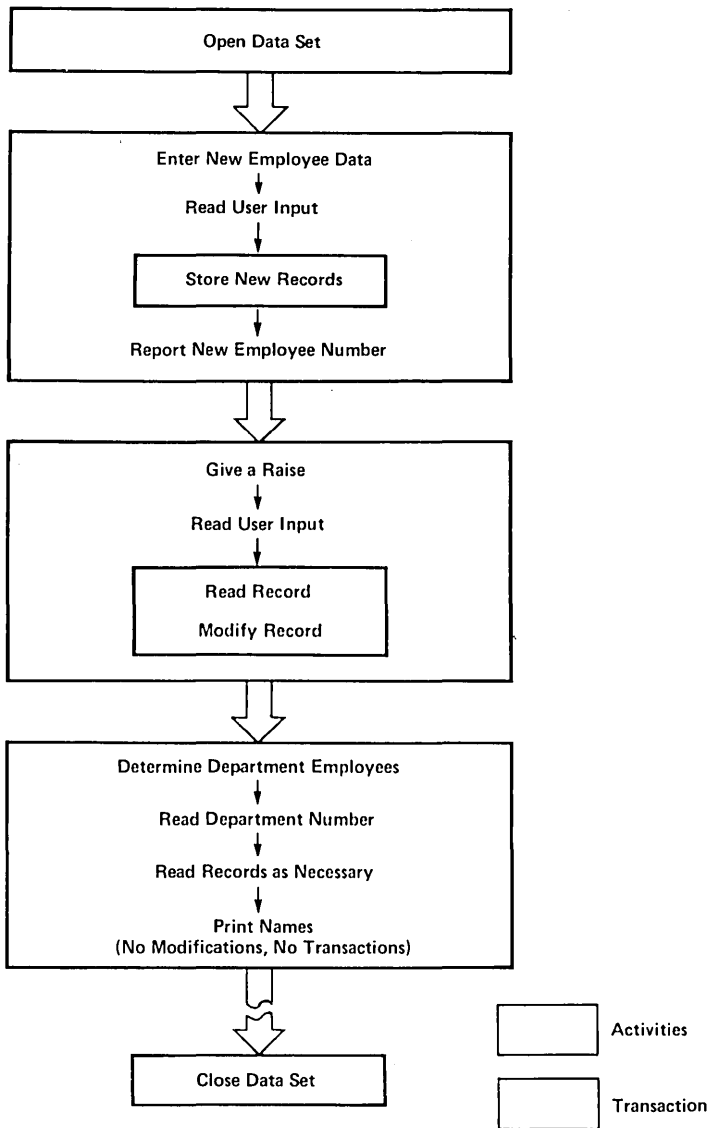


Figure 2-1. Series of Transactions

Table 2-3. Operations and Transactions

Operations Allowed Only During a Transaction	Operations Allowed At Any Time
CommitTransaction	Close ISAM
DeleteISAMRecord	GetISAMRecords
DeleteISAMRecordByKey	ISAMRequest
GetISAMRecordsHold	NormalizeISAMStatus
HoldISAMDataSet	OpenISAM
HoldISAMRecord	QueryTransactionParams
ModifyISAMRecord	ReadISAMRecordByUri
ModifyISAMRecordByKey	ReadNextISAMRecord
ReadISAMRecordByUriHold	ReadUniqueISAMRecord
ReadNextISAMRecordHold	RollBackTransaction
ReadUniqueISAMRecordHold	SetTransactionParams
ReleaseISAMDataSet	SetUpISAMIterationLimits
ReleaseISAMRecord	SetUpISAMIterationPrefix
StoreISAMRecord	

To prevent concurrent modification by multiple applications, ISAM allows an application to lock a record or data set. This ability gives the application exclusive access to the record or data set.

Table 2-3 illustrates ISAM operations allowed during transactions. There are no transaction-related constraints for the following operations: CreateISAM, DeleteISAM, LoadSingleUserISAM, RenameISAM, SetISAMProtection, and VerifyMultiuserISAM. ISAM does not allow BeginTransaction during a transaction.

If an application locks a record, only that application has access to the record until ISAM releases or unlocks it. If an application locks a data set, only that application has access to the data set and all of its records until it is unlocked.

Locking a record allows other application systems to access the remaining records of the data set. You should lock a data set only when necessary; locking prevents other applications from accessing any of the data set's records.

Deadlocks

When more than one application attempts to lock the same record or data set, the first application to request the record or data set obtains it; other application requests queue up until the record or data set unlocks.

A deadlock can occur when an application that locked a record or data set attempts to lock another record or data set that is locked by a different application. If the second application awaits the record or data set locked by the first application, both applications wait for the locked record or data set and can remain this way indefinitely. This deadlock becomes complicated when a number of applications and requests are involved, as shown in figure 2-2.

To avoid a deadlock, ISAM uses timeouts when an application requests a locked record or data set. You specify the maximum time a request can queue in the timeout value of the Transaction Parameters Block (refer to section 7). The timeout value, `wTicksWait`, specifies the maximum time a request waits to lock a record or data set. If the time specified in `wTicksWait` runs out, ISAM reports that the record or data set is not available.

ISAM INSTALLATION

You can use ISAM on a standalone workstation or in cluster configurations; it supports both single and multiuser access. An application loads single-user ISAM as a task, and you install multiuser ISAM as a system service. You can install multiuser ISAM in either a single-partition or multipartition operating system.

In a single-partition operating system, you install ISAM permanently in memory. Once you install ISAM, you cannot remove the ISAM server, nor can you reallocate its memory without bootstrapping BTOS.

In a multipartition BTOS, you install ISAM in a secondary application partition. (For further information about secondary application partitions, refer to the B 20 Systems Operating System (BTOS) Reference Manual.)

_____ Lock
 - - - - - Request to Lock

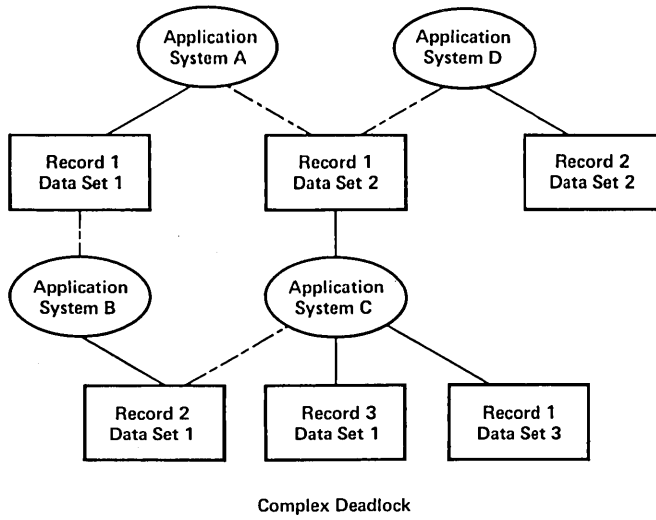
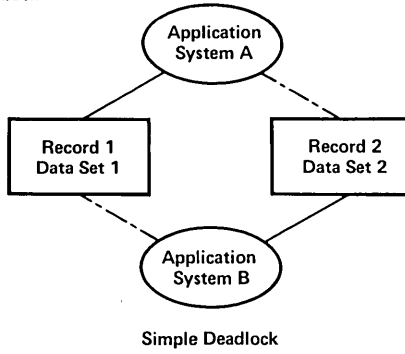


Figure 2-2. Deadlock (Samples)

ISAM Configuration File

An ISAM configuration file specifies the sizes of the ISAM server's memory areas, based on the number of users. ISAM provides default values for this file. Additionally, there is a utility that modifies the file to improve ISAM's performance or reduce memory requirements.

ISAM COMMANDS

ISAM provides commands that you can use from the Executive to maintain and modify ISAM data sets:

- ISAM COPY
- ISAM CREATE
- ISAM DELETE
- ISAM RENAME
- ISAM REORGANIZE
- ISAM SET PROTECTION
- ISAM STATUS
- ISAM TERMINATE

Table 4-1 describes these commands.

DATA SECURITY

Two associated passwords provide data security at each ISAM data set level: read and modify. In addition, BTOS protects the files of an ISAM data set from unauthorized access.

DATA INTEGRITY

ISAM includes features for maintaining and monitoring the integrity of data on disk files through:

- **Error logging**

The system logs errors discovered in the ISAM file structures in the file [Sys]<Sys>Log.Sys. You can use the **PLOG** command to display the log contents.

- **Internal consistency checking**

Hardware or software errors can introduce anomalies into a data set. ISAM algorithms minimize these anomalies by detecting them where possible and preventing them from becoming worse.

- **Write-through cache**

ISAM maintains and uses a set of I/O buffers that bring segments of disk records into memory as needed. Whenever ISAM stores or modifies a record, ISAM writes the changed data in the buffers back to disk. ISAM always updates the disk before completing an operation that changes a data set.

This policy makes the data less susceptible to damage from hardware or software failures, unless the failure occurs in the middle of a Modify, Store, or Delete operation. ISAM updates all files; no partial modifications remain in memory without updating the disk.

If hardware or software failures damage an ISAM data set, you can recover undamaged records by using the **MAINTAIN FILE** command, then you can use the **ISAM REORGANIZE** command to reconstruct the data set.

SECTION 3

ISAM DATA SETS

This section discusses the concept and specifics of ISAM data sets. An ISAM data set consists of two physical files:

- a data store file that contains fixed-length records
- an index file containing keys and addresses of the records in the data store file

There is a logical dependency between the data store file and the index file of a data set.

INDEXES AND KEYS

An index is a structure designed to help you locate a particular record of a data set. ISAM bases index keys on the key field values for records contained in a data set. Index keys can be in ascending or descending order. Figure 3-1 illustrates an index for a data set.

An application uses an index to:

- read records in key order
- read a single record using a unique key

An example of sequential access using personnel data sets is reading the employee records in order by employee number, from 100 to 999. An example of direct access is reading the employee record for employee number 15.

Both of these examples read the records using an employee number index. To access records directly, the index must contain unique keys. Unique keys indicate that only one record exists for each key value. In this case, you cannot use duplicate keys.

Index keys can be simple or composite. A single field comprises a simple key. ISAM sorts an index composed of simple keys in the natural order for the key type. This allows applications to sequentially access records in order by key value. Multiple fields make up a composite key.

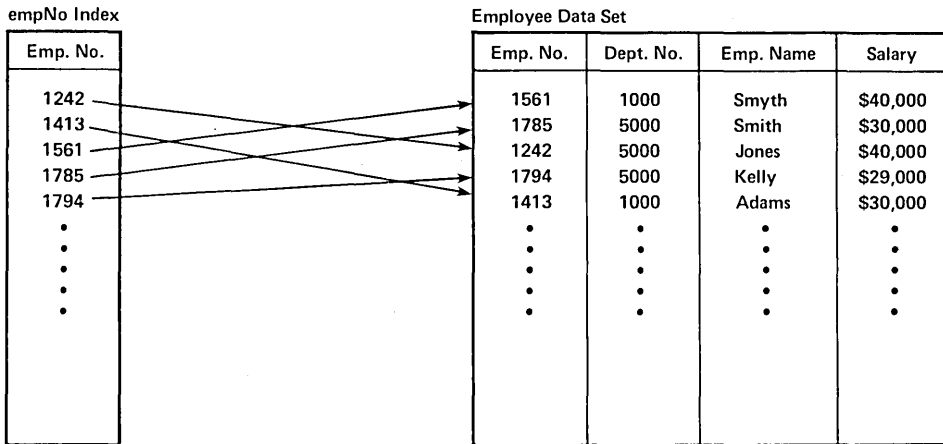


Figure 3-1. Data Set Index (Sample)

ISAM uses the first field to sort an index composed of composite keys. The second field in a series sorts groups of records with duplicate values for the first field, and so forth. If the entire key contains duplicates, ISAM accesses records with duplicate values in random order. If a composite key is unique, no duplication is possible.

In composite keys:

- total key length cannot exceed 64 bytes
- all fields must be character or byte strings
- sort order must be ascending or descending for all fields
- fields must be adjacent in the records and appear in order of significance

For example, for a data set with the following structure:

Byte	Length	Type	Name
0	4	Byte string	deptNo
4	5	Byte string	empNo

You can define a composite key (deptNo,empNo), but not (empNo,deptNo). The order of the fields does not permit (empNo,deptNo) keys.

A single key field can also define a composite key. Using the example above, you define the composite key (deptNo,empNo) as a 9-byte byte string key located at offset 0.

Key Types

ISAM supports different types of keys by which you can use most data representations specified in each of the B 20 programming languages. You can use 12 different key types to specify an index. Each of the 12 types has a notation for non-COBOL applications (types 0 to 11) and a corresponding notation for COBOL applications (types 20 to 31).

Table 3-1 describes each key type. For more information on the relationships between key types and programming language representations, refer to table 3-2.

Table 3-1. Description of Key Types

Key	Description
Binary	<p>an unsigned 1-byte to 8-byte integer</p> <p>The high-address byte of a binary key is significant for determining sort order on the workstation processor. For COBOL COMP fields, the low-address byte is most significant.</p>

Table 3-1. Description of Key Types (Cont)

Key	Description
Byte String	<p>an uninterpreted fixed-length string of 1 to 64 binary bytes</p> <p>The low-address byte is most significant for determining sort order; ISAM distinguishes between uppercase and lowercase ASCII characters. Byte strings have the same representation in all programming languages, including COBOL.</p>
Character String	<p>a fixed-length string of 1 to 64 binary bytes</p> <p>ISAM sorts a character string with the low-address byte as most significant for determining sort order. ISAM sorts character string keys with no distinction between uppercase and lowercase ASCII characters; character strings have the same representation in all programming languages, including COBOL.</p>
Decimal (Odd)/ Decimal (Even)	<p>contains two decimal digits in each byte, except for the last (high-address) byte where the rightmost four bits are sign-reserved</p> <p>This format is the same as COBOL COMP-3. You use decimal (odd) for values that have odd numbers of digits and decimal (even) for values with even numbers of digits. The number of digits before packing determines if the system uses the odd or even decimal type. A decimal key can contain 1 to 18 decimal digits.</p>
Display	<p>for USAGE is DISPLAY numeric fields</p> <p>It supports all of the COBOL sign options. Display keys can be 1 to 19 bytes long, containing 1 to 18 decimal digits.</p>

Table 3-1. Description of Key Types (Cont)

Key	Description
Integer	a signed 1-byte to 8-byte integer The high-address byte of an integer key is the most significant for determining sort order in a workstation application. For COBOL COMP fields, however, the low-address byte is the most significant.
Long/Short/Extended IEEE	for real numbers in Pascal or FORTRAN applications A long IEEE key is 8 bytes long; a short IEEE is 4 bytes; an extended IEEE is 10 bytes.
Long/Short Real	for BASIC applications A long real key is an 8-byte real number; a short real key is a 4-byte real number.

Table 3-2. ISAM Key Types and Programming Language Representations

Language and Key Type	cbIndexField	wType
BASIC Interpreter		
Integer (%)	2	7
ShortReal (!)	4	5
LongReal (#)	8	4
BASIC Compiler		
Integer (%)	2	7
ShortReal (!)	4	5
LongReal (#)	8	4

Table 3-2. ISAM Key Types and Programming Language Representations (Cont)

Language and Key Type	cbIndexField	wType
COBOL		
USAGE is DISPLAY (n-byte) (numeric types)	n	31
USAGE is COMP (n-byte) (signed)	n	27
USAGE is COMP (n-byte) (unsigned)	n	20
USAGE is COMP-3 (n-digit) (n even)	$(n+2)/2$	26
USAGE is COMP-3 (n-digit)	$(n+2)/2$	23
FORTRAN (Microsoft)		
INTEGER*2	2	7
INTEGER*4	4	7
REAL*4	4	9
REAL*8	8	8
DOUBLE PRECISION	8	8
FORTRAN-86		
(same as FORTRAN above)		
TEMPREAL	10	10
Pascal (Microsoft)		
Byte	1	0
Integer	2	7
Real	4	9
SInt	1	7
Word	2	0

SECTION 4

ISAM UTILITIES

This section describes the Executive ISAM commands you can use for data set maintenance. Each command requires exclusive control of the data set. Table 4-1 describes each command.

Section 5 describes **ISAM REORGANIZE**; you use **ISAM REORGANIZE** to recover lost data and to build a data set from any standard access method file with fixed-length records.

Section 6 describes **ISAM INSTALL**; you use **ISAM INSTALL** to install the multiuser ISAM server.

Table 4-1. ISAM Commands

Command	Description
ISAM COPY	copies data set files to produce a new data set (The passwords for the new data set remain the same.)
ISAM CREATE	creates an empty data set with the specified record size and index fields
ISAM DELETE	deletes both the data store and index files of the data set and destroys all the data
ISAM RENAME	renames the data store and index files to rename the data set (The passwords are unchanged.)
ISAM SET PROTECTION	changes the passwords used to gain access to an existing data set
ISAM STATUS	displays information about a data set (ISAM can also print information or write it to a disk file.)

Table 4-1. ISAM Commands (Cont)

Command	Description
ISAM TERMINATE	removes the ISAM server from a secondary partition in a multipartition system

DEFAULT INDEX FILE NAME

Several ISAM commands include an optional field for an index file specification. If you do not enter a file specification, the data store file specification issues one by default. ISAM copies the file specification for the data store and replaces the suffix with .Ind.

For example, if the file specification for the data store is [vol]<dir>DataSet.Isam, the file specification for the index file is [vol]<dir>DataSet.Ind. If the data store file is DataSet, the index file is DataSet.Ind.

ISAM COPY

The ISAM COPY command creates a new data set by copying an existing one. The command copies both the data store file and the index file of the existing data set. Figure 4-1 illustrates the ISAM COPY form. Table 4-2 describes the ISAM COPY form parameters.

```
ISAM COPY
ISAM data set from 
ISAM data set to
[Index file to]
[Overwrite ok?]
```

Figure 4-1. ISAM COPY Form

Table 4-2. ISAM COPY Form Parameters

Parameter	Description
ISAM data set from	file specification for the data store file ISAM copies (If you supply a password, it must be the volume, directory, or file password for both data set files.)
ISAM data set to	file specification for the new data set's data store file (The password must be the volume or directory password for the new data store file.)
[Index file to]	file specification for the index file of the new data set ISAM derives the default file specification from the data store file specification, described in this section. The password must be the volume or directory password for the new index file.
[Overwrite ok?]	If you specify Yes, ISAM performs the copy. If existing files have the same names, ISAM overwrites them without issuing a confirmation message. The default (No) directs ISAM to display a prompt to confirm overwriting the file (if it exists). Overwriting a file destroys the data. You press GO to confirm the overwrite, or press CANCEL or FINISH to deny it.

The following example copies a new data set from an existing one.

```
ISAM COPY
ISAM data set from Employee.ISAM
ISAM data set to EmployeeNew.ISAM
[Index file to]
[Overwrite ok?]
```

Since you did not enter the index file specification, ISAM creates the default index specification, EmployeeNew.Ind. If a file called EmployeeNew.ISAM or EmployeeNew.Ind already exists, ISAM prompts for an overwrite confirmation by default.

ISAM CREATE

The **ISAM CREATE** command creates an empty data set with the record size and index fields that you enter on the ISAM CREATE form. Figure 4-2 illustrates the ISAM CREATE form.

Table 4-3 describes the fields of the ISAM CREATE form.

<p>ISAM Create</p> <p>ISAM data set <input type="text"/></p> <p>[Index file]</p> <p>Record size (e.g., 20 bytes)</p> <p>Index keys (e.g., Byte:10.8.ANU.W)</p> <p>[B-tree node size (default 2 sectors)]</p> <p>[Growth increment for data store file (default 30 sectors)]</p> <p>[Growth increment for index file (default 30 sectors)]</p> <p>[Initial size of data store file (default 30 sectors)]</p> <p>[Initial size of index file (default 30 sectors)]</p> <p>[Overwrite ok?]</p>

Figure 4-2. ISAM CREATE Form

Table 4-3. ISAM CREATE Form Parameters

Parameter	Description
ISAM data set	file specification ISAM creates for the data set's data store file (The password must be the volume or directory password for the new data store file.)
[Index file]	file specification for the new data set's index file The default is described in this section. The password you supply must be the volume or directory password for the new Index file.
Record size (e.g., 20 bytes)	byte size of the records in the new data set. Records must be at least 4 bytes long. Maximum record size is 65,528 bytes.
Index keys (e.g., Byte:10.8.ANU.W)]	parameter list specifying the index fields of the data set You specify each parameter in the list using the following format; do not embed spaces. t:l.o.anu.w or t.o.anu.w t is the field type and can be any of the following: Binary Byte Character Decimal Display Integer LongIEEE ShortIEEE ExtendedIEEE LongReal ShortReal

Table 4-3. ISAM CREATE Form Parameters (Cont)

Parameter	Description
l	is the field length in bytes for byte or character strings, number of digits for decimal, or number of bytes for binary, display, and integer numbers Binary fields default to two bytes if you omit this entry. Long, short, and extended IEEE index fields and long and short real index fields do not use this entry.
o	is the byte offset in the record for the index field (You specify it as a decimal number.)
a	represents ascending key order; D represents descending key order
n	indexes null values (binary 0's); S suppresses null values
u	represents a unique key; D permits duplicate keys
w	is for non-COBOL applications; M is for COBOL applications (This field is optional; default is W.)
[B-Tree node size (default 2 sectors)]	number of sectors in the B-tree nodes for new data set (maximum value is 12 sectors)
[Growth increment for data store file (default 30 sectors)]	
[Growth increment for index file (default 30 sectors)]	
[Initial size of data store file (default 30 sectors)]	

Table 4-3. ISAM CREATE Form Parameters (Cont)

Parameter	Description
[Initial size of index file (default 30 sectors)]	values used to avoid wasting disk space and disk fragmentation caused by excessive numbers of disk extents
[Overwrite ok?]	If you specify Yes, the system creates the data set. The default (No) directs the system to display a prompt to confirm overwriting the existing file when either of the files for the new data set exists. Overwriting an existing file destroys the data in it. You can press GO to confirm the overwrite, or press CANCEL or FINISH to deny the overwrite.

The following example creates the Employee data set from the Personnel example.

```

ISAM Create
ISAM data set                               Employee.ISAM
[Index file]
Record size (e.g., 20 bytes)                 43
Index keys (e.g., Byte:10.8.ANU.W)
[B-tree node size (default 2 sectors)]       Employee.Keys
[Growth increment for data store file (default 30 sectors)]
[Growth increment for index file (default 30 sectors)]
[Initial size of data store file (default 30 sectors)]
[Initial size of index file (default 30 sectors)]
[Overwrite ok?]
    
```

The data set name is Employee.ISAM, and the index file is Employee.Ind, by default. The record size is 43 bytes. Employee.Keys is a text file containing the following index definitions:

```
BYTE:5.4.ANU.W
CHARACTER:30.9.AND.W
BYTE:9.0.ANU.W
```

The data store, index file sizes, and growth increments use default values. You create both files with an initial size of 30 sectors. Each time either of the files is full, the length extends by 30 sectors.

By default, you receive a prompt to confirm overwriting if either Employee.ISAM or Employee.Ind exists.

ISAM DELETE

The **ISAM DELETE** command deletes both the data store file and the index file of the data set. It destroys all the data in the data set.

There is only one parameter in the ISAM DELETE form: ISAM data set. This is the file specification for the data store file. The password you supply must be the volume, directory, or file password for both files of the data set.

The following example deletes the Employee data set of the Personnel example. The system deletes all records in Employee.ISAM and all indexes in Employee.Ind.

```
ISAM Delete
ISAM data set Employee.ISAM
```

ISAM RENAME

The **ISAM RENAME** command changes the name of an existing data set by renaming both the data store file and the index file. You must rename both files.

The system implements **ISAM RENAME** by using two invocations of the BTOS RenameFile operation: one changes the data store file name; the other changes the data set index file name.

There are certain RenameFile operations that are invalid (renaming a file from one volume to another, or renaming a file using an incorrect password). If one of the two required BTOS RenameFile operations is invalid, **ISAM RENAME** detects the error. It then renames the data set using a valid name for both the data store and index files. In this case, one or both of the files can retain the original name.

Figure 4-3 illustrates the ISAM RENAME form.

Table 4-4 describes the parameters of the ISAM RENAME form.

```
ISAM Rename
ISAM data set from [REDACTED]
ISAM data set to
[Index file to]
[Overwrite ok?]
```

Figure 4-3. ISAM RENAME Form

The following example renames the Employee data set of the Personnel example

```
ISAM Rename
ISAM data set from Employee.ISAM
ISAM data set to <NewDir>Employee.ISAM
[Index file to]
Overwrite ok?]
```

The renamed data store file is the same, but it now resides on a different directory on the logged-in volume. The data set file specification constructs the index file specification by default. The new index file name is <NewDir>Employee.Ind.

Table 4-4. ISAM RENAME Form Parameters

Parameter	Description
ISAM data set from	file specification of the renamed data set's data store file (The password must be the volume, directory, or file password for both files of the data set.)
ISAM data set to	file specification for the renamed set's data store file (The volume must be the same volume specified in ISAM data set from . The password must be the volume or directory password for the renamed data store file.)
[Index file to]	file specification for the renamed data set's index file (If you do not make an entry, ISAM derives the default index file specification as described in this section.) The volume must be the same volume on which the existing index file resides. The password must be the volume or directory password for the new index file.
[Overwrite ok?]	Specify Yes to rename a data set The default (No) directs the system to display a prompt to confirm overwriting the existing file (if either of the files for the new data set already exists). Overwriting an existing file destroys the data in it. You can press GO to confirm the overwrite, or press CANCEL or FINISH to deny it.

ISAM SET PROTECTION

The **ISAM SET PROTECTION** command enters or modifies passwords that permit access to a data set. **ISAM SET PROTECTION** does not change file system passwords for the files of the data set. You can change file system passwords using the

Executive **SET PROTECTION** command. Passwords also open data sets in administrator mode (refer to section 2 for more information on the use of administrator mode).

Figure 4-4 illustrates the ISAM SET PROTECTION form.

Table 4-5 explains the parameters on the ISAM SET PROTECTION form.

ISAM Set Protection	
ISAM data set	<input type="text"/>
[Password for modification]	<input type="text"/>
[Password for reading]	<input type="text"/>

Figure 4-4. ISAM SET PROTECTION Form

Table 4-5. ISAM SET PROTECTION Form Parameters

Parameter	Description
ISAM data set	file specification for the data store file of the data set (This is the data set whose passwords you are modifying. The password must be the volume, directory, or file password for both files of the data set.)
[Password for modification]	new password for modify access to the data set (Without this password, any password permits modify access; passwords have a maximum length of 12 characters.)
[Password for reading]	new password for read access to the data set (Without this password, any password permits read access; passwords have a maximum length of 12 characters.)

The following example protects the Employee.ISAM data set for read and modify access.

```
ISAM Set Protection
ISAM data set      Employee.ISAM
[Password for modification] xxy
[Password for reading]   xxz
```

Command password requirements are as follows:

- batch and transaction read modes require either xxy or xxz
- batch and transaction modify modes require xxy
- administrator mode requires the file system password

ISAM STATUS

The **ISAM STATUS** command produces a status report for a data set on the display; you can also print or write it to a disk file. The information displayed includes:

- file names
- sizes
- growth increments
- record and B-tree node sizes
- description of each index, including the depth of each B-tree

If you enter Yes in response to **[Details?]**, the command includes the following additional information:

- number of records (and deleted records) in the data set
- number of B-tree nodes for each index (and the number of deleted nodes)
- number of records indexed under each key
- average percentage of node space currently used for each index's node

Retrieving this additional information involves scanning the files of the data set; **ISAM STATUS** takes significantly longer to execute if you request these details.

Figure 4-5 illustrates the ISAM Status form.

Table 4-6 explains the parameters of the ISAM STATUS form.

The image shows a rectangular box representing the ISAM Status form. Inside the box, the text 'ISAM Status' is at the top left. Below it are three lines of text: 'ISAM data set', '[Log file]', and '[Details?]', each followed by a horizontal rectangular input field. The 'ISAM data set' field is filled with a stippled pattern, while the other two fields are empty.

Figure 4-5. ISAM STATUS Form

Table 4-6. ISAM STATUS Form Parameters

Parameters	Description
ISAM data set	file specification for the data store file for which you want the status report (The password you use must be the volume, directory, or file password for both files of the data set.)
[Log file]	file specification for the file the system writes the status report to (If you do not specify a log file, the report appears only on the display.)
[Details?]	Specify Yes to display additional details. The default (No) directs the system to display the standard information. (Refer to ISAM STATUS information, in this section.)

The following example produces a status report with details shown in the second part of figure 4-6. Figure 4-6 shows both forms of the ISAM STATUS report.

```

ISAM Status
ISAM data set Employee.ISAM
[Log file]
[Details?] Yes

```

ISAM TERMINATE

The **ISAM TERMINATE** command removes the ISAM system service from memory. You can use this command only on a multi-partition operating system and only at the workstation at which you installed ISAM. **ISAM TERMINATE** has no parameters.

Data store file	[Win]<Personnel>Employee.ISAM	
Index file	[Win]<Personnel>Employee.Ind	
Last accessed	Thu Jul 11, 1985 10:09 AM	
Last modified	Thu Jul 11, 1985 10:09 AM	
Record size (bytes)	43	
Node size (sectors)	2	
Size of Data store file (sectors)	5	
Size of Index file (sectors)	10	
Growth increment for Data store file (sectors)	5	
Growth increment for Index file (sectors)	5	
Number of records (valid and deleted)	7	
Number of nodes (valid and deleted)	3	
Index Number	Index Specification	BTree Depth
0	Byte:5.4.ANU.W	1
1	Character:30.9.AND.W	1
2	Byte:9.0.ANU.W	1

Data store file	[Win]<Personnel>Employee.ISAM				
Index file	[Win]<Personnel>Employee.Ind				
Last accessed	Thu Jul 11, 1985 10:09 AM				
Last modified	Thu Jul 11, 1985 10:09 AM				
Record size (bytes)	43				
Node size (sectors)	2				
Size of Data store file (sectors)	5				
Size of Index file (sectors)	10				
Growth increment for Data store file (sectors)	5				
Growth increment for Index file (sectors)	5				
Number of records	7				
Number of deleted records	0				
Number of nodes	3				
Number of deleted nodes	0				
Index Number	Index Specification	BTree Depth	Number of nodes	Number of records indexed	Average Fullness (%)
0	Byte:5.4.ANU.W	1	1	7	7
1	Character:30.9.AND.W	1	1	7	24
2	Byte:9.0.ANU.W	1	1	7	9

Figure 4-6. ISAM STATUS Reports (Samples)

SECTION 5

ISAM REORGANIZATION

The **ISAM REORGANIZE** command builds a data set from any Standard Access Method file of fixed-length records, including the data store file of a data set.

You can use this command independently to:

- initially load a data set from a Direct Access Method (DAM) file
- change the definition of the indexes of an existing data set or add new indexes
- change CreateISAM parameters, such as B-tree node sizes

You can use this command with the **MAINTAIN FILE** command to:

- recover records from a data set damaged by a software or hardware failure
- reclaim space in a data set from which you deleted records
- merge several data sets or DAM files into a new data set

You can use this command with the **SORT** command (described in the B 20 Systems Sort/Merge Reference Manual) to:

- sort the physical records to organize the data set in order by key values
- recover records, reclaim space, and merge data sets (as with the **MAINTAIN FILE** command)
- modify records by changing, adding, deleting, or rearranging parameters

Sorting the records of a data set improves the performance of applications that access the records of the data set in the sort order. No other difference is visible at the application level.

The ISAM distribution diskette contains two forms of the ISAM REORGANIZE run file: resident and swapping. Hard disk systems automatically install the swapping version on the disk, but dual floppy systems use only the resident version. The resident version uses more memory than the swapping form, but performance is better.

ISAM REORGANIZE performs an in-place reorganization. The DAM file or the data store file of the data set becomes the data store file of the reorganized data set. ISAM deletes the index file of the data set and then rebuilds it.

At the end of this section, there are examples of how to use **ISAM REORGANIZE** alone or with **MAINTAIN FILE** or **SORT** to perform the tasks described previously. Figure 5-1 illustrates the ISAM REORGANIZE form.

Table 5-1 explains the parameters on the ISAM REORGANIZE form.


ISAM Reorganize	
ISAM data set or DAM file	
[Index file]	
[Work file 1]	
[Work file 2]	
[Use parameters from ISAM data set?]	
[Index keys (e.g., Byte:10.8.ANU.U)]	
[B-tree node size (2 sectors)]	
[Data store file growth increment (30 sectors)]	
[Index file growth increment (30 sectors)]	
[Initial index file size (30 sectors)]	
[Maximum initial B-tree node fullness (80%)]	
[Overwrite ok?]	

Figure 5-1. ISAM REORGANIZE Form

Table 5-1. ISAM REORGANIZE Form Parameters

Parameter	Description
ISAM data set or DAM file	file specification for a data set If you specify a DAM file, the DAM file becomes the data store file of the new data set. ISAM REORGANIZE can determine whether the file is a DAM file or an ISAM data set by the content of the data file or ISAM data set.
[Index file]	file specification for the index file of the reorganized data set If the file specified in the ISAM data set or DAM file is an existing data set, the index file name defaults to the name of the existing index file for the data set. If Index file changes an existing data set index file name, ISAM deletes the old index file. If the ISAM data set or DAM file above specifies the name of a DAM file, the index file name defaults to the name of the DAM file with the suffix .Ind .
[Work file 1] [Work file 2]	two work files, each approximately $cRecords * (sKeyMax + 8) + 512$ bytes long (The number of records in the data set is $cRecords$, and $sKeyMax$ is the length of the longest key parameter in bytes.) The work files default to <code>[Sys]<\$nnn>ISAMWork1tmp</code> and <code>[Sys]<\$nnn>ISAMWork2tmp</code> (nnn is the workstation number).

Table 5-1. ISAM REORGANIZE Form Parameters (Cont)

Parameter	Description
	<p>If the directory [Sys]<\$nnn> does not exist, the work files default to ISAMWork1tmp and ISAMWork2tmp in the logged-in directory (using the logged-in default file prefix).</p> <p>You can optimize ISAM REORGANIZE performance by putting the work files on different Winchester disk drives.</p>
[Use parameters from ISAM data set?]	<p>If you specify Yes, ISAM reorganizes an existing data set. If you leave any of the parameters in the form blank from [Index keys (e.g., Byte:10.8.ANU.W)] through [Index file growth increment (30 sectors)], ISAM uses the values you specified when you created or last reorganized the data set, instead of the listed defaults.</p> <p>If you specify Yes for this parameter and the index file of the data set was deleted or unusable, a status message appears on the display and ISAM REORGANIZE terminates.</p> <p>If you accept the default (No), for this parameter, you must fill in [Index keys (e.g., Byte:10.8.ANU.W)]. If you do not fill in any of the parameters [B-tree node size (2 sectors)], [Data store file growth increment (30 sectors)], or [Index file growth increment (30 sectors)], ISAM uses the listed default.</p>

Table 5-1. ISAM REORGANIZE Form Parameters (Cont)

Parameter	Description
Index keys (e.g., Byte:10.8.ANU.W)]	parameter list that specifies the index parameters of the data set
	You specify each parameter in the list using the following format, with no embedded spaces.
	t:l.o.anu.w or t.o.anu.w
	t is the field type and can be any of the following:
	Binary Byte Character Decimal Display Integer LongIEEE ShortIEEE ExtendedIEEE LongReal ShortReal
	l is the length of the field in bytes for byte or character strings, the number of digits for decimal, or the number of bytes for binary, display, and integer numbers
	If you omit this entry, binary parameters default to two bytes. Long, short, and extended IEEE index parameters and long and short real index parameters do not use this entry.
	o is the byte offset in the record for the index parameter. You specify it as a decimal.

Table 5-1. ISAM REORGANIZE Form Parameters (Cont)

Parameter	Description
a	represents ascending key order; D represents descending key order
n	indexes null values (binary 0's); S suppresses null values
u	represents a unique key; D permits duplicate keys
w	is for non-COBOL applications; M is for COBOL applications (This parameter is optional; default is W.)
[B-tree node size (2 sectors)]	number of sectors in B-tree nodes for new data set (Maximum value is 12 sectors.)
[Data store file growth increment (30 sectors)]	
[Index file growth increment (30 sectors)]	
[Minimum index file size (30 sectors)]	values you use to avoid wasting disk space and disk fragmentation (caused by excessive numbers of disk extents)
[Maximum initial B-tree node full- ness (80%)]	percentage of allocated space for maximum capacity of each B-tree node; minimum capacity 50% (If you specify a percentage less than 50%, ISAM REORGANIZE ignores it and uses 50%.)

Table 5-1. ISAM REORGANIZE Form Parameters (Cont)

Parameter	Description
[Overwrite ok?]	<p>If you specify yes, the system performs the reorganization without confirmation prompts.</p> <p>Under any of the following conditions, ISAM REORGANIZE issues a prompt to confirm before overwriting the file contents:</p> <ul style="list-style-type: none">• you accept the default (No)• the ISAM data set or DAM file entry specifies a DAM file• the index file exists

LOADING A DATA SET

By using the Executive **COPY** command, you can load a data set from a DAM file. You copy the DAM file to the data store file of the data set. Then you use **ISAM REORGANIZE** to build the indexes for the new data set and to initialize all file structures necessary for ISAM's access to the data set.

If the DAM file, **Employee.DAM**, contains records to load into the Employee data set, **Employee.ISAM**, you copy **Employee.DAM** to **Employee.ISAM**. The following example illustrates this operation.

```
Copy
File from      Employee.DAM
File to        Employee.ISAM
[Overwrite ok?]
[Confirm each?]
```

After invoking **ISAM REORGANIZE**, you fill in the form as follows:

```
ISAM Reorganize
  ISAM data set or DAM file           Employee.ISAM
  [Index file]
  [Work file 1]
  [Work file 2]
  [Use parameters from ISAM data set?]
  [Index keys (e.g., Byte:10.8.ANU.W)] Employee.Keys
  [B-tree node size (2 sectors)]
  [Data store file growth increment (30 sectors)]
  [Index file growth increment (30 sectors)]
  [Initial index file size (30 sectors)]
  [Maximum initial B-tree node fullness (80%)]
  [Overwrite ok?]
```

Employee.Keys is a text file containing the following key specifications:

```
BYTE:5.4.ANU.W
CHARACTER:30.9.AND.W
BYTE:9.0.ANU.W
```

CHANGING INDEXES AND OTHER ISAM CREATE PARAMETERS

You can use **ISAM REORGANIZE** to:

- change the index definitions of an existing data set
- add new indexes
- change other **ISAM CREATE** parameters such as B-tree node sizes

In the Personnel data sets example, the records contained in Employee.ISAM have the following structure:

Offset	Length	Parameter	Type
0	4	deptNo	byte
4	5	empNo	byte
9	30	empName	character
39	4	salary	decimal

EmpNo and empName are defined as keys; there is also a composite key (deptNo,empNo).

In this example, you added an index on the salary parameter. All four indexes now list records in ascending order by key parameter(s), and all four now support indexing of null values. The empName and salary indexes allow duplicates, and the empNo and (deptNo,empNo) keys uniquely identify records.

When the ISAM REORGANIZE form appears, you enter the name of the data set in the ISAM data set or DAM file parameter. Then you specify Yes for [Use parameters from ISAM data set?], and fill in the parameters you want changed.

```

ISAM Reorganize
ISAM data set or DAM file           Employee.ISAM
[Index file]
[Work file 1]
[Work file 2]
[Use parameters from ISAM data set?]  Yes
[Index keys (e.g., Byte:10.8.ANU.W)]  Employee.Keys
[B-tree node size (2 sectors)]
[Data store file growth increment (30 sectors)]
[Index file growth increment (30 sectors)]
[Initial index file size (30 sectors)]
[Maximum initial B-tree node fullness (80%)]
[Overwrite ok?]

```

Employee.Keys is a text file containing the following key specifications:

BYTE:5.4.ANU.W
CHARACTER:30.9.AND.W
BYTE:9.0.ANU.W
DECIMAL:6.AND.W

ISAM REORGANIZE then:

- extracts the old parameters from the data set
- replaces any that have new values with specified values (in this case, only the index keys)
- rebuilds the indexes of the data set

RECOVERING RECORDS, RECLAIMING SPACE, AND MERGING DATA

You use the Executive **MAINTAIN FILE** command and **ISAM REORGANIZE** to:

- recover records from a data set damaged by a software or hardware failure
- reclaim space in a data set from which you deleted records
- merge several data sets or DAM files into a new data set

MAINTAIN FILE performs each of these tasks and produces a single file. This file, along with **ISAM REORGANIZE**, builds the indexes.

The following example shows you how to recover data from Employee.ISAM. You first invoke **MAINTAIN FILE** and fill in the form as follows:

```
Maintain File
Input files           Employee.ISAM
[Output file]        Temp.DAM
[Log file]
[Remove deleted records?]
[Suppress confirmation?]
```

MAINTAIN FILE scans the data store file of the Employee data set, verifying the file structures, recovering data, and reclaiming the space the deleted records occupied. Then ISAM copies the records copied into Temp.DAM.

Next, you use the Executive **COPY** or **RENAME** command to replace the data store file of Employee.ISAM with the file that **MAINTAIN FILE** produces, as follows:

```

Rename
Old file name   Temp.DAM
New file name   Employee.ISAM
[Overwrite ok?] Yes
[Confirm each?]

```

Finally, you use **ISAM REORGANIZE** to build the indexes of the merged data set. The existing index file can supply the index keys, node sizes, etc. (If the index file of the old Employee.ISAM data set does not have the default name, Employee.Ind, you must specify its name in the Index file parameter.)

```

ISAM Reorganize
ISAM data set or DAM file           Employee.ISAM
[Index file]
[Work file 1]
[Work file 2]
[Use parameters from ISAM data set?]
[Index keys (e.g., Byte:10.8ANU.W)]   Yes
[B-tree node size (2 sectors)]
[Data store file growth increment (30 sectors)]
[Index file growth increment (30 sectors)]
[Initial index file size (30 sectors)]
[Maximum initial B-tree node fullness (80%)]
[Overwrite ok?]

```

SORTING DATA SET RECORDS

You use the **SORT** command and **ISAM REORGANIZE** to:

- sort the physical records to organize the data set in order by key values

- recover records, reclaim space, and merge data sets
- modify records by changing, adding, deleting, or rearranging parameters

You modify records by using the Own Code feature in the **Sort** utility. After ISAM sorts the data store file **ISAM REORGANIZE** builds the indexes for the data set. (For more information on Sort, refer to the B 20 Systems Sort/Merge Reference Manual.)

In the Personnel data sets example, the Employee data set, Employee.ISAM, has the following parameters.

Offset	Length	Parameter	Type
0	4	deptNo	byte
4	5	empNo	byte
9	30	empName	character
39	4	salary	decimal

EmpNo and empName are key parameters and (deptNo,empNo) is a composite key.

In this example, a major application system processes records in order by employee number, so it is advantageous to sort the data set by the empNo parameter. You should invoke **Sort** and fill in the form as follows:

```
Sort
Input files      Employee.ISAM
Output files     Temp.DAM
Keys             BYTE:5.4.A.U.
[Stable sort?]
[Work file 1]
[Work file 2]
[Log file]
[Suppress confirmation]
```

Next, you use the Executive **COPY** or **RENAME** command to replace the data store file of Employee.ISAM with the file **Sort** produces, as follows:

```
Rename
Old file name    Temp.DAM
New file name    Employee.ISAM
[Overwrite ok?] Yes
[Confirm each?]
```

Finally, you use **ISAM REORGANIZE** to build the sorted data set's indexes. The existing index file can supply the index keys, node sizes, etc. (If the index file of the old Employee.ISAM data set does not have the default name, Employee.Ind, you must specify its name in the Index file parameter.)

```
ISAM Reorganize
  ISAM data set or DAM file           Employee.ISAM
  [Index file]
  [Work file 1]
  [Work file 2]
  [Use parameters from ISAM data set?]
  [Index keys (e.g., Byte:10.8ANU.W)]   Yes
  [B-tree node size (2 sectors)]
  [Data store file growth increment (30 sectors)]
  [Index file growth increment (30 sectors)]
  [Initial index file size (30 sectors)]
  [Maximum initial B-tree node fullness (80%)]
  [Overwrite ok?]
```


SECTION 6

ISAM SERVER INSTALLATION

ISAM supports both multiuser and single-user access to the ISAM server. The system installs multiuser ISAM in memory as a system service, while an application loads single-user ISAM as a task. If the system installs multiuser ISAM, applications can share access to data sets. An application that opens a data set with single-user ISAM, however, has exclusive use of the data set; no other application can access that data set.

ISAM handles the differences between multiuser and single-user access internally. Once ISAM establishes access, applications are independent of any differences and can run in either environment. (Refer to table 6-1 for more information on the differences between multiuser and single-user access.)

For example, ISAM treats transaction-related constraints associated with multiuser access the same way that it does single-user access. Using the same requirements regardless of access method enables you to write and test an application in a single-user environment before using it in a multiuser environment.

Similarly, asynchronous requests from applications to ISAM use the same procedure, `ISAMRequest`, in both multiuser and single-user environments.

To support multiuser (shared) access to data sets, you must install the ISAM server in memory as a system service at a standalone, master, or cluster workstation. You use the **ISAM INSTALL** command to install multiuser ISAM. The ISAM distribution diskettes contain a default configuration file that you can modify to support the requirements of your installation.

You can also load ISAM as a task of an application by invoking the `LoadSingleUserISAM` operation. Loading the ISAM server from an application allows single-user exclusive access to data sets located at the workstation where the application is running, without installing ISAM as a system service.

You can install two different forms of the ISAM server: resident and swapping. The resident server uses more memory than the swapping form, but performance is better.

The distribution diskettes include both forms; the type of system environment in which you are using ISAM determines which form you use. Hard disk systems automatically install the swapping server on the disk. Systems equipped with floppy disk drives use only the resident server.

MULTIUSER INSTALLATION

The following information pertains to multiuser installation in a single-partition BTOS and in a multipartition BTOS.

Single Partition Operating System

In a single-partition operating system, the system permanently installs ISAM in memory. Once you install it, you cannot remove it or reallocate its memory, unless you reset the system.

Table 6-1. Differences Between Multiuser and Single-User Access

Multiuser Access	Single-User Access
ISAM installs as a system service at the master, cluster, or stand-alone work station.	ISAM loads as a task of the application.
ISAM services requests from all applications.	ISAM services requests only from the application that called it.
Both shared and exclusive access to a data set are available.	Only exclusive access to a data set available; no other application can access a data set that an application opened.
ISAM uses Request and Respond operations and asynchronous processing.	ISAM uses the Send operation for asynchronous processing.

Multipartition Operating System

In a multipartition operating system, the system installs ISAM in a secondary application partition. You can use the **ISAM TERMINATE** command to remove the server from a secondary partition.

ISAM INSTALL

The **ISAM INSTALL** command installs the ISAM server in memory at a standalone, cluster, or master workstation. It uses the values contained in a configuration file to determine how to allocate memory. Figure 6-1 illustrates the ISAM INSTALL form.

Memory Allocation

ISAM server installation requires memory for:

- resident code and data
- a virtual code segment management buffer (swap zone) when it uses the swapping server
- a heap
- data and index buffers

ISAM Install
[Number of ISAM users (default from OS configuration)] <input type="text"/>
[Configuration file (default SysISAM.Config)]

Figure 6-1. ISAM INSTALL Form

Table 6-2. ISAM INSTALL Form Parameters

Parameter	Description
[Number of ISAM users (default from OS configuration)]	<p>specifies the average number of users who use ISAM</p> <p>(The default is the total number of users specified in the BTOS system build configuration file. ISAM determines memory requirements for the specified number of users according to the values specified in the configuration file for the data set.)</p> <p>For example, if a cluster has five users, three of whom use ISAM extensively and two of whom use ISAM occasionally, you should specify four. ISAM allows simultaneous use by all five with some performance degradation.</p>
[Configuration file (default [Sys]<Sys> ISAM.Config)]	<p>specifies the configuration file ISAM uses to determine memory requirements</p> <p>(The default value is [Sys]<Sys>ISAM.Config, which is the configuration file you receive with the distribution diskettes. To modify the default configuration, refer to ISAM CONFIGURE, in this section.)</p>

Resident Code and Data

The following are the code and data requirements for the swapping and resident servers. You can specify other memory areas, although the distribution configuration file contains default values.

Resident code and data
(swapping system service): 39K

Resident code and data
(resident system service): 80K

Swap Zone

For the resident ISAM server, the swap zone size is always 0. The size of the swap zone can vary for the swapping server. In general, the more memory allocated for the virtual code segment, the better ISAM performs. For the swapping form of the ISAM server, the size of the swap zone (virtual code segment management buffer) must be at least 8K of memory; the maximum amount is 48K of memory.

Heap

The heap is an area of memory containing internal ISAM data structures that allocates control blocks for open data sets, indexes, and record locks. If you need a large number of control blocks, you should increase the heap size; however, the size of the heap cannot exceed 40K of memory. This section provides further information on determining heap size.

Data Buffers

Data buffers are fixed-length I/O buffers into which ISAM reads portions of data files. ISAM allocates data buffers as 512-byte sectors. A data buffer must be at least two sectors (1K) and not more than 127 sectors (63.5K). The buffer size should be kept small unless the records in the data set are large.

You determine the total memory area allocated for the data buffers by multiplying the number of sectors needed for a buffer by the total number of buffers required. The number of users determines the number of buffers needed.

This section provides information on how to determine the number of buffers needed; it also gives a more detailed explanation of how to determine the data buffer size.

Index Buffers

Index buffers are fixed-length I/O buffers into which ISAM reads portions of index files. The system allocates Index buffers as 512-byte sectors. An index buffer must be at least one sector (512 bytes) and no more than 12 sectors (6K). You cannot access a data set if the largest B-tree node in the index file does not fit in the index buffer. The system allocates nodes, like index buffers, in full sectors. The index buffer must be no smaller than the largest B-tree node. You determine total memory area allocated for the index buffers by multiplying the number of sectors needed for a buffer by the number of buffers needed.

The number of buffers required depends on the number of users. This section provides further information on determining the number of index buffers you may need.

ISAM CONFIGURE

An ISAM configuration file specifies the sizes of the ISAM server's memory areas; you determine size according to the number of users you specify. The configuration file provided on the distribution diskette has default values for these memory areas. However, changing the values with the **ISAM CONFIGURE** command can improve performance, depending on the typical patterns of access in a particular installation.

The **ISAM CONFIGURE** command creates or changes the configuration file that the **ISAM INSTALL** command uses to determine memory allocation (refer to **ISAM INSTALL**, described in this section). **ISAM CONFIGURE** displays the values contained in the specified configuration file and allows you to change the values.

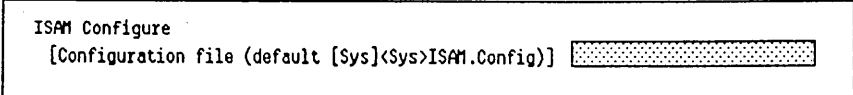
If the specified configuration file does not exist, **ISAM CONFIGURE** creates a new configuration file with the specified name, and inserts default values. The following message appears before **ISAM CONFIGURE** creates the configuration file. (In this case, the term file refers to the specified configuration file.)

File file does not exist. Create?
(Press **GO** to confirm, **CANCEL** to stop command)

If the specified configuration file exists but is not an ISAM configuration file, **ISAM CONFIGURE** overwrites the contents of the configuration file with default configuration values. The following message appears before **ISAM CONFIGURE** overwrites the contents of the specified configuration file:

```
File is not an ISAM configuration file.  
Overwrite?  
(Press GO to confirm, CANCEL to stop command)
```

By changing the values in the form, you can modify the configuration file values. This section also provides detailed information on calculating the size of the memory areas. Figure 6-2 illustrates the ISAM CONFIGURE form.



The screenshot shows a rectangular window titled "ISAM Configure". Inside the window, there is a text prompt "[Configuration file (default [Sys]<Sys>ISAM.Config)]" followed by a rectangular input field with a dotted pattern, indicating it is a text entry area.

Figure 6-2. ISAM CONFIGURE Form

The default configuration file is [Sys]<Sys>ISAM.Config, which is the configuration file provided with the distribution diskettes.

ISAM Configure Display

The ISAM CONFIGURE display appears when you invoke the command. The values from the configuration file specified in **ISAM CONFIGURE** appear. (Refer to figure 6-3.)

By changing the entries in the form, you can change the configuration file.

Cursor Movement

When the ISAM CONFIGURE display appears, the cursor is in the first field. If you press the **NEXT**, **RETURN**, or **TAB** key, the cursor moves from field to field. When the cursor reaches the last field, pressing the **NEXT** or **RETURN** key brings the cursor to the first field.

The cursor-control keys move the cursor vertically or horizontally as follows:

- The **Left Arrow** and **Right Arrow** keys move the cursor within a field.
- **SHIFT-Left Arrow** moves the cursor to the previous field.
- **SHIFT-Right Arrow** moves the cursor to the next field (the same as the **NEXT** and **RETURN** keys).
- The **Up Arrow**, **SHIFT-Up Arrow**, and **SHIFT-Down Arrow** keys move the cursor vertically.
- **CODE-Up Arrow** moves the cursor to the first field.
- **CODE-Down Arrow** moves it to the last field.

To delete one or more characters, you use the following key or set of keys:

- Press **DELETE** to delete one character at a time in a field.
- Press **CODE-DELETE** to delete all field characters.

Display

The following items describe the ISAM CONFIGURE display. They correspond to the circled numbers in figure 6-3.

- ① **ISAM Configuration Utility X.XX** is the display title. X.XX is the version.
- ② **Buffer Sizes (number of 512-byte sectors)** specifies the number of 512-byte sectors to be allocated for each data and index buffer. The maximum number of sectors for data buffers is 127; the index buffer maximum is 12. The default value for both data and index buffers is 2.
- ③ **Number of ISAM Users** specifies the range of numbers ISAM uses as column headings for the tabular portion of the form. Each entry is the number of users for which that column provides configuration information. You can specify up to 10 numbers. You can change the range of numbers as long as the numbers ascend from left to right.

① ISAM Configuration Utility X.XX

② Buffer Sizes (number of 512-byte sectors)

Data Buffers 2

Index Buffers 2

③ Number of ISAM Users

1 2 3 4 5 6 16 32 48 64

④

Heap Size
(K bytes)

4 13 16 28 36 50 63

⑤

Data Buffers
(number)

3 8 10 20 36 52 66

⑥

Index Buffers
(number)

3 16 20 40 72 104 132

⑦

Swap Zone Size (K bytes) 10

⑧

Press GO to confirm changes, or CANCEL to stop command

Figure 6-3. ISAM CONFIGURE Display (Sample)

④ **Heap Size (K bytes)** specifies the number of bytes allocated for the heap, based on the number of users. The minimum size is 1.

There must be at least one entry in this row.

⑤ **Data Buffers (number)** specifies the number of data buffers allocated, based on the number of users. The minimum number is 2.

There must be at least one entry in this row.

⑥ **Index Buffers (number)** specifies the number of index buffers allocated, based on the number of users. The minimum number is 3.

There must be at least one entry in this row.

- ⑦ **Swap Zone Size (K bytes)** specifies the number of bytes allocated for virtual code segment management. The minimum number is 8. The resident server ignores this entry.
- ⑧ **Message line.** Error messages replace this line if any errors occur.

MEMORY ALLOCATION CALCULATION

Before installing the server, **ISAM INSTALL** allocates memory by calculating the memory requirements for resident code and data, a swap zone (if you use the swapping server), a heap, and data and index buffers.

You predetermine memory allocation for resident code and data; the configuration file specifies memory allocation for the swap zone.

ISAM INSTALL uses the configuration file values listed in the **ISAM CONFIGURE** display (see figure 6-3) to calculate memory allocation for the heap and the data and index buffers, based on the number of users accessing **ISAM**.

ISAM INSTALL calculates the heap size by selecting the **Heap Size** entry for the appropriate number of users. If the specified number of users is a column entry, **ISAM INSTALL** then reads the specified heap size from that column in the **Heap Size** row. If **ISAM INSTALL** does not find that number of users, or if the corresponding column in the **Heap Size** row is blank, it calculates heap size by either interpolating between or extrapolating from the existing entries.

ISAM INSTALL calculates the memory area needed for the data buffers from two entries in the configuration file. Then it determines the actual number of data buffers to use in the same way it determines heap size, using the **Data Buffers (number)** entries in the tabular section of the form. **ISAM INSTALL** then multiplies this number by the **Buffer Sizes** entry for data buffers. The product is the size of the memory area allocated for all the data buffers. This section provides further information on calculating the size of the data buffers.

ISAM INSTALL calculates the memory area the index buffers need in the same way it calculates the area the data buffers need; it uses the **Index Buffers** entries.

For example, the default configuration file contains the heap size and the number of data and index buffers for 1, 4, 6, 16, 32, 48, and 64 users. If you specify four users during installation, then **ISAM INSTALL** allocates 13K-bytes of memory for the heap, eight 1024-byte data buffers, and 16 1024-byte index buffers.

If you specify three users in the previous example, **ISAM INSTALL** interpolates values for the heap size and the number of data and index buffers, then calculates their memory allocation. If you specify more than 64, **ISAM INSTALL** extrapolates the values for the heap size and the number of data and index buffers. **ISAM INSTALL** then calculates their memory allocation.

BUFFER SIZE GUIDELINES

Whenever ISAM reads a record or B-tree node into memory, it reads in the entire record or node. Therefore, buffers must be large enough to accommodate the largest record or B-tree node. Data buffers must be large enough to hold the largest record in the data set. This requirement becomes complex because records can overlap sector boundaries, but I/O operations always access whole sectors. If a record overlaps two sectors, ISAM must read both sectors into the data buffer.

The rule for allocating data buffers is:

$$\text{buffer size (bytes)} = \text{record size (bytes)} + \text{overhead (8 bytes)} + \text{overlap sector (511 bytes)}, \text{ rounded up to full sectors}$$

If the record size + 8-byte overhead is a power of 2, the records align on sector boundaries, and the calculation does not need the extra 511 bytes for sector overlap.

The number of buffers allocated depends on the number of users. The minimum is two data buffers. Random-access performance improves with more buffers because this increases the probability that a record still exists in memory if you need it a second time. Sequential-access performance improves with larger buffers.

SECTION 7

ISAM OPERATIONS

You use the ISAM procedures and services in applications to operate on data in ISAM data sets. You can write ISAM applications in any of the B 20 programming languages.

Table 7-1 categorizes procedures and services by function. The first part of this section contains brief descriptions and general information on each functional category. The operations are then presented in alphabetical order with a brief description of the operation, the procedural interface, and the request block parameters where applicable.

You access most ISAM services by a procedural interface or by the ISAMRequest and Wait operations. You use the ISAMRequest operation to send requests to ISAM; multiuser ISAM uses the Request operation while single-user ISAM uses the Send operation.

Using the procedural interface is easier because the system performs most of the work automatically. Using the ISAMRequest and Wait operations allow for a greater degree of overlap between computation and I/O operations.

Previous applications can use operations that are no longer standard in Release 5.0. Programs that call these operations still run with ISAM 5.0. Appendix B lists services that are no longer part of standard ISAM operations.

STATUS BLOCK

All ISAM operations include the pStatusBlockRet parameter, which points to the address of a status block used to report errors to the application. The 4-byte status block contains two status codes, erc and ercDetail, as shown in table 7-2.

Erc is either 0 (OK) or one of the ISAM status codes listed in appendix A. If erc is nonzero, ercDetail gives additional information about the error, for example, when a sector of the index file becomes unreadable because of a device error. Erc contains 3119 (**Index file error**) and ercDetail contains 301 (**I/O error**).

Table 7-1. ISAM Operations by Function

Data Set Management	Record Locking
CreateISAM	HoldISAMDataSet
DeleteISAM	HoldISAMRecord
RenameISAM	ReleaseISAMDataSet
SetISAMProtection	ReleaseISAMRecord
Data Set Access	Transactions
CloseISAM	BeginTransaction
OpenISAM	CommitTransaction
	QueryTransactionParams
	RollBackTransaction
	SetTransactionParams
Record Management	ISAM Service Access
DeleteISAMRecord	LoadSingleUserISAM
DeleteISAMRecordByKey	VerifyMultiuserISAM
ModifyISAMRecord	
ModifyISAMRecordByKey	
StoreISAMRecord	
Single Record Access	Asynchronous Requests
ReadISAMRecordByUri	ISAMRequest
ReadISAMRecordByUriHold	NormalizeISAMStatus
ReadUniqueISAMRecord	
ReadUniqueISAMRecordHold	
Multiple Record Access (Iteration)	
GetISAMRecords	
GetISAMRecordsHold	
ReadNextISAMRecord	
ReadNextISAMRecordHold	
SetUpISAMIterationLimits	
SetUpISAMIterationPrefix	

Table 7-2. Status Block Format (pStatusBlockRet Parameter)

Offset	Size (bytes)	Parameter	Contents
0	2	erc	Status code (see appendix A)
2	2	ercDetail	Detail status code

If you use ISAM operations within a COBOL application, the two bytes of each status code appear in reverse order. You can display the status codes correctly by first calling ConvertWord for each status code.

DATA SET MANAGEMENT

Table 7-3 describes data set management operations. Section 3 describes ISAM data sets.

ISAM DESCRIPTION BLOCK

When you create a data set with the CreateISAM operation, an ISAM Description Block supplies a data set description that includes the record size, key description, and sector allocation policies. Table 7-4 illustrates the block structure.

Table 7-3. Data Set Management Operations

Operation	Description
CreateISAM	data set index structure; creates a new, empty data set with the specified structure
DeleteISAM	deletes the open data set files, and destroys the data set
RenameISAM	changes the name of an existing data set (the data store and index file name)
SetISAMProtection	changes the data set passwords

Table 7-4. ISAM Description Block

Offset	Parameter	Size (bytes)	Description
0	lfaInitSize- IndexFile	4	initial index file size of the data set (It must be a multiple of 512. The size defaults to 15360, 30 sectors, if lfaInitSizeIndexFile is 0.)
4	qbGrowIndex- File	4	number of bytes by which the index file grows when memory is exhausted (The value must be a multiple of 512. The index file grows by a default of 15360, 30 sectors, if qbGrowIndexFile is 0.)
8	cSectorsNode	2	B-tree nodes size in sectors (cSectorsNode must not exceed the index buffer size specified in the configuration file you use to install ISAM. It defaults to that buffer size.)
10	lfaInitSize- DataStoreFile	4	initial data store file size of the data set (It must be a multiple of 512. The size defaults to 15360, 30 sectors, if lfaInitSizeDataStoreFile is 0.)

Table 7-4. ISAM Description Block (Cont)

Offset	Parameter	Size (bytes)	Description
14	gbGrowData-File	4	number of bytes by which the data store file grows when file space is gone (The value must be a multiple of 512. The data store file grows by a default of 15360, 30 sectors if gbGrowDataStoreFile is 0.)
18	sRecord	2	size in bytes of data set records (Records must be at least four bytes.)
20	cIndexes	2	number of data set indexes
22	rgIndexSpec	Indexes *20	data set Index Specification Blocks (There is one ISAM Index Specification Block per index. The ISAM Index Specification Block structure is shown in table 7-5.)

Table 7-5. ISAM Index Specification Block

Offset	Parameter	Size (bytes)	Description
0	rbIndexField	2	key component offset in each data set record
2	cbIndexField	2	key component size in bytes (The fields wType and cbIndexField together specify the type and size of the key component in each ISAM Index specification Block. Refer to table 7-6.)

Table 7-5. ISAM Index Specification Block (Cont)

Offset	Parameter	Size (bytes)	Description
4	wType	2	one of the values 0 to 11 (20 to 31 for COBOL) used to represent a key type as in table 7-6 (The fields wType and cbIndexField together specify the type and size of the key component in each ISAM Index Specification Block. Refer to table 7-6.)
6	fAscending	2	TRUE (OFFh) if keys for this index are in ascending order; FALSE (0h) if the keys are in descending order
8	fNullIsIndexed	2	TRUE (OFFh) if null values (binary 0's) are indexed; FALSE (0h) if null values are not indexed
10	fDuplicates- Allowed	2	TRUE (OFFh) if duplicate values are valid for this index; FALSE (0h) if ISAM prevents a record from being stored or modified when a key value duplicates the key of a data set record When fDuplicatesAllowed is FALSE, you use index to directly access the data set records; each key is unique
12	pad	8	Reserved

Table 7-6. Type of Key Component

Type	Name of Type	Description
0	Binary	cbIndexField contains the key length in bytes (1 to 8 are valid values)
1	Byte	cbIndexField contains the key length in bytes (1 to 64 are valid values)
2	Character	cbIndexField contains the key length in bytes (1 to 64 are valid values)
3	Decimal (Odd)	cbIndexField contains $(d + 2)/2$; d is the number of decimal digits in the key ($d \leq 18$)
4	LongReal	cbIndexField must contain 8
5	ShortReal	cbIndexField must contain 4
6	Decimal (Even)	See Decimal (Odd) above for the value of cbIndexField (You use this type for keys with an even number of decimal digits.)
7	Integer	cbIndexField contains the key length in bytes (1 to 8 are valid values)
8	LongIEEE	cbIndexField must contain 8
9	ShortIEEE	cbIndexField must contain 4
10	ExtendedIEEE	cbIndexField must contain 10
11	Display	cbIndexField contains the key length in bytes (1 to 19 are valid values.)

COBOL applications use the values 20 to 31 for the key types listed in this table.

DATA SET ACCESS

There are two Data Set Access and ISAM handle operations:

CloseISAM (closes an open data set)

OpenISAM (opens an existing data set)

An ISAM handle is a word value which identifies an open data set. The OpenISAM operation returns the ISAM handle; in subsequent ISAM operations you use it as the parameter ISAMhandle. An ISAM handle is valid until you close or delete the data set.

RECORD MANAGEMENT AND ACCESS

Table 7-7 describes Record Management Operations, table 7-8 describes Single Record Access Operations, and table 7-9 describes Multiple Record Access (Iteration) Operations.

Table 7-7. Record Management Operations

Operation	Description
DeleteISAMRecord	removes a data set record (identified by its unique record identifier)
DeleteISAMRecordByKey	removes a data set record (identified by a unique key)
ModifyISAMRecord	modifies an existing record (identified by its unique record identifier) in the data set and updates all indexes accordingly
ModifyISAMRecordByKey	modifies a data set record (identified by a unique key) and updates all indexes accordingly
StoreISAMRecord	creates a new data set record with specified data

Table 7-8. Single Record Access Operations

Operation	Description
ReadISAMRecordByUri	reads a record identified by its unique record identifier
ReadISAMRecordByUri-Hold	reads and locks a record identified by its unique record identifier
ReadUniqueISAMRecord	reads a record identified by a unique key
ReadUniqueISAMRecord-Hold	reads and locks a record identified by a unique key

Table 7-9. Multiple Record Access (Iteration) Operations

Operation	Description
GetISAMRecords	reads several records or unique record identifiers in key order
GetISAMRecordsHold	reads and locks several records or unique record identifiers in key order
ReadNextISAMRecord	reads the next record in key order
ReadNextISAMRecord-Hold	reads and locks the next record in key order
SetUpISAMIteration-Limits	initializes a sequence of Read operations for records with a specific key within a given range
SetUpISAMIteration-Prefix	initializes a sequence of Read operations for records with a specific byte or character string key having a given prefix

LOCKING

Table 7-10 describes Locking operations.

TRANSACTIONS

Table 7-11 describes Transaction operations.

Table 7-10. Locking Operations

Operation	Description
HoldISAMDataSet	locks an ISAM data set
HoldISAMRecord	locks a record identified by its unique record identifier
ReleaseISAMDataSet	unlocks a locked data set without ending the current transaction
ReleaseISAMRecord	unlocks a locked record without ending the current transaction

Table 7-11. Transaction Operations

Operation	Description
BeginTransaction	marks the start of a transaction
CommitTransaction	signifies a successful transaction completion; unlocks all records and data sets locked by the application
QueryTransactionParams	accesses parameters associated with transactions for the application
RollBackTransaction	signifies the unsuccessful completion of a transaction; unlocks all records locked by the application (This operation does not undo any changes made to ISAM data sets.)
SetTransactionParams	sets parameters associated with transactions for the application

Transaction-Related Constraints

The constraints associated with each ISAM operation used during transaction processing are shown in table 7-12. These constraints do not apply to operations on data sets open for batch access (in administrator, batch modify, or batch read mode).

Transaction Parameters Block

The Transaction Parameters Block (refer to table 7-13) contains values that control transaction operations. The wTicksWait parameter specifies the maximum time an application can wait to lock a record or data set. The other parameters do not affect ISAM data set access. (For further information about the use of wTicksWait, refer to section 2.)

You can examine these parameters with the QueryTransactionParams operation and change them with the SetTransactionParams operation.

Table 7-12. Transaction-Related Constraints

Must Be in Transaction	Must Not Be in Transaction	No Transaction Constraints
CommitTransaction	BeginTransaction	CloseISAM
DeleteISAMRecord		CreateISAM*
DeleteISAMRecordByKey		DeleteISAM**
GetISAMRecordsHold		GetISAMRecords
HoldISAMDataSet		ISAMRequest
HoldISAMRecord		OpenISAM
ModifyISAMRecord		QueryTransaction-Params
ModifyISAMRecordByKey		ReadISAMRecordByUri
NormalizeISAMStatus		ReadNextISAMRecord
ReadISAMRecordByUriHold		ReadUniqueISAMRecord
ReadNextISAMRecordHold		RenameISAM**
		RollBackTransaction
		SetISAMProtection**

* Not applicable. A data set is not open when you create it.

**Not applicable. A data set must be open in administrator mode to use this operation.

Table 7-12. Transaction-Related Constraints (Cont)

Must Be in Transaction	Must Not Be in Transaction	No Transaction Constraints
ReadUniqueISAMRecord-Hold		SetTransactionParams
ReleaseISAMDataSet		SetUpISAMIteration-Limits
ReleaseISAMRecord		SetUpISAMIteration-Prefix
StoreISAMRecord		

Table 7-13. Transaction Parameters Block Format

Offset	Length	Parameter	Description
0	2	wTicksWait	A word specifying the maximum number of 0.1-second ticks that a request waits when it tries to lock a record or data set (The default value is 100.)
5	1	Reserved	Padding returned as 0 (null)
6	*	Reserved	The asterisk indicates that it is for expansion and not required at present.

ISAM SERVICE ACCESS

Table 7-14 describes ISAM service access operations and how to use single-user ISAM.

An application initializes single-user ISAM by calling the LoadSingleUserISAM operation. The single-user ISAM server does not inform BTOS that ISAM is serving requests. Instead, the application builds request blocks and sends the request directly to ISAM using the BTOS Send operation rather than the BTOS Request operation.

ISAM sends the requests back using the BTOS Send operation instead of the BTOS Respond operation.

MEMORY USAGE

Single-user ISAM requires the same amount of memory as multiuser ISAM installed for a single user. (Refer to section 1 for memory requirements information.) When you invoke the LoadSingleUserISAM operation, ISAM allocates this memory as short-lived memory from the unallocated memory pool available to the application.

USING EITHER MULTIUSER OR SINGLE-USER ISAM

You can write an application to use either multiuser ISAM or single-user ISAM. By incorporating calls to both multiuser and single-user ISAM, an application is not dependent on multiuser ISAM server installation.

The application calls the VerifyMultiUserISAM operation to check if multiuser ISAM is available. If it is, a status code of 0 (OK) returns, and the application uses multiuser ISAM. If any other status code returns, the application calls the LoadSingleUserISAM operation to load single-user ISAM.

ASYNCHRONOUS REQUESTS

Table 7-15 describes operations and asynchronous requests versus procedural requests.

Table 7-14. ISAM Service Access

Operations	Description
LoadSingleUserISAM	loads ISAM as a task and initializes communications with ISAM
VerifyMultiUserISAM	establishes whether or not multi-user access to ISAM is available on the standalone, cluster, or master workstation

Table 7-15. Asynchronous Requests

Operations	Description
ISAMRequest	issues an ISAM request and returns without waiting for request completion
NormalizeISAMStatus	ensures the validity of a status block returned by an asynchronous ISAM operation

You can indirectly access ISAM system services by a procedural interface, or directly access them by the ISAMRequest, Wait, and NormalizeISAMStatus operations.

Using the procedural interface is easier because it automatically performs most necessary housekeeping and issues the ISAMRequest and Wait operations.

Using the ISAMRequest, Wait, and NormalizeISAMStatus operations enables applications to run more quickly and efficiently because there is a greater degree of overlap between processing by ISAM and computation by the application.

Only one ISAM operation should be outstanding at a time because the order in which ISAM performs the operations is undefined. Overlapping certain operations can cause problems (for example, doing a CommitTransaction operation while a ModifyISAMRecord operation is pending).

When using asynchronous requests, the NormalizeISAMStatus operation must follow the Wait operation. Otherwise, the values in the status block can be invalid.

PROCEDURE DEFINITIONS

The following information presents all ISAM procedures in alphabetical order.

BeginTransaction Procedure

The BeginTransaction procedure marks the start of a transaction for the application. The transaction ends when the

application uses either a CommitTransaction or a RollBackTransaction operation.

You cannot call BeginTransaction from within a transaction. BeginTransaction is an object module procedure.

The procedural interface is:

Begintransaction (pStatusBlockRet): ErcType

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

CloseISAM Procedure

The CloseISAM procedure closes and releases all resources associated with an open data set. An ISAM handle for the closed data set is not valid after an application calls this procedure. If an application opens the same data set more than once, CloseISAM releases all records which were held for the data set, regardless of the handle used. CloseISAM is an object module procedure.

The procedural interface is:

CloseISAM (ISAMHandle, pStatusBlockRet): ErcType

ISAMHandle is the ISAM handle that identifies the open data set.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

CommitTransaction Procedure

The CommitTransaction procedure signifies the successful completion of a transaction and unlocks all records and data sets locked by the application.

You can use CommitTransaction only in a transaction. CommitTransaction is an object module procedure.

The procedural interface is:

CommitTransaction (pStatusBlockRet): ErcType

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

CreateISAM Procedure

The CreateISAM service defines the index structure of a data set and creates a new empty data set with the specified structure. The parameters supplied enable the application program to locate the data store and index files on different volumes and to control the allocation of disk sectors for the two files independently.

The procedural interface is:

```
CreateISAM (pbFileSpecDataStoreFile,  
            cbFileSpecDataStoreFile,  
            pbPasswordDataStoreFileCreate,  
            cbPasswordDataStoreFileCreate,  
            pbFileSpecIndexFile, cbFileSpecIndexFile,  
            pbPasswordIndexFileCreate,  
            cbPasswordIndexFileCreate, pISAMDesc,  
            sISAMDesc, pStatusBlockRet): ErcType
```

pbFileSpecDataStoreFile and cbFileSpecDataStoreFile describe a file specification for the data store file of the new data set.

pbPasswordDataStoreFileCreate and cbPasswordDataStoreFileCreate describe a password used to create the data store file of the new data set.

pbFileSpecIndexFile and cbFileSpecIndexFile describe a file specification for the index file of the new data set. If cbFileSpecIndexFile is 0, the file specification for the index is derived from the file specification for the data store file.

ISAM copies the file specification defined by pbPasswordDataStoreFileCreate and cbPasswordDataStoreFileCreate, then replaces the suffix beginning with the period character with the characters .Ind.

For example, if the file specification for the data store file is [vol]<dir>DataSet.Isam, the file specification for the index file is [vol]<dir>DataSet.Ind.

pbPasswordIndexFileCreate and cbPasswordIndexFileCreate describe a password used to create the index file of the new

data set. If the name of the index file is null, these two parameters are ignored. The system uses the password specified for the data store file to create the index file as well.

pISAMDesc and sISAMDesc describe a memory area containing an ISAM Description Block which has the structure shown in table 7-4.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-16 details the CreateISAM request block.

Table 7-16. CreateISAM Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	0
2	nReqPbCb	1	5
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	79
12	pbFileSpecDataStoreFile	4	
16	cbFileSpecDataStoreFile	2	
18	pbPasswordDataStore- FileCreate	4	
22	cbPasswordDataStore- FileCreate	2	
24	pbFileSpecIndexFile	4	
28	cbFileSpecIndexFile	2	
30	pbPasswordIndexFile- Create	4	
34	cbPasswordIndexFile- Create	2	
36	pISAMDesc	4	
40	sISAMDesc	2	
42	pStatusBlockRet	4	
46	sStatusBlock	2	4

DeleteISAM Procedure

The DeleteISAM service deletes the files of a data set, thereby destroying all information in the data set. The data set must be opened in administrator mode.

The procedural interface is:

```
DeleteISAM (ISAMHandle, pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-17 describes the DeleteISAM request block.

DeleteISAMRecord Procedure

The DeleteISAMRecord service removes a record from a data set. You must open the data set in modify mode to change it. The disk space occupied by the record is available for a subsequent StoreISAMRecord operation. ISAM automatically removes all keys for the record from the indexes of the data set and destroys data in the record.

Table 7-17. DeleteISAM Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	2
2	nReqPbCb	1	0
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	80
12	ISAMHandle	2	
14	pStatusBlockRet	4	
18	sStatusBlock	2	4

If the data set is in transaction mode, you can call DeleteISAMRecord only from within a transaction and when the record to be deleted is locked.

The procedural interface is:

```
DeleteISAMRecord (ISAMHandle, uriRecord,
                  pStatusBlockRet): ErcType
```

ISAMHandle the ISAM handle that identifies the open data set.

uriRecord is the unique record identifier of the deleted record. UriRecord is the value returned by a previous Read operation.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-18 describes the DeleteISAMRecord request block.

Table 7-18. DeleteISAMRecord Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	0
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	81
12	ISAMHandle	2	
14	uriRecord	4	
18	pStatusBlockRet	4	
22	sStatusBlock	2	4

DeleteISAMRecordByKey Procedure

The DeleteISAMRecordByKey service removes a record from a data set. The data set must be open in modify mode to change it. DeleteISAMRecordByKey, however, uses a unique key to identify the record instead of a unique record identifier.

The disk space occupied by the record is available for a subsequent StoreISAMRecord operation. ISAM automatically removes all keys for the record from the indexes of the data set and destroys all data in the record.

If the data set is open in a transaction mode, you can call DeleteISAMRecordByKey only from within a transaction. The record to be deleted does not have to be locked. DeleteISAMRecordByKey locks the record before deleting it.

The procedural interface is:

```
DeleteISAMRecordByKey (ISAMHandle, iIndex, pKey, sKey,  
                       pUriRecordRet, pStatusBlockRet):  
                       ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

iIndex identifies the index used. (The indexes are numbered from 0 in the order specified in CreateISAM.)

pKey and sKey describe the memory area containing the key that identifies the record to be deleted. sKey must be the correct length, in bytes, of the key for index iIndex.

pUriRecordRet is the memory address of the 4-byte structure where the unique record identifier for the record to be deleted is returned.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-19 details the DeleteISAMRecordByKey request block.

GetISAMRecords and GetISAMRecordsHold Procedures

With a single call to ISAM, the GetISAMRecords service reads several records or unique record identifiers in key order from a data set. Records are returned in key order for the current iteration. If there are no more records for the

Table 7-19. DeleteISAMRecordByKey Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	4
2	nReqPbCb	1	1
3	nRespPbCb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	199
12	ISAMHandle	2	
14	iIndex	2	
16	pKey	4	
20	sKey	2	
22	pUriRecordRet	4	
26	sUriRecord	2	4
28	pStatusBlockRet	4	
32	sStatusBlock	2	4

iteration, the status code 3127 (No more records) is returned.

The GetISAMRecordsHold service is identical to GetISAMRecords, except GetISAMRecordsHold also locks the records read (or those that have unique record identifiers returned). When records are read, each one is locked. If unique record identifiers are returned, the record identified by each unique record identifier is locked.

If you use GetISAMRecordsHold, none of the accessed records can be accessed by other applications until you release them, usually by a subsequent CommitTransaction operation.

The GetISAMRecords and GetISAMRecordsHold operations return as many records (or unique record identifiers) in sequence as possible, subject to the following constraints:

- The buffer capacity is not exceeded.
- The range specified for the current iteration is not exceeded.
- Records locked by other applications are not read.

For example, a particular GetISAMRecords operation reads 4-byte unique record identifiers and 40-byte records into a

500-byte buffer. The buffer can contain up to 11 records and unique record identifiers (440 + 44). If the seventh record in the sequence is locked by another application, the GetISAMRecords operation returns only the first six records and unique record identifiers.

If an application uses this service to get records from a remote workstation, it can encounter the status code 45 (**Request block too large**). This occurs when you specify a buffer size too large for the operating system to accommodate. You can either decrease the buffer size or increase line and transmission buffers at a system build.

If GetISAMRecords is called to read only unique record identifiers, it returns unique record identifiers for locked records. Applications can "skip over" records locked by other applications.

If the data set is open in a transaction mode, the application does not have to be in a transaction before calling GetISAMRecords, but it must be in a transaction before calling GetISAMRecordsHold.

The procedural interface is:

```
GetISAMRecords and
GetISAMRecordsHold (ISAMHandle, fReadRecords, pBuffer,
                    sBuffer, pCRecordsReadRet,
                    pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

fReadRecords specifies whether records and unique record identifiers are returned. fReadRecords is FALSE if only unique record identifiers are returned, and TRUE if records are returned also.

pBuffer and sBuffer describe the memory area into which the unique record identifiers and records are to be read.

If only unique record identifiers are returned (fReadRecords is FALSE), the record identifiers are packed into the buffer without padding. If both unique record identifiers and records are returned (fReadRecords is TRUE), the unique record identifiers and records are packed together into the buffer without padding. Each record and its record identifier are packed as a pair with the record identifier preceding the record.

For example, table 7-21 shows the buffer structure after three 46-byte records are read.

pCRecordsReadRet is the memory address of the word where the number of unique record identifiers and/or records read is returned.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-20 describes the GetISAMRecords and GetISAMRecordsHold request block.

Table 7-20. GetISAMRecords and GetISAMRecordsHold Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	4
2	nReqPbCb	1	0
3	nRespPbCb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	*
12	ISAMHandle	2	
14	fReadRecords	1	
15	reserved	1	0
16	pBuffer	4	
20	sBuffer	2	
22	pCRecordsReadRet	4	
26	sCRecordsRead	2	2
28	pStatusBlockRet	4	
32	sStatusBlock	2	4

*GetISAMRecords = 82; GetISAMRecordsHold = 0.

Table 7-21. Buffer Structure for GetISAMRecords and GetISAMRecordsHold when Records are Read (46-Byte Records)

Offset	Description	Size (bytes)
0	Record identifier of record 1	4
4	Record 1	46
50	Record identifier of record 2	4
54	Record 2	46
100	Record identifier of record 3	4
104	Record 3	46

HoldISAMDataSet Procedure

The HoldISAMDataSet service locks an open data set. If the data set is opened in a transaction mode, HoldISAMDataSet can be called only from within a transaction.

The procedural interface is:

```
HoldISAMDataSet (ISAMHandle, fLockRecords, pStatusBlockRet):
    ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

fLockRecords specifies whether a subsequent ReleaseISAMDataSet operation unlocks the data set's locked records as well as the data set itself. fLockRecords is either TRUE or FALSE.

If fLockRecords is TRUE and if the data set is unlocked by a subsequent ReleaseISAMDataSet operation, locked records in the data set continue to be locked.

If fLockRecords is FALSE and if the data set is unlocked by the ReleaseISAMDataSet operation, all records in the data set are unlocked.

For example, data set X has five records and the following events occur:

1. Record 1 is locked (for example, with the HoldISAMRecord operation).
2. Data set X is locked with the HoldISAMDataSet operation.
3. Record 2 is locked (for example, with the ReadNextISAMRecordHold operation).
4. Data set X is unlocked with the ReleaseISAMDataSet operation.

If fLockRecords is TRUE in step 2, Record 1 and Record 2 remain locked after step 4. If fLockRecords is FALSE in step 2, Record 1 and Record 2 are unlocked after step 4.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-22 describes the HoldISAMDataSet request block.

Table 7-22. HoldISAMDataSet Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	4
2	nReqPbCb	1	0
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	205
12	ISAMHandle	2	
14	fLockRecords	1	
15	reserved	1	0
16	pStatusBlockRet	4	
20	sStatusBlock	2	4

HoldISAMRecord Procedure

The HoldISAMRecord service locks a record identified by its unique record identifier. The record identifier is not checked for validity by HoldISAMRecord.

If the data set is opened in a transaction mode, HoldISAMRecord can be called only from within a transaction.

The procedural interface is

```
HoldISAMRecord (ISAMHandle, uriRecord,  
                pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

uriRecord is the unique record identifier of the record to be locked. uriRecord is usually the value returned by a previous Read operation.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-23 details the HoldISAMRecord request block.

Table 7-23. HoldISAMRecord Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	0
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	130
12	ISAMHandle	2	
14	uriRecord	4	
18	pStatusBlockRet	4	
22	sStatusBlock	2	4

ISAMRequest Procedure

The ISAMRequest procedure issues an ISAM request and returns without waiting for the request to be completed.

This operation is used to issue asynchronous ISAM requests in both multiuser and single-user ISAM. For multiuser ISAM, ISAMRequest uses the Request operation, but for single-user ISAM, ISAMRequest uses the Send operation. ISAMRequest allows applications to issue asynchronous requests without regard to whether multiuser or single-user ISAM is being used. (For more information on the Request and Send operations, refer to the B 20 Systems Operating System (BTOS) Reference Manual.)

ISAMRequest is an object module procedure.

The procedural interface is:

```
ISAMRequest (pRq): ErcType
```

pRq is the memory address of the request block specifying the ISAM.

LoadSingleUserISAM Procedure

The LoadSingleUserISAM procedure loads ISAM as a task and initializes communication with ISAM. The ISAM task is loaded into short-lived memory allocated from the pool of unallocated memory pool available to the application.

LoadSingleUserISAM is an object module procedure.

The procedural interface is:

```
LoadSingleUserISAM (pbFileSpecRunFile,  
                   cbFileSpecRunFile, pbPwRunFile,  
                   cbPwRunFile, pbConfigFile,  
                   cbConfigFile, pbPwConfigFile,  
                   cbPwConfigFile, pStatusBlockRet):  
ercType
```

pbFileSpecRunFile and cbFileSpecRunFile describe the memory area containing the ISAM run file name. If cbFileSpecRunFile is 0, the run file name defaults to [Sys]<Sys>ISAMServer.Run.

pbPwRunFile and cbPwRunFile describe the memory area containing the password used to open the ISAM run file.

pbConfigFile and cbConfigFile describe the memory area containing the ISAM configuration file name. If cbConfigFile is 0, the configuration file name defaults to [Sys]<Sys>ISAM.Config.

pbPwConfigFile and cbPwConfigFile describe the memory area containing the password used to open the ISAM configuration file.

pStatusBlockRet is the memory address of the status block into which the status from the operation are returned.

ModifyISAMRecord Procedure

The ModifyISAMRecord service modifies an existing record in a data set and updates all indexes. The record is identified by its unique record identifier.

If any key is changed, the record entry for the old value is removed from the index and the record is reindexed under the new value. If the new value of the key duplicates an existing key for the same index in another record and duplicates are not allowed for that index, the record is not modified, the index is not changed, and status code 3118 (**Duplicate key**) is returned.

The data set must be open for modification. The record remains locked after modification.

If the data set is open in a transaction mode, ModifyISAMRecord can be called only from within a transaction.

The procedural interface is:

```
ModifyISAMRecord (ISAMHandle, uriRecord, pRecord,  
                  sRecord, pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

uriRecord is the unique record identifier of the record to be modified. uriRecord is usually the value returned by a previous Read operation.

pRecord and sRecord describe the memory area containing the record to be written. sRecord must be equal to the record size for the data set.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-24 details the ModifyISAMRecord Request Block.

ModifyISAMRecordByKey Procedure

The ModifyISAMRecordByKey service modifies a data set record and updates all indexes. It uses a unique key to identify the record instead of a record identifier.

If any key is changed, the record is removed from the index under the old value of the key and reindexed under the new value. If the new value of the key duplicates an existing key for the same index in another record, but duplicates are not allowed for that index, the record is not modified, the index is not changed, and status code 3118 (Duplicate key) is returned.

Table 7-24. ModifyISAMRecord Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	1
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	84
12	ISAMHandle	2	
14	uriRecord	4	
18	pRecord	4	
22	sRecord	2	
24	pStatusBlockRet	4	
28	sStatusBlock	2	4

The parameters for this operation include the index and the new record contents, but not a separate key for the record. The key used to identify the record is taken from the record itself.

For example, to change the salary of employee 150 in the Employee data set from \$34,000 to \$37,500, `iIndex` is specified as 0. (`empNo` is index 0, a unique key index.) ISAM extracts the `empNo` field, uses the `empNo` index to determine which record is to be modified and modifies the record.

This service cannot be used to change the key being used to identify the record.

The data set must be open for modification.

If the data set is open in a transaction mode, `ModifyISAMRecordByKey` can be called only from within a transaction. The record to be modified need not be locked; `ModifyISAMRecordByKey` locks the record before modifying it.

The procedural interface is:

```
ModifyISAMRecordByKey (ISAMHandle, iIndex, pRecord,  
                      sRecord, pUriRecordRet,  
                      pStatusBlockRet): ErcType
```

`ISAMHandle` is the ISAM handle that identifies the open data set.

`iIndex` identifies the index used. (The indexes are numbered from 0 in the order specified in `CreateISAM`.)

`pRecord` and `sRecord` describe the memory area containing the record to be modified. `sRecord` must be equal to the record size for the data set.

`pUriRecordRet` is the memory address of the 4-byte structure where the unique record identifier of the modified record is returned.

`pStatusBlockRet` is the memory address of the status block into which the status codes from the operation are returned.

Table 7-25 describes the `ModifyISAMRecordByKey` request block.

Table 7-25. ModifyISAMRecordByKey Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	4
2	nReqPbCb	1	1
3	nRespPbCb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	198
12	ISAMHandle	2	
14	iIndex	2	
16	pRecord	4	
20	sRecord	2	
22	pUriRecordRet	4	
26	sUriRecord	2	2
28	pStatusBlockRet	4	
32	sStatusBlock	2	4

NormalizeISAMStatus Procedure

The NormalizeISAMStatus procedure ensures that both the ercRet field of the request equals the erc field of the status block and the ercDetail field of the status block is valid.

An application must call NormalizeISAMStatus after asynchronous ISAM requests (issued by ISAMRequest) are completed. If not, the status block that the ISAM operation returns has undefined contents.

The status code that NormalizeISAMStatus returns diagnoses problems encountered while normalizing the status block. In general, the status code returned differs from the ercRet field in the request block.

For example, if an asynchronous Read operation returns status code 3127 (**no more records**) and a detail status of 0 (OK), NormalizeISAMStatus ensures that the ercRet field in the request block matches the status block erc field and returns 0 (OK).

NormalizeISAMStatus is an object module procedure.

The procedural interface is:

NormalizeISAMStatus (pRq): ErcType

pRq is the memory address of the request block.

OpenISAM Procedure

The OpenISAM procedure opens an existing data set and returns an ISAM handle for it. The ISAM handle is used to refer to the data set in subsequent operations. OpenISAM is an object module procedure.

The procedural interface is:

OpenISAM (pISAMHandleRet, pbDataSetName, cbDataSetName, pbPassword, cbPassword, mode, sRecord, pStatusBlockRet): ErcType

pISAMHandleRet is the memory address of the word into which the ISAM handle of the opened data set is returned.

pbDataSetName and cbDataSetName describe the memory area containing the character string representing the name of the data set.

pbPassword and cbPassword describe either the modify or read password for the opened data set, or a file system password that gives modify access to the data set files. If the mode is batch modify or transaction modify, you must supply the modify password. If the mode is batch read or transaction read, you can supply either password. If the mode is administrator, you must supply a file system password.

Mode specifies the mode in which the data set is opened as detailed in table 7-26.

In these ASCII constants, the first character is the high-order byte, and the second character is the low-order byte. This is the reverse of the byte order of strings in the B 20 programming languages.

For further information regarding modes, refer to section 2.

sRecord is the fixed-length record size for the data set.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-26. Data Set Modes

Value (ASCII)	Mode
ad	administrator
bm	batch modify
br	batch read
tm	transaction modify
tr	transaction read

QueryTransactionParams Procedure

The QueryTransactionParams procedure retrieves the application's current transaction parameters. (For more information, refer to Transaction Parameters Block, in this section.)

QueryTransactionParams is an object module procedure.

The procedural interface is:

```
QueryTransactionParams (pParamsRet, sParamsMax,  
                        pSPParamsRet, pStatusBlockRet):  
                        ErcType
```

pParamsRet and sParamsMax describe the memory area into which the transaction parameters are stored. The Transaction Parameters Block has the format shown in table 7-13.

pSPParamsRet is the memory address of a word into which the number of bytes retrieved are stored. The value stored in this word cannot exceed sParamsMax.

pStatusBlockRet is the status block memory address into which the status codes from the operation are returned.

ReadISAMRecordByUri and ReadISAMRecordByUriHold Procedures

The ReadISAMRecordByUri service reads a record identified by its unique record identifier. The ReadISAMRecordByUriHold service is identical to ReadISAMRecordByUri, except ReadISAMRecordByUriHold also locks the record when it is read.

If the data set is open in a transaction mode, then ReadISAMRecordByUriHold can be called only from within a transaction.

The procedural interface is:

```

ReadISAMRecordByUri and
ReadISAMRecordByUriHold (ISAMHandle, uriRecord,
                          pRecordRet, sRecord,
                          pStatusBlockRet): ErcType

```

ISAMHandle is the ISAM handle that identifies the open data set.

uriRecord is the unique record identifier of the record to be read. uriRecord is usually the value that a previous Read operation returns.

pRecordRet and sRecord describe the memory area into which the record is read. sRecord must be equal to the record size for the data set.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-27 describes the ReadISAMRecordByUri and ReadISAMRecordByUriHold request block.

Table 7-27. ReadISAMRecordByUri and ReadISAMRecordByUriHold Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	0
3	nRespPbCb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	*
12	ISAMHandle	2	
14	uriRecord	4	
18	pRecordRet	4	
22	sRecord	2	
24	pStatusBlockRet	4	
28	sStatusBlock	2	4

*ReadISAMRecordByUri = 86; ReadISAMRecordByUriHold = 131.

ReadNextISAMRecord and ReadNextISAMRecordHold Procedures

The ReadNextISAMRecord service reads the next record in key order from a data set. The unique record identifier of the record is returned. The ReadNextISAMRecordHold service is identical to ReadNextISAMRecord, except it also locks the record when it is read.

If the data set is open in a transaction mode, ReadNextISAMRecordHold can be called only from within a transaction.

The procedural interface is:

```
ReadNextISAMRecord and
ReadNextISAMRecordHold (ISAMHandle, pRecordRet,
                        sRecord, pUriRecordRet,
                        pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

pRecordRet and sRecord describe the memory area into which the record is read. sRecord must be equal to the record size for the data set.

pUriRecordRet is the memory address of the 4-byte structure into which the unique record identifier of the record is returned.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-28 describes the ReadNextISAMRecord and ReadNextISAMRecordHold request block.

ReadUniqueISAMRecord and ReadUniqueISAMRecordHold Procedures

The ReadUniqueISAMRecord service reads a record uniquely identified by a given key from a data set. Duplicates are not allowed for the index used, so that the key appears at most in one record of the data set.

The ReadUniqueISAMRecordHold service is identical to ReadUniqueISAMRecord, except ReadUniqueISAMRecordHold also locks the record. If the data set is open in a transaction mode, ReadUniqueISAMRecordHold can be called only from within a transaction.

Table 7-28. ReadNextISAMRecord and ReadNextISAMRecordHold Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	2
2	nReqPbCb	1	0
3	nRespPbCb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	*
12	ISAMHandle	2	
14	pRecordRet	4	
18	sRecord	2	
20	pUriRecordRet	4	
24	sUriRecord	2	4
26	pStatusBlockRet	4	
30	sStatusBlock	2	4

*ReadNextISAMRecord = 87; ReadNextISAMRecordHold = 132.

The procedural interface is:

```

ReadUniqueISAMRecord and
ReadUniqueISAMRecordHold (ISAMHandle, iIndex, pKey,
                           sKey, pRecordRet, sRecord,
                           pUriRecordRet,
                           pStatusBlockRet): ErcType
    
```

ISAMHandle is the ISAM handle that identifies the open data set.

iIndex identifies the index used. (The indexes are numbered from 0 in the order specified in CreateISAM.)

pKey and sKey describe the memory area containing the unique key that identifies the record to be read. sKey must be the correct length, in bytes, of keys for the index, iIndex.

pRecordRet and sRecord describe the memory area into which the record is read. sRecord must be equal to the record size for the data set.

pUriRecordRet is the memory address of the 4-byte structure into which the unique record identifier of the record is returned.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-29 details the ReadUniqueISAMRecord and ReadUniqueISAMRecordHold request block.

ReleaseISAMDataSet Procedure

The ReleaseISAMDataSet service releases a locked data set without ending the current transaction.

NOTE

Use this operation with care, since other applications can hold and even modify the released data set before the transaction is ended.

Table 7-29. ReadUniqueISAMRecord and ReadUniqueISAMRecordHold Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	4
2	nReqPbCb	1	1
3	nRespPbCb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	*
12	ISAMHandle	2	
14	iIndex	2	
16	pKey	4	
20	sKey	2	
22	pRecordRet	4	
26	sRecord	2	
28	pUriRecordRet	4	
32	sUriRecord	2	4
34	pStatusBlockRet	4	
38	sStatusBlock	2	4

*ReadUniqueISAMRecord = 88; ReadUniqueISAMRecordHold = 133.

If the data set is open in a transaction mode, ReleaseISAMDataSet can be called only from within a transaction.

The procedural interface is:

```
ReleaseISAMDataSet (ISAMHandle, pStatusBlockRet):
                    ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-30 describes the ReleaseISAMDataSet request block.

ReleaseISAMRecord Procedure

The ReleaseISAMRecord service releases a locked record without ending the current transaction.

NOTE

Other applications can hold and even modify the released record before the transaction ends.

Table 7-30. ReleaseISAMDataSet Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	2
2	nReqPbCb	1	0
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	206
12	ISAMHandle	2	
14	pStatusBlockRet	4	
18	sStatusBlock	2	4

If the data set is open in a transaction mode, ReleaseISAMRecord can be called only from within a transaction.

The procedural interface is:

```
ReleaseISAMRecord (ISAMHandle, uriRecord,
                  pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

uriRecord is the unique record identifier for the record to be released. uriRecord is usually the value returned by a previous Read operation.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-31 describes the ReleaseISAMRecord request block.

RenameISAM Procedure

The RenameISAM service changes the name of a data set by changing the name of both the data store and index files. It is invalid to rename only one of the two files.

RenameISAM uses two invocations of the BTOS RenameFile operation; one to change the data store file name and one to change the index file name.

Table 7-31. ReleaseISAMRecord Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	0
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	134
12	ISAMHandle	2	
14	uriRecord	4	
18	pStatusBlockRet	4	
22	sStatusBlock	2	4

There are certain RenameFile operations that are invalid, such as renaming a file from one volume to another, or renaming a file using an incorrect password. If a RenameISAM operation is attempted where one of the two required BTOS RenameFile operations is invalid, ISAM detects the error and renames the data set by using a valid name for both the data store and index file. One or both of the files, in this case, can retain the original name.

You must open the data set in administrator mode.

The procedural interface is:

```
RenameISAM (ISAMHandle, pbFileSpecDataStoreFile,
            cbFileSpecDataStoreFile,
            pbPasswordDataStoreFileRename,
            cbPasswordDataStoreFileRename,
            pbFileSpecIndexFile, cbFileSpecIndexFile,
            pbPasswordIndexFileRename,
            cbPasswordIndexFileRename, pStatusBlockRet):
            ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

pbFileSpecDataStoreFile and cbFileSpecDataStoreFile describe the file specification for the new name of the data store file for the data set.

pbPasswordDataStoreFileRename and cbPasswordDataStoreFileRename describe the password used when renaming the data store file for the data set.

pbFileSpecIndexFile and cbFileSpecIndexFile describe a file specification for the new index file name for the data set. If cbFileSpecIndexFile is 0, the file specification for the index is derived from the file specification for the data store file. ISAM copies the file specification defined by pbPasswordDataStoreFile and cbPasswordDataStoreFile. Then ISAM replaces the suffix (beginning with the period character) with the characters **.Ind**.

For example, if the file specification for the data store file is [vol]<dir>DataSet.Isam, the file specification for the index file is [vol]<dir>DataSet.Ind.

pbPasswordIndexFileRename and cbPasswordIndexFileRename describe the password used when renaming the data set index file. If the index file name is null, these two parameters are ignored and the password specified for the data store file is also used to create the index file.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-32 details the RenameISAM request block.

RollBackTransaction Procedure

The RollBackTransaction procedure signifies the unsuccessful end of a transaction and unlocks all records and data sets locked by the application.

Although you can call RollBackTransaction, it is effective only when the calling application is in a transaction.

RollBackTransaction is an object module procedure.

The procedural interface is:

```
RollBackTransaction (pStatusBlockRet): ErcType
```

Table 7-32. RenameISAM Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	2
2	nReqPbCb	1	4
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	89
12	ISAMHandle	2	
14	pbFileSpecDataStoreFile	4	
18	cbFileSpecDataStoreFile	2	
20	pbPasswordDataStore- FileRename	4	
24	cbPasswordDataStore- FileRename	2	
26	pbFileSpecIndexFile	4	
30	cbFileSpecIndexFile	2	
32	pbPasswordIndexFile- Rename	4	
36	cbPasswordIndexFile- Rename	2	
38	pStatusBlockRet	4	
42	sStatusBlock	2	4

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

SetISAMProtection Procedure

The SetISAMProtection service changes the passwords that allow an application to gain access to an open data set.

SetISAMProtection does not change the file system passwords for the data set files. You use the Executive **SET PROTECTION** command to change these passwords. You must open the data set in administrator mode.

The procedural interface is:

```
SetISAMProtection (ISAMHandle, pbPasswordOpenRead,
                  cbPasswordOpenRead,
                  pbPasswordOpenModify,
                  cbPasswordOpenModify,
                  pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

pbPasswordOpenRead and cbPasswordOpenRead describe the password that opens the data set for reading in either batch read or transaction read modes. The password cannot be longer than 12 bytes.

pbPasswordOpenModify and cbPasswordOpenModify describe the password that opens the data set for modification in either batch modify or transaction modify modes. This password can also be used to open the data set for reading in batch read or transaction read modes. The password cannot be longer than 12 bytes.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

Table 7-33 describes the SetISAMProtection request block.

SetTransactionParams Procedure

The SetTransactionParams procedure sets the application's current transaction parameters. (For more information, refer to Transaction Parameters Block, in this section.) SetTransactionParams is an object module procedure.

Table 7-33. SetISAMProtection Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	2
2	nReqPbCb	1	2
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	90
12	ISAMHandle	2	
14	pbPasswordOpenRead	4	
18	cbPasswordOpenRead	2	
20	pbPasswordOpenModify	4	
24	cbPasswordOpenModify	2	
26	pStatusBlockRet	4	
30	sStatusBlock	2	4

The procedural interface is:

```
SetTransactionParams (pParams, sParams,
                    pStatusBlockRet): ErcType
```

pParams and sParams describe the memory area containing the transaction parameters to be set. The Transaction Parameters Block has the format shown in table 7-13. If sParams is less than the length of the Transaction Parameters Block:

1. The supplied block must not include a partial field at the end.
2. Only parameters included in the block are changed.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

SetUpISAMIterationLimits Procedure

The SetUpISAMIterationLimits service initializes a sequence of Read operations for records that have keys for a specified index within a given range. Subsequent calls to GetISAMRecords(Hold) and ReadNextISAMRecord(Hold) operations read each record that has a key value within the range. Records are read in key value order.

If the index is not defined to include null values (binary 0's), records with null keys are not read.

The procedural interface is:

```
SetupISAMIterationLimits (ISAMHandle, iIndex,  
                           matchKind, pKey1, sKey1,  
                           pKey2, sKey2,  
                           pStatusBlockRet): ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

iIndex identifies the index used. (The indexes are numbered from 0 in the order specified in CreateISAM.)

matchKind specifies which records are retrieved according to the following:

- 0 all records, regardless of their key values (key1 and key2 are both ignored; sKey1 and sKey2 should be 0.)
- 1 all records containing a key less than key1 (key2 is ignored, and sKey2 should be 0.)
- 2 all records containing a key less than or equal to key1 (key2 is ignored, and sKey2 should be 0.)
- 3 all records containing a key equal to key1. (key2 is ignored, and sKey2 should be 0.)
- 4 all records containing a key greater than or equal to key1. (key2 is ignored, and sKey2 should be 0.)
- 5 all records containing a key greater than key1 (key2 is ignored, and sKey2 should be 0.)
- 6 all records containing a key greater than key1, but less than key2
- 7 all records containing a key greater than or equal to key1, but less than key2
- 8 all records containing a key greater than or equal to key1, but less than or equal to key2
- 9 all records containing a key greater than key1, but less than or equal to key2

pKey1 and sKey1 describe the memory area containing a key record. The key affects the set of records that subsequent read operations return (see matchKind for further information).

pKey2 and sKey2 describe the memory area containing a key record. The key affects the set of records that subsequent read operations return (see matchKind for further information).

pStatusBlockRet is the status block memory address into which the status codes from the operation are returned.

Table 7-34 describes the SetUpISAMIterationLimits request block.

SetUpISAMIterationPrefix Procedure

The SetUpISAMIterationPrefix service initializes a sequence of Read operations for records with keys for a specified index having a given prefix. Subsequent calls to GetISAMRecords(Hold) and ReadNextISAMRecord(Hold) operations

Table 7-34. SetUpISAMIterationLimits Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	2
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	210
12	ISAMHandle	2	
14	iIndex	2	
16	matchKind	1	
17	reserved	1	0
18	pKey1	4	
22	sKey1	2	
24	pKey2	4	
28	sKey2	2	
30	pStatusBlockRet	4	
34	sStatusBlock	2	4

read each record for which the given key is a prefix of the key stored in the record.

The index must contain either byte string or character string keys.

If the index is not defined to include null values (binary 0's), records with null keys are not read.

The procedural interface is:

```

    SetUpISAMIterationPrefix (ISAMHandle, iIndex, pKey,
                              sKey, pStatusBlockRet):
                              ErcType
  
```

ISAMHandle is the ISAM handle that identifies the open data set.

iIndex identifies the index used. (The indexes are numbered from 0 in the order specified in CreateISAM.)

pKey and sKey describe the memory area containing the key. sKey must be no larger than the length of the keys for index, iIndex.

pStatusBlockRet is the status block memory address into which the status codes from the operation are returned.

Table 7-35 details the SetUpISAMIterationPrefix request block.

Table 7-35. SetUpISAMIterationPrefix Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	4
2	nReqPbCb	1	1
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	92
12	ISAMHandle	2	
14	iIndex	2	
16	pKey	4	
20	sKey	2	
22	pStatusBlockRet	4	
26	sStatusBlock	2	4

StoreISAMRecord Procedure

The StoreISAMRecord service creates a new record in a data set. If necessary, the length of the data store file for the data set is increased to accommodate the new record.

The indexes are updated as required to reflect the presence of the new record. If the value of the new key duplicates an existing key in the same index, and duplicates are not allowed for that index, the record is not stored and the status code 3118 (**duplicate key**) is returned.

The data set must be open for modification.

If the data set is open in transaction mode, StoreISAMRecord can be called only from within a transaction. The created record is locked after StoreISAMRecord is called.

The procedural interface is:

```
StoreISAMRecord (ISAMHandle, pRecord, sRecord,  
                pUriRecordRet, pStatusBlockRet):  
    ErcType
```

ISAMHandle is the ISAM handle that identifies the open data set.

pRecord and sRecord describe the memory area containing the record to be written. sRecord must be equal to the record size for the data set.

pUriRecordRet is the memory address of the 4-byte structure where the unique record identifier of the record to be stored is returned.

pStatusBlockRet is the memory address of the ISAM status block in which the status codes from the operation are returned.

Table 7-36 describes the StoreISAMRecord request block.

Table 7-36. StoreISAMRecord Request Block

Offset	Parameter	Size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	2
3	nRespPbCb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqCode	2	94
12	ISAMHandle	2	
14	pRecord	4	
18	sRecord	2	
20	pUriRecordRet	4	
24	sUriRecord	2	4
26	pStatusBlockRet	4	
30	sStatusBlock	2	4

VerifyMultiUserISAM Procedure

The VerifyMultiUserISAM procedure sends a request to ISAM at either the B-NET node where the application is running or at another B-NET node. If ISAM is installed, a status code of 0 (OK) is returned. Any other status code indicates that multiuser ISAM is not available.

You can use the reserved symbols {Local} and {Master} whether or not B-Net is installed.

VerifyMultiUserISAM is an object module procedure.

The procedural interface is:

```
VerifyMultiUserISAM (pbNode, cbNode, pStatusBlockRet):
                    ercType
```

pbNode and cbNode describe the memory area containing the name of the B-NET node. The default node is the master workstation for the cluster or the standalone workstation, where the application is running.

pStatusBlockRet is the memory address of the status block into which the status codes from the operation are returned.

APPENDIX A

STATUS CODES

Decimal Value	Hexadecimal Value	Meaning
3100	0C1C	No such index exists. An operation that includes a key was invoked, but the specified index does not exist.
3101	0C1D	Prefix is not valid. SetUpISAMIterationPrefix was invoked for an index that is neither a byte nor character string.
3102	0C1E	Bad key length. The key length is inconsistent with the index type.
3103	0C1F	Bad ISAM or data base handle. The ISAM handle does not identify an open ISAM data set.
3104	0C20	Bad ISAM header size. The ISAM data set cannot be opened by the OpenISAM operation due to an inconsistency in the header of one of the files of the data set.
3105	0C21	Bad ISAM header. See 3104
3106	0C22	Internal error.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3107	0C23	ISAM is already installed. This code is generated by the InstallISAM operation or the ISAM INSTALL command if ISAM is already installed.
3108	0C24	Internal error.
3109	0C25	Internal error.
3110	0C26	Internal error.
3111	0C27	No free ISAM handles. The ISAM data set cannot be opened because the maximum ISAM data sets that can be simultaneously opened by all users combined (256) has been reached.
3112	0C28	Internal error.
3113	0C29	Buffers are too large The amount of RAM required by either the data store or index cache buffers exceeds a megabyte.
3114	0C2A	Internal error.
3115	0C2B	ISAM terminated abnormally. Following detection of an internal error, all subsequent ISAM operations and services generate this status code.
3116	0C2C	Internal error.
3117	0C2D	Bad unique record identifier. An incorrect unique record identifier parameter was passed to ISAM.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3118	0C2E	Duplicate key. An attempt to store or modify a record would duplicate the key stored in another record for the same index, but the index does not allow duplicates.
3119	0C2F	Index file error. This status code is returned as the status code of an ISAM operation. The detail status code refers to a problem with the index file of the ISAM data set.
3120	0C30	Attempted privacy breach. An attempt was made to modify a data set that is open in batch read or transaction read mode.
3121	0C31	Bad request. Either the request block is invalid, or the parameters of an operation are inconsistent or have invalid values.
3122	0C32	Data store file error. This is returned as the status code of an ISAM operation. The detail status code refers to a problem with the data store file of the ISAM data set.
3123	0C33	Index to data error. An inconsistency has arisen between the index and data store files of the ISAM data set.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa- decimal Value	Meaning
3124	0C34	Record size incorrect. The specified record size is incorrect for the ISAM data set.
3125	0C35	Duplicates allowed. An operation specified a unique key parameter (that is, duplicates are not allowed), but the index allows duplicates.
3126	0C36	No such record exists. A unique key was used to identify a record, but no record is stored in the ISAM data set with that key.
3127	0C37	No more records. A ReadNextISAMRecord(Hold) or GetISAMRecords(Hold) operation has read all the records within the range specified for the current iteration. No record is read when this status code is generated.
3128	0C38	Bad key. Either (1) a key field is longer than 64 bytes or is defined to occupy bytes past the end of the record, or (2) a supplied key is invalid. For example, a DECIMAL key with a digit that is not between 0 and 9 is invalid.
3129	0C39	Bad index. The specified key field does not exist, that is, the Index parameter of an ISAM operation is greater than or equal to the number of indexes in the ISAM data set.

ISAM STATUS CODES (Cont)

Decimal Value	Hexadecimal Value	Meaning
3130	0C3A	Bad ISAM mode. The mode parameter of the OpenISAM operation is invalid.
3131	0C3B	Cannot open ISAM. This is returned as the status code of an OpenISAM operation. The detail status code gives the reason for this failure.
3132	0C3C	Bad ISAM password. Either the password does not give the access desired by the OpenISAM operation, or the password is larger than the 12 bytes accepted by the SetISAMProtection operation.
3133	0C3D	Wrong record size. The OpenISAM operation detects the wrong size record.
3134	0C3E	Incompatible ISAM mode. An attempt was made to open a data set when the data set is already open in an incompatible mode.
3135	0C3F	Internal error.
3136	0C40	Not administrator. An operation for which the data set must be open in administrator mode is attempted with the data set open in some other mode.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3137	0C41	Cannot create ISAM. This is returned as the status code of the CreateISAM operation. The detail status code gives the reason for the failure.
3138	0C42	ISAM buffer is too small. The data set being opened or created requires buffers larger than those installed.
3139	0C43	Internal error.
3140	0C44	Small ISAM record. An attempt was made to create an ISAM data set with records shorter than four bytes.
3141	0C45	Not in transaction. An operation that can be invoked only when the application is in a transaction was invoked while the application was not in a transaction.
3142	0C46	Data set or relation is not available. An attempt to read or hold a record or to hold a data set failed because the data set was held by another user.
3143	0C47	Record is not available. An attempt to read or hold a record failed because the record was held by another user.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa decimal Value	Meaning
3144	0C48	Record is not locked. An operation for which the record (or its data set) must be held was invoked when neither the record nor its data set was held.
3145	0C49	Too many records are locked. An attempt was made to hold a record when the maximum allowable number of records was already held.
3146	0C4A	In transaction. BeginTransaction was invoked during a transaction.
3147	0C4B	Internal error.
3148	0C4C	Transaction purged. The application finished a transaction while the request was queued. (This status code is generated only if the application has a request pending when invoking RollBackTransaction or CommitTransaction.)
3149	0C4D	Internal error.
3150	0C4E	Internal error.
3151	0C4F	Data set is not locked. An operation for which the data set must be held was called when the data set was not held.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3152	0C50	ISAM heap is full. The operation failed because there is not enough room in the ISAM server's heap to store data structures required by the operation.
3153	0C51	Internal error.
3154	0C52	Files are not on same workstation. An attempt was made to either create or rename a data set so that the data store and index files would be on different workstations or to open a data set (in other than administrator mode) that is on a different workstation from the one where the ISAM server is running.
3155	0C53	Bad match kind. A <code>SetUpISAMIterationLimits</code> operation contained an invalid <code>matchKind</code> argument.
3156	0C54	Internal error.
3157	0C55	Transaction barrier after modification. An ISAM transaction barrier operation was called after a modification to the data set or data base in the current transaction, but the <code>SetTransactionParams</code> operation was previously invoked with <code>fBarrierAfterModify</code> set to <code>FALSE</code> (0).

ISAM STATUS CODES (Cont)

Decimal Value	Hexadecimal Value	Meaning
3158	0C56	Key is locked. This code is returned as the detail status code when the primary status code is 3143 (Record not available). It indicates that the record was not available because of key locking constraints.
3159	0C57	Record is held. This code is returned as the detail status code when the primary status code is 3143 (Record not available). It indicates that the record was not available because another user had locked it.
3160	0C58	Transaction has been rolled back. The applications current transaction has been rolled back. This status code is returned for every operation by an application after a lock has been broken, until the application invokes the RollbackTransaction operation.
3161	0C59	Journal file error. An error has occurred during access to a journal file. See the detail error for more information.
3162	0C5A	Journal open error. An error has occurred while opening the journal file. See the detail error for more information.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3163	0C5B	Journal close error. An error has occurred while closing the journal file. See the detail error for more information.
3164	0C5C	Journal write error. An error has occurred while writing to the journal file. See the detail error for more information.
3165	0C5D	Internal error. Journal record illegal. An error has occurred while reading the journal file during an attempt to roll back a transaction. A record in the journal file has an incorrect format.
3166- 3199	0C5E- 0C7F	Internal errors.
3200	0C80	Bad key type. The type field of a key specification for Sort/Merge or ISAM is invalid.
3201	0C81	Incorrect key length. The cbKey field of a key specification for a Sort/Merge or ISAM operation does not correspond to the type field of the key specification. (For example, for binary keys, cbKey must be 2.)

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3202	0C82	Bad key. A key contained in a record for Sort/Merge or ISAM, or a key described by a parameter of an ISAM operation is not of the correct type. (For example, each digit of a BCD key must be between 0 and 9.)
3203- 3299	0C83- 0CE3	Internal errors.
3300	0CE4	Not a STAM file. The operation failed because the file did not contain the proper signature.
3301	0CE5	STAM header bad checksum. The operation failed because the checksum computed on the file header did not match the checksum computed when the file was created.
3302	0CE6	Record does not exist. The operation failed because the specified record does not exist.
3303	0CE7	Malformed record. The operation failed because data read from the disk contained an inconsistency in the record header and trailer.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-Decimal Value	Meaning
3304	0CE8	Not fixed-length record. The operation failed because the access method cannot reference variable-length records.
3305	0CE9	Bad file type. The operation failed because the file cannot be accessed with the specified access method.
3306	0CEA	Bad buffer size. The operation failed because the buffer size was too small or not a multiple of 512.
3307	0CEB	Buffer is not word-aligned. The operation failed because the buffer was not word-aligned.
3308-3399	0CEC-0D47	Internal errors.
3900	0F3C	Cannot auto-restart. Because of a problem with the journal file, auto-restart recovery cannot be used to recover a data base (or one or more ISAM data sets). Use DB Recover to recover the data base. Recover ISAM data sets by another mechanism such as restoring archived copies.
3901	0F3D	Bad ISAM configuration file An attempt was made to use a file as an ISAM configuration file, but the file has been damaged or it was not generated by ISAM Configure.

ISAM STATUS CODES (Cont)

Decimal Value	Hexa-decimal Value	Meaning
3902	0F3E	Bad ISAM configuration version An attempt was made to use an ISAM configuration file that was generated by an incompatible version of ISAM Configure. Regenerate the file by using a compatible version.
3903- 3999	0F3F- 0F9F	Internal errors.

APPENDIX B

UPWARD COMPATIBILITY SUPPORT

ISAM 5.0 supports the use of applications written for previous releases of ISAM, even though ISAM's set of operations do not include these procedures.

SUPERSEDED PROCEDURES

The following eight operations are no longer part of the standard set of ISAM operations:

- EndISAMTransaction
- LockISAM
- PurgeISAMTransaction
- SetupISAMIteration
- SetupISAMIterationKey
- SetupISAMIterationRange
- StartISAMTransaction
- UnlockISAM

ISAM supports all eight operations only as object modules (no longer request blocks). ISAM 5.0 handles each of these operations as described in this appendix. Refer to section 7, for descriptions of the current ISAM operations that are mentioned here.

EndISAMTransaction

EndISAMTransaction is equivalent to CommitTransaction.

LockISAM

You implement LockISAM by BeginTransaction followed by a series of HoldISAMDataSet operations.

PurgeISAMTransaction

PurgeISAMTransaction is equivalent to RollBackTransaction.

SetupISAMIteration SetupISAMIterationKey SetupISAMIterationRange

You implement these operations by equivalent calls to SetupISAMIterationLimits.

StartISAMTransaction

You implement StartISAMTransaction by BeginTransaction followed by a series of HoldISAMDataSet operations.

UnlockISAM

If the application uses UnlockISAM in a transaction, this operation is equivalent to CommitTransaction; otherwise, UnlockISAM has no effect.

InstallISAM SUPPORT

Previous releases of ISAM used two libraries: ISAMSingle.Lib for single-user applications, and ISAMMulti.Lib for multiuser application. ISAMSingle.Lib included the ISAM procedural interfaces and all the implementing modules. The linked run file included all of ISAM. ISAMMulti.Lib included the ISAM procedural interfaces as request interfaces. The system sent requests to the multiuser ISAM service installed at the master workstation or a standalone workstation.

The system provided InstallISAM in previous ISAM releases to initialize single-user ISAM. The ISAMMulti.Lib version enabled compatibility; if allowed, you link an application as either a multiuser or single-user. The multiuser version determined whether or not the multiuser server was installed and returned an error status if it was not installed.

In ISAM 5.0, there is only one library: ISAM.Lib. Single-user applications now call LoadSingleUserISAM to load the ISAM service as a task. Multiuser applications can call VerifyMultiuserISAM to determine whether or not the system installed the multiuser ISAM service.

To provide compatibility for applications that use InstallISAM, there are two versions of InstallISAM in ISAM.Lib. The module named IsaMin is the multiuser version; it calls VerifyMultiuserISAM. The single-user version, the module IsaSin, calls LoadSingleUserISAM. When you link an application that uses InstallISAM, you must specify the appropriate version of InstallISAM in the object modules line of the Link command form. For single-user applications, specify **ISAM.Lib(IsaSin)** when you link the run file.

For multiuser applications, specify **ISAM.Lib(IsaMin)** when you link the run file.

B-TREE NODE SIZES

In ISAM 5.0, the memory requirements for the ISAM server are significantly reduced without degrading performance. In releases prior to ISAM 4.0, the default B-tree node size was six sectors in **ISAM CREATE**, **ISAM REORGANIZE**, and the index buffers in **ISAM INSTALL**. The default B-tree node size in ISAM 5.0 is two sectors.

For existing data sets where you did not set the B-tree node sizes to one or two sectors, you must do one of the following:

- use **ISAM REORGANIZE** to rebuild the index file with 2-sector B-tree nodes or
- reconfigure ISAM with the **ISAM CONFIGURE** command to use larger index buffers (six sectors).

APPENDIX C

ESTIMATED INDEX FILE SIZES

B-tree nodes can fluctuate between 50 percent and 100 percent full. Adding a key to a full node causes the node to split into two 50 percent full nodes. A key is also added to the B-tree node above the full node that split. When the full node is the root node, the current root node splits into two 50 percent full nodes, and ISAM creates a new root node with two nodes below it. This introduces a new level in the B-tree.

Removing a key from a 50 percent full node causes absorption of the node by its neighbors. ISAM uses either one or two nodes to eliminate the node that is now less than 50 percent full. The current size of the neighbor nodes determines the distribution of the keys. The result is that one or two nodes become at least two-thirds full.

When ISAM eliminates a node, it deletes a key from the node one level up in the B-tree. If eliminating a node causes the root node to have only one node below it at the next level, ISAM deletes the root node. The next lowest level becomes the root node, and the B-tree has one less level.

INDEX FILE SIZES

You can only compute the exact size of an index file immediately after ISAM rebuilds the indexes. After a series of updates (additions, modifications, deletions), you can only estimate the size.

You base the estimate on:

- the number of records in the data set, n
- the definition of the indexes
- the average load factor, f , which is the fraction of each B-tree node that is in use

Whenever ISAM rebuilds the indexes for a relation, the loading factor for the B-tree nodes is 80 percent full.

As ISAM stores, modifies, or deletes records, the portion of the node that fills varies between 50 percent and 100 percent. The node never falls below 50 percent full, and it is likely to remain near 80 percent full most of the time.

Example and Calculations

This example uses a relation with:

- 50,000 records
- a 10-byte key
- a 4-byte key
- 6-sector nodes

If the load factor, f , is 80 percent, then you calculate the index file size as follows:

1. the average number of keys per node:

$$\text{10-byte key: } b = 0.8 * \frac{6*512 - 16}{10 + 4} = 174.6$$

$$\text{4-byte key: } b = 0.8 * \frac{6*512 - 16}{4 + 4} = 305.6$$

2. the index sizes:

$$\begin{aligned} \text{10-byte key: } & 6*50000*(1/174.6 + 2/174.6^2) \\ & = 1738 \text{ sectors} \end{aligned}$$

$$\begin{aligned} \text{4-byte key: } & 6*50000*(1/305.6 + 2/305.6^2) \\ & = 989 \text{ sectors} \end{aligned}$$

$$\text{total size} = 2727 \text{ sectors}$$

Similar computations for $f = 50$ percent and $f = 100$ percent yield the following:

1. for 50 percent:

10-byte key: $b = 109$
index size = 2803 sectors

4-byte key: $b = 191$
index size = 1587 sectors

total size = 4390 sectors

2. for 100 percent:

10-byte key: $b = 218$
index size = 1389 sectors

4-byte key: $b = 382$
index size = 789 sectors

total size = 2178 sectors

APPENDIX D

GLOSSARY

Access Mode

An access mode is the method of opening a data set to read or modify records. The access mode affects the extent to which other application systems can share the data set.

Administrator Mode

Administrator mode is an access mode you use to perform data set-level activities such as deleting, renaming, and setting protection.

Application

An application is a task you invoke to access a data set for a particular program.

Asynchronous Request

An asynchronous request is a method of accessing ISAM system services directly so that data set access and internal computations overlap. Asynchronous requests allow applications to execute more efficiently and rapidly than a procedural interface method.

B-Tree

A B-tree is the type of structure used to contain ISAM indexes. A B-tree is usually pictured as an upside down tree, much like a family tree, with a "root" node at the top and "leaf" nodes below the root.

Configuration File

A configuration file specifies the sizes of the ISAM server's memory areas according to the number of users utilizing the server.

DAM

See Direct Access Method.

Data Buffer

A data buffer is an I/O buffer into which ISAM reads portions of data files.

Data Set

A data set contains one type of fixed-length records. ISAM accesses them through fixed-length keys.

Data Store File

A data store file is the physical file that holds the records of a data set.

Deadlock

Deadlock occurs when two or more transactions request records or data sets already locked by the other transaction.

Direct Access Method

The Direct Access Method (DAM) provides random access to disk file records identified by record number.

Exclusive Access

Exclusive access limits the accessibility of a data set or record to a single user. Compare with Shared Access.

File

A file is a set of related records (on a disk) treated as a unit.

Heap

A heap is an area of memory containing internal ISAM data structures.

Index

ISAM uses an index (structure) to locate particular records within a data set. Each key field of a data set defines one index. Also see Key and Record.

Index Buffer

An index buffer is an I/O buffer into which ISAM reads B-tree nodes. Also see B-Tree and Node.

Indexed Sequential Access Method (ISAM)

The Indexed Sequential Access Method provides random access to fixed-length records identified by multiple keys stored in disk files. Compare with Direct Access Method and Record Sequential Access Method.

Index File

An index file holds the indexes for all of the data set's keys.

ISAM Handle

An ISAM handle is a word value used in ISAM operations to identify an open data set.

Key

ISAM uses keys to access data set records. ISAM defines a key by its position in the record, its length, and type.

Key Type

Key types support the various character and numeric representations used by the B 20 programming languages and processors.

Locking

Locking is the process of obtaining Exclusive Access to a record or data set in multiuser ISAM.

Node

A node is a portion of a B-tree that stores index keys.

Password

A password is a text string used to validate an application or user's access to the data set.

Record

A record is a group of related data fields treated as a unit.

Record Sequential Access Method

The Record Sequential Access Method (RSAM) provides sequential read-only access to the records of a data set.

Reorganization

Reorganization is the process of freeing up space in a data set by removing deleted records and rebuilding the indexes for the data set.

RSAM

See Record Sequential Access Method.

Shared Access

Shared access enables multiple users to simultaneously access the same data set. Compare with Exclusive Access.

Status Block

A status block is a 4-byte memory area that ISAM uses to report status codes to an application.

Swap Zone

A swap zone is a virtual code segment management buffer.

Timeout

ISAM uses timeouts to prevent a deadlock. A timeout value specifies the maximum time a request to lock a data set or record can queue. Also see Deadlock.

Transaction

A transaction is a unit of work. Transactions permit shared access to a data set.

Unique Record Identifier (URI)

A Unique Record Identifier is a 4-byte unsigned integer used to identify a record in a data set.

Write-Through Cache

The write-through cache is a set of I/O buffers that ISAM uses to bring segments of disk records into memory as needed.

INDEX

- Access mode, D-1
- Administrator mode, 2-5, D-1
- Advantages of B-tree structure, 2-1
- Application, D-1
- Asynchronous requests, 7-13, 7-14, D-1
- Avoiding a deadlock, 2-10
- Batch mode, 2-5
- BeginTransaction
 - operation, 7-10
 - procedure, 7-14
- Binary key, 3-3
- BTOS
 - multipartition, 6-3
- Buffers
 - data, 6-5
 - index, 6-6
- Buffer size guidelines, 6-11
- Byte string key, 3-4
- B-tree node sizes, B-3
- B-tree structure, D-1
 - advantages of, 2-1
- Changing indexes and other ISAM CREATE parameters, 5-8
- Character string key, 3-4
- CloseISAM procedure, 7-15
- Commands (ISAM), 2-12
 - ISAM CONFIGURE, 6-6
 - ISAM COPY, 4-1, 4-2
 - ISAM CREATE, 4-1
 - ISAM DELETE, 4-1
 - ISAM INSTALL, 6-1, 6-3
 - ISAM RENAME, 4-1
 - ISAM SET PROTECTION, 4-1
 - ISAM STATUS, 4-1
 - ISAM TERMINATE, 4-2
- CommitTransaction
 - operation, 7-10
 - procedure, 7-15
- Composite keys, 3-2
- Concepts, 2-1
- Configuration file, D-1
- CreateISAM
 - procedure, 7-16
 - request block, 7-17
- Cursor movement in the ISAM CONFIGURE form, 6-7
- Data
 - buffers, 6-5, D-2

- Data (continued)
 - integrity, 2-13
 - security, 2-12
- Data set, D-2
 - access 7-8
 - index (sample), 3-2
 - loading a, 5-7
 - locking a, 2-7
 - management operations, 7-3
 - modes, 7-33
- Data set access modes, 2-5
 - description of, 2-5
 - multiuser access to, 2-7
- Data set records
 - sorting, 5-11
- Data store file, 2-1, D-2
- Deadlock, 2-10, D-2
 - avoiding a, 2-10
 - samples of, 2-11
- Decimal (odd)/decimal (even) key, 3-4
- Default index file name, 4-2
- Definitions of procedures, 7-14
- DeleteISAM
 - procedure, 7-18
 - request block, 7-18
- DeleteISAMRecord
 - procedure, 7-18
 - request block, 7-19
- DeleteISAMRecordByKey
 - operation, 7-8
 - procedure, 7-20
 - request block, 7-21
- DeleteISAMRecord operation, 7-8
- Deleting records, 2-4
- Description of data set access modes, 2-5
- Differences between multiuser and single-user access, 6-2
- Direct Access Method (DAM), D-2
- Disk requirements, 1-2
- Display key, 3-4
- Distributed ISAM, 2-4
- Dual floppy standalone systems
 - installing ISAM on, 1-3
- Error logging, 2-13
- Estimating index file sizes, C-1
- Exclusive access, D-2
- Features overview, 1-1
- File, D-2
- File types, 2-1

- Function
 - ISAM operations by, 7-2
- GetISAMRecords
 - operation, 7-9
 - procedure, 7-20
 - request block, 7-23
- GetISAMRecordsHold
 - operation, 7-9
 - procedure, 7-21
 - request block, 7-23
- Hard disk systems
 - installing ISAM on, 1-3
- Heap, 6-5, D-2
- HoldISAMDataSet
 - operation, 7-10
 - procedure, 7-24
 - request block, 7-25
- HoldISAMRecord
 - operation, 7-10
 - procedure, 7-26
 - request block, 7-26
- Index, 3-1 , D-2
 - buffers, 6-6, D-2
 - file, 2-1, D-2
 - keys, 3-1
- Indexed Sequential Access Method (ISAM), D-2
- Index file sizes
 - estimating, C-1
 - examples and calculations, C-2
- Installation
 - instructions, 1-3
 - multiuser, 6-2
- Installing ISAM, 2-10
 - on dual floppy standalone systems, 1-3
 - on hard disk systems, 1-3
 - on XE520 systems, 1-4
- InstallISAM support, B-2
- Integer key, 3-5
- Integrity of data, 2-13
- Internal consistency checking, 2-13
- ISAM
 - commands, 2-12, 4-1
 - configuration file, 2-12, 6-6
 - data sets, 3-1
 - handle, D-3
 - installation, 2-10
 - new features, 1-1
 - operations, 2-3

ISAM (continued)

- standard features, 1-1
- ISAM CONFIGURE command, 6-6
- ISAM CONFIGURE display, 6-7, 6-8, 6-9
- ISAM CONFIGURE form, 6-7
 - cursor movement in the, 6-7
- ISAM COPY command, 4-1, 4-2
- ISAM COPY form, 4-2
 - parameters, 4-3
- ISAM CREATE command, 4-1, 4-4
- ISAM CREATE form, 4-4
 - parameters, 4-5
- ISAM DELETE command, 4-1, 4-8
- ISAM DELETE form, 4-8
 - parameters, 4-8
- ISAM description block, 7-3, 7-4
- ISAM index specification block, 7-5
- ISAM INSTALL command, 6-1, 6-3
- ISAM INSTALL form, 6-3
 - parameters, 6-4
- ISAM key types and programming language representations, 3-5
- ISAM operations, 7-1
 - by function, 7-2
- ISAM RENAME command, 4-1, 4-8
- ISAM RENAME form, 4-9
 - parameters, 4-9
- ISAM REORGANIZE command, 2-13, 5-1
- ISAM REORGANIZE form, 5-2
 - parameters, 5-3
- ISAMRequest procedure, 7-27
- ISAM server installation, 6-1
 - memory allocation for, 6-3
- ISAM service access, 7-12, 7-13
- ISAM SET PROTECTION command, 4-1, 4-10
- ISAM SET PROTECTION form, 4-11
 - parameters, 4-11
- ISAM STATUS command, 4-1, 4-12
- ISAM STATUS form, 4-13
 - parameters, 4-13
- ISAM STATUS reports (samples), 4-14
- ISAM TERMINATE command, 4-2, 4-14
- ISAM utilities, 4-1
- Key component
 - type of, 7-7
- Key position, 2-2
- Keys, D-3
 - and indexes, 3-1
 - composite, 3-3

- Key types, 2-2, 3-3, D-3
 - description of, 3-3
- Loading a data set, 5-7
- LoadSingleUserISAM procedure, 7-27
- Locking, D-3
- Locking a record or data set, 2-7
- Locking operations, 7-10
 - HoldISAMDataSet, 7-10
 - HoldISAMRecord, 7-10
 - ReleaseISAMDataSet, 7-10
 - ReleaseISAMRecord, 7-10
- Long/short/extended/IEEE key, 3-5
- Long/short/real key, 3-5
- MAINTAIN FILE command, 2-2, 2-13
- Memory allocation calculation, 6-10
- Memory requirements, 1-2
 - for ISAM server installation, 6-3
- Memory usage, 7-13
- Modes
 - administrator, 2-5
 - batch, 2-5
 - transaction, 2-5
- Modifying records, 2-4
- ModifyISAMRecord
 - procedure, 7-28
 - request block, 7-29
- ModifyISAMRecordByKey
 - operation, 7-8
 - procedure, 7-29
 - request block, 7-31
- ModifyISAMRecord operation, 7-8
- Multipartition BTOS, 6-3
- Multiple record access (iteration) operations, 7-9
 - GetISAMRecords, 7-9
 - GetISAMRecordsHold, 7-9
 - ReadNextISAMRecord, 7-9
 - ReadNextISAMRecordHold, 7-9
 - SetUpISAMIterationLimits, 7-9
 - SetUpISAMIterationPrefix, 7-9
- Multiuser access to data set access modes, 2-7
- Multiuser installation, 6-2
- Multiuser ISAM
 - using either single-user or, 7-13
- New ISAM features, 1-1
- Node, D-3
- NormalizeISAMStatus procedure, 7-31
- OpenISAM procedure, 7-32
- Operations and transactions, 2-9

- Operations (ISAM), 2-3, 7-1
- Overview, 1-1
 - of features, 1-1
- Parameters
 - on ISAM COPY form, 4-3
 - on ISAM CREATE form, 4-5
 - on ISAM DELETE form, 4-8
 - on ISAM INSTALL form, 6-4
 - on ISAM RENAME form, 4-10
 - on ISAM REORGANIZE form, 5-3
 - on ISAM SET PROTECTION form, 4-11
 - on ISAM STATUS form, 4-13
- Password, D-3
- Procedures, 7-14
 - BeginTransaction, 7-14
 - CloseISAM, 7-15
 - CommitTransaction, 7-15
 - CreateISAM, 7-16
 - DeleteISAM, 7-18
 - DeleteISAMRecord, 7-18
 - DeleteISAMRecordByKey, 7-20
 - GetISAMRecords, 7-20
 - GetISAMRecordsHold, 7-21
 - HoldISAMDataSet, 7-24
 - HoldISAMRecord, 7-26
 - ISAMRequest, 7-27
 - LoadSingleUserISAM, 7-27
 - ModifyISAMRecord, 7-28
 - ModifyISAMRecordByKey, 7-29
 - NormalizeISAMStatus, 7-31
 - OpenISAM, 7-32
 - QueryTransactionParams, 7-33
 - ReadISAMRecordByUri, 7-33
 - ReadISAMRecordByUriHold, 7-33
 - ReadNextISAMRecord, 7-35
 - ReadNextISAMRecordHold, 7-35
 - ReadUniqueISAMRecord, 7-35
 - ReadUniqueISAMRecordHold, 7-35
 - ReleaseISAMDataSet, 7-37
 - ReleaseISAMRecord, 7-38
 - RenameISAM, 7-39
 - RollBackTransaction, 7-41
 - SetISAMProtection, 7-42
 - SetTransactionParams, 7-42
 - SetUpISAMIterationLimits, 7-43
 - SetUpISAMIterationPrefix, 7-45
 - StoreISAMRecord, 7-47
 - VerifyMultiUserISAM, 7-48

- QueryTransactionParams
 - operation, 7-10
 - procedure, 7-33
- Reading records, 2-4
- ReadISAMRecordByUri
 - operation, 7-9
 - procedure, 7-33
 - request block, 7-34
- ReadISAMRecordByUriHold
 - operation, 7-9
 - procedure, 7-33
 - request block, 7-34
- ReadNextISAMRecord
 - operation, 7-9
 - procedure, 7-35
 - request block, 7-36
- ReadNextISAMRecordHold
 - procedure, 6-35
 - request block, 7-36
- ReadUniqueISAMRecord
 - procedure, 7-35
 - request block, 7-37
- ReadUniqueISAMRecordHold
 - operation, 7-9
 - procedure, 7-35
 - request block, 7-37
- Record, D-3
- Record management and access, 7-8
- Record management operations, 7-8
 - DeleteISAMRecord, 7-8
 - DeleteISAMRecordByKey, 7-8
 - ModifyISAMRecord, 7-8
 - ModifyISAMRecordByKey, 7-8
 - StoreISAMRecord, 7-8
- Records
 - deleting, 2-4
 - locking, 2-7
 - modifying, 2-4
 - reading, 2-4
 - storing, 2-3
- Record Sequential Access Method (RSAM), D-3
- Related software, 1-3
- ReleaseISAMDataSet
 - operation, 7-10
 - procedure, 7-37
 - request block, 7-38
- ReleaseISAMRecord
 - operation, 7-10

- ReleaseISAMRecord
 - procedure, 7-38
 - request block, 7-39
- RenameISAM
 - procedure, 7-39
 - request block, 7-41
- Reorganization, D-4
- Requests
 - asynchronous, 7-13, 7-14
- Requirements
 - disk, 1-2
 - memory, 1-2
- Resident code and data requirements, 6-4
- RollBackTransaction
 - operation, 7-10
 - procedure, 7-41
- Security of data, 2-12
- SetISAMProtection
 - procedure, 7-42
 - request block, 7-43
- SetTransactionParams
 - operation, 7-10
 - procedure, 7-42
- SetUpISAMIterationLimits
 - operation, 7-9
 - procedure, 7-43
 - request block, 7-45
- SetUpISAMIterationPrefix
 - operation, 7-9
 - procedure, 7-45
 - request block, 7-45
- Shared access, D-4
- Single record access operations, 7-9
 - ReadISAMRecordByUri, 7-9
 - ReadISAMRecordByUriHold, 7-9
 - ReadUniqueISAMRecord, 7-9
 - ReadUniqueISAMRecordHold, 7-9
- Single-user ISAM
 - using either multiuser or, 7-13
- Software
 - related, 1-3
- Sorting data set records, 5-11
- Standard ISAM features, 1-1
- Status block, 7-1, D-4
 - format, 7-2
- Status codes, A-1
- StoreISAMRecord
 - operation, 7-8

- StoreISAMRecord (continued)
 - procedure, 7-47
 - request block, 7-48
- Storing records, 2-3
- Superseded procedures, B-1
 - EndISAMTransaction, B-1
 - LockISAM, B-1
 - PurgeISAMTransaction, B-2
 - SetupISAMIteration, B-2
 - SetupISAMIterationKey, B-2
 - SetupISAMIterationRange, B-2
 - StartISAMTransaction, B-2
 - UnlockISAM, B-2
- Swap zone, 6-5, D-4
- Timeout, D-4
- Transaction, D-4
- Transaction mode, 2-5
- Transaction operations, 7-10
 - BeginTransaction, 7-10
 - CommitTransaction, 7-10
 - QueryTransactionParams, 7-10
 - RollBackTransaction, 7-10
 - SetTransactionParams, 7-10
- Transaction parameters block, 7-11
 - format, 7-12
- Transaction related constraints, 7-11
- Transactions, 2-6
 - and operations, 2-9
- Type of key component, 7-7
- Types
 - of files, 2-1
 - of keys, 2-2, 3-3
- Unique record identifier (URI), 2-3, D-4
- Upward compatibility support, B-1
- Using either multiuser or single-user ISAM, 7-13
- Utilities (ISAM), 4-1
- VerifyMultiUserISAM procedure, 7-48
- Write-through cache, 2-13, D-4
- XE520 systems
 - installing ISAM on, 1-4

