StorHouse®

Concepts and Facilities Manual

StorHouse Release 5.6

Publication Number
900026 Rev. P
September 29, 2004

FileTek

No part of this publication may be reproduced, translated, stored in any electronic retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of FileTek, Inc.

This publication Copyright ©2004 FileTek, Inc. As an Unpublished Licensed Work. All Rights Reserved. Publication Number 900026 Rev. P

NOTICE: U.S. GOVERNMENT USERS
This notice applies to all acquisitions of this work by or for the U.S. Government ("Government"), or by any prime contractor or subcontractor (at any tier) under any contract, cooperative agreement or other activity with the Government. By accepting delivery of this work, the Government agrees that this work and the Licensed Program(s) described herein qualify as "commercial" computer software within the meaning of the acquisition regulation(s) applicable to this procurement. The terms of conditions of the license for the Licensed Program(s) shall pertain to the Government's use and disclosure of this work and the Licensed Program(s), and shall supersede any conflicting contractual terms or conditions. If the license for this work and the Licensed Program(s) fails to meet the Government's need or is inconsistent in any respect with Federal law, the Government agrees to return this work and the Licensed Program(s), unused, to FileTek, Inc. The following additional statement applies only to acquisitions governed by DFARS Subpart 227.4 (October 1988) "Restricted Rights – Use, duplication and disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (OCT. 1988)." Unpublished licensed work property of FileTek, Inc. Unauthorized use, duplication or distribution prohibited. All rights reserved. A copyright notice on this work and/or on the Licensed Program(s) by itself does not constitute publication or public disclosure of this work or the Licensed Program(s). The contractor/manufacturer is:

FileTek, Inc.
9400 Key West Avenue
Rockville, Maryland 20850

Information in this document is subject to change without notice and does not represent a commitment on the part of FileTek, Inc. Further, FileTek, Inc. reserves the right to supplement the document with information not available at the time of creation of the document. FILETEK, INC. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND CANNOT WARRANT THE RESULTS YOU MAY OBTAIN USING THE DOCUMENT. IN NO EVENT SHALL FILETEK, INC. BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, EVEN IF FILETEK, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES ARISING FROM ANY DEFECT OR ERROR IN THIS PUBLICATION. Some states or jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

FileTek and StorHouse are registered U.S. trademarks of FileTek, Inc. VRAM is a U.S. trademark of FileTek, Inc. All other brand or product names are trademarks or registered trademarks of their respective owners.

Documentation for FileTek's StorHouse product. Protected by the following U.S. Patents: 4,864,572; 5,247,660; 5,727,197; 6,049,804.

Contents

Welcome xiii

What is StorHouse? xiii
- StorHouse/SM xiii
- StorHouse/RM xiii
- StorHouse/Control Center xiv

Purpose of This Document xiv

Intended Audience xiv

Content xiv

Related Documentation xv

Notational Conventions xvi

Chapter 1: StorHouse Introduction 1-1

About StorHouse 1-1

StorHouse Software and Hardware 1-2

StorHouse Users, Accounts, and Sessions 1-3

StorHouse Interfaces 1-4
- User Interfaces 1-4
- Application Program Interface 1-5
- Basic StorHouse Interface 1-5
- Network Support Interface 1-5

Storage and File System 1-5

System Identification 1-6

System Parameters 1-6

System Logging 1-7
Administration Log 1-7
User Log 1-7

Chapter 2: Physical Storage 2-1

Device Identification 2-1
Storage and Device Levels 2-2
Unit Number 2-2
Subunit Type 2-3
Subunit Number 2-3

Physical Volume Identification 2-3
Volume Identification Code 2-4
Media Type and Recording Type 2-4
Volume Label 2-10
Side 2-11

Volume and Device Access 2-11
Magnetic Disk Drives 2-11
The Performance Buffer 2-12
Optical Disk Library Devices 2-12
Optical Disk Drives 2-13
Optical Disk Library Exchange Station 2-14
Automated Magnetic Tape Library Devices 2-15
Bar Code Readers 2-16
Magnetic Tape Drives 2-16
Magnetic Tape Library Exchange Station 2-17
Library Device Malfunction 2-17
Performance Enhancements for Library Devices 2-17
Improved Performance for Volume Operations 2-17
Ability to Minimize Volume Mounts 2-18
Support for ACSLS and LibraryStation Software 2-18

Chapter 3: Data Identification and Access Control 3-1

Files and File Access Groups 3-1
File Identifier and File Number 3-1
File Names 3-2
Access Group Names 3-2

File Versions 3-2
File Revisions 3-2

File Structures 3-3
File Extents 3-3

Frames 3-4
Records 3-4
Control Record Formats 3-4
Data Record Formats 3-5

File Organization 3-5
SEQUENTIAL Files 3-5
RECORD Files 3-6
KEYED Files 3-6
KEYSEQUENTIAL Files 3-7
STORHOUSE Files 3-7

File and Record Format Identification 3-7
Transportable Formats 3-8

File Extent Types 3-8
Relation of File Extents to Revisions 3-9
SEQUENTIAL Files 3-10
VRAM Files 3-10
STORHOUSE Files 3-11

File Directories 3-12
Primary Directory 3-13
Backup Directory 3-13
Archive Directory 3-13

Directory Information 3-13
Group Directory Information 3-13
File Directory Information 3-14
Version Directory Information 3-14
Extent Directory Information 3-16
Directory Functions 3-16

File and Record Access 3-17
Basic File Functions 3-17
Access Methods 3-17
Access Modes 3-18
Record Functions 3-18
File Transfer Functions 3-19
Software Disabled File Version 3-20

File Access Control 3-20
Access Group Passwords and Access Types 3-20
File Passwords and Access Types 3-21
File Locks 3-21

Chapter 4: Storage Allocation 4-1

Volumes 4-1
Volume Label Records 4-1
Volume Table of Contents 4-2
Volume Formats 4-2
Magnetic Disk 4-2
Optical Disk 4-2
Magnetic Tape 4-2
Residence 4-3
Volume HOLD Attribute 4-3
Deactivation Time and Deactivated Volumes 4-3
Cycle Time and Cycled Volumes 4-4
Expiration Time and Expired Volumes 4-4
Writelocked Volumes 4-4
Disabled Volumes 4-5
Memos for Volumes 4-5

Volume Sets 4-6
Volume Set Name 4-6
Volume Set Attributes 4-6
Size 4-6
LIMIT Attribute 4-6
Residence 4-7
Library Device and Media Attributes 4-7
Volume Set HOLD Attribute 4-7
Deactivation Time 4-8
Cycle Time 4-8
Expiration Time 4-8
Level F Volume Set 4-9
Memos for Volume Sets 4-9

File Sets 4-9
File Set Name 4-10
File Set Attributes 4-10
Size 4-11
LIMIT Attribute 4-11
Storage Allocation Attribute 4-11
Contiguous Allocation 4-11
Noncontiguous Allocation 4-11
Update Space 4-12
Staging Attribute 4-12
File Set Retention Attribute 4-12
Replication Class Name Attribute 4-13
Level F File Sets 4-13

File Storage and Statuses 4-14

File Residence 4-14
File Extent Statuses 4-14

File Size 4-15
SEQUENTIAL File 4-15
VRAM File Size 4-16
VRAM Data Extent 4-16
VRAM DF Extents 4-17
All Files 4-17
Checkpointed Files 4-17
Files with Keys 4-17
Keysequential Files 4-17
Updated Files 4-18
VRAM K Extents 4-18
VRAM Update Extents 4-19

Storage Allocation and Deallocation 4-19
Blank Volumes, Empty Volumes, and Free Pool Volume Sets 4-20
Allocation of Blank Volumes 4-20
Allocation of Empty Volumes 4-21
Placement of File Sets in Volume Sets 4-21
Initial Size Allocations 4-22
Contiguous Space 4-22
Noncontiguous Space 4-22
Size Extensions 4-22
Contiguous Space 4-22
Noncontiguous Space 4-22
Placement of Files in File Sets 4-23
Volume Set Storage Deallocation 4-23
File Set Storage Deallocation 4-24
File Storage Deallocation 4-24

Chapter 5: Storage and User File Management 5-1

User File Backup 5-1
File Attributes 5-2
BACKUP Attribute 5-2
VTF Attribute 5-3
CREATE BACKUP Command 5-3
BACKUP Command 5-4
ARCHIVE Command 5-4

User File Duplexing 5-4
Accessing the Duplex Copy of a File Extent in Place of a Disabled Primary Volume 5-5
Accessing the Duplex Copy of a File Extent in Place of an Offline Primary Volume 5-6
Accessing the Duplex Copy of a File Extent for Load Balancing 5-6

User File Retention 5-7
File Retention Attribute 5-7
Calculating File Retention Periods 5-7
Setting Retention Periods 5-8
File-level Retention 5-8
File Set-Level Retention 5-8
Determining File Retention Values When DEFAULT is Specified 5-9
Effect of Retention on Select StorHouse Commands 5-9
Release-Dependent Retention Information 5-10

User File Replication 5-10
Replication Class Name Attribute 5-11
Implementing Replication 5-12

User File Recovery 5-12

User File Removal 5-13
Removing Deleted Files 5-13
LIMIT Attribute 5-13

User File Migration From the Performance Buffer 5-14
ATF Attribute 5-14
MIGRATE Function 5-14
Migration Factor 5-14
Migration Function Operation 5-15

User File Migration Between Volume Sets 5-16
How Files are Selected for Volume Set File Migration 5-17
How Files are Migrated for Volume Set File Migration 5-18
Volume Set File Migration Operation 5-19

User File Staging 5-20

Volume Migration 5-21

Volume Export and Import 5-21
Volume Export and Uncatalog 5-22
Special Considerations for Uncataloging Volumes with Retained Files 5-22
Volume Import and Catalog 5-22

Volume Erasure 5-23

Volume Retirement 5-24

Volume Recovery 5-26

Chapter 6: The Account System 6-1

Account Identification Codes 6-1

Account Passwords 6-2

Account Privileges 6-2

Account Access Levels 6-2

Default Environment 6-3

Chapter 7: System Backup, Recovery, and Error Reporting 7-1

System File Backup 7-1
- Automatic Backup 7-1
- Manual System File Backup 7-2
  - System File Checkpoints 7-2
  - System File Extractions 7-2

System Recovery 7-3
- Normal Recovery 7-3
- Extended Recovery 7-4
  - Checkpoint Recovery 7-5
  - Directory Recovery From Extraction Files 7-5
  - Directory Recovery From Physical Volumes 7-6
  - Device Scan Recovery 7-7
    - Using Device Scan Recovery for Disaster Recovery. 7-7

Call Home Error Reporting 7-7

Appendix A: ASCII Characters A-1

Appendix B: Access and Command Privileges B-1

Index Index-1

Contents

Figures

Figure 1-1: StorHouse Application Examples 1-2
Figure 1-2: Typical StorHouse Configuration 1-3
Figure 1-3: StorHouse Interface Levels 1-4
Figure 2-1: 5.25-inch and 12-inch Optical Disk Library Devices 2-13
Figure 2-2: Magnetic Tape Library Device 2-15
Figure 3-1: File Organizations and Their File Extents 3-10
Figure 4-1: Optical Volume Set with Three File Sets 4-10
Figure 4-2: Volume Allocation 4-21
Figure 5-1: BACKUP, CREATE BACKUP, and ARCHIVE Commands 5-2
Figure 5-2: File Write-back and Migration 5-16
Figure 5-3: File Staging 5-20
Figure 5-4: IMPORT/CATALOG and UNCATALOG/EXPORT Operations 5-23
Figure 7-1: Checkpoint Recovery 7-5
Figure 7-2: Directory Recovery from Extraction Files 7-6

Tables

Table 3-1: File, File System, and Host Type Information 3-8
Table 3-2: File Extent Types 3-8
Table 3-3: File Access Functions 3-17
Table 3-4: File Access Methods 3-17
Table 3-5: File Access Modes 3-18

Tables

Table 3-6: Record Access Functions 3-18
Table 3-7: File Transfer Functions 3-19
Table 4-1: File Extent Statuses 4-14
Table 5-1: Replication Class Components 5-11
Table 5-2: RETIRE VOLUME Command Modifiers 5-25
Table 5-3: RECOVER VOLUME Modifiers 5-27
Table 7-1: Checkpoint System Parameters 7-2
Table 7-2: Extended Recovery Methods 7-4
Table A-1: Printable ASCII Characters A-1
Table A-2: ASCII Characters and Hexadecimal Values A-2
Table B-1: Access Privileges B-1
Table B-2: Command Privileges B-2

Welcome

Welcome to the *StorHouse Concepts and Facilities* manual, the primary reference document for the StorHouse/SM software. This book explains the basic concepts, structures, and functions of StorHouse/SM, including hardware and software components, capabilities, physical storage hierarchy and definition, data identification, storage allocation and control procedures, and account system.

What is StorHouse?

StorHouse® is FileTek's enterprise-wide software solution for managing the capture, storage, movement, and access of gigabytes to petabytes of relational and non-relational detail data. StorHouse technology combines industry-leading, scalable storage devices and Open Systems processors with FileTek's specialized storage management and relational database management system (RDBMS) software components.

StorHouse/SM

StorHouse/SM, FileTek's storage management component, controls a hierarchy of storage devices including cache, RAID, ATA disk, MAID, erasable and write-once-read-many (WORM) optical disk jukeboxes, and high-performance and high-capacity erasable and WORM tape in automated libraries. StorHouse/SM is also responsible for automating critical system management tasks such as data migration, backup, and recovery.

StorHouse/RM

StorHouse/RM, FileTek's RDBMS component, works in conjunction with StorHouse/SM to specifically administer the storage, access, and movement of

relational data. For more information on the StorHouse/RM product, refer to the StorHouse/RM User Document Set. Specifically, the *StorHouse/RM Concepts*, publication number 900132, provides an overview of StorHouse/RM concepts, structures, and functions.

StorHouse/Control Center

StorHouse/Control Center (CC) is FileTek's Windows®-based application for StorHouse system and database administration. It provides an easy-to-use graphical user interface (GUI) that simplifies StorHouse system and database management tasks. You can use the StorHouse/Control Center *StorHouse/Admin* module to define and manage StorHouse system resources. For more information about StorHouse/Control Center, refer to the StorHouse/Control Center User Document Set.

Purpose of This Document

This document defines StorHouse concepts, structures, and functions. It also describes StorHouse capabilities and operation.

Intended Audience

This document is intended for all StorHouse users, including programmers, system administrators, engineers, and operators. In the discussions that follow, the general user is referred to as *you*. The system administrator is specifically referred to as *system administrator* and the system operator is specifically referred to as *system operator*.

Contents

The document consists of seven chapters and two appendices:

- Chapter 1, "StorHouse Introduction," provides an introduction to StorHouse concepts and general functions.
- Chapter 2, "Physical Storage," explains how to identify devices and physical volumes. It also discusses volume and device access.
- Chapter 3, "Data Identification and Access Control," explains file identification, organization, directories, access, and access control.

- Chapter 4, “Storage Allocation,” describes how to allocate and control volumes, volume sets, and file sets.
- Chapter 5, “Storage and User File Management,” discusses file storage attributes, how to estimate file sizes, and storage management concepts.
- Chapter 6, “The Account System,” describes the StorHouse account system, including account identification, passwords, privileges, and account access levels.
- Chapter 7, “System Backup, Recovery, and Error Reporting,” describes system logging, backup, and recovery capabilities.
- Appendix A, “ASCII Characters,” lists the ASCII characters and their hexadecimal values.
- Appendix B, “Access and Command Privileges,” describes the access and command privileges required to execute StorHouse Command Language commands.

Related Documentation

You may want to refer to the following additional StorHouse documentation:

- The *Host Installation and Operations Guide*, publication number 900011 for IBM™ MVS™ hosts, 900051 for UNIX® hosts, and 900052 for DOS hosts, explains how to install the StorHouse host software. These guides are intended for system administrators and system programmers.
- The *StorHouse Glossary,* publication number 900027, defines StorHouse terminology. It is intended for all StorHouse users.
- The *Callable Interface Programmer’s Guide*, publication number 900013 for IBM MVS hosts and the *Generic Callable Interface Programmer’s Guide*, publication number 900046 for all other hosts, are references for programmers who write applications that use the Callable Interface. These guides explain the functions of the Callable Interface and contain a sample program.
- The *Command Language Reference Manual*, publication number 900005, is a general reference for StorHouse Command Language, the standard command interface between StorHouse and all host computers. It is intended for all StorHouse users.
- The *System Operator’s Guide*, publication number 900008, contains basic operating instructions for StorHouse hardware and software. It is intended for the system operator.

- The *System Administrator's Guide,* publication number 900007, describes system recovery, account administration, and storage management procedures and concepts for StorHouse. It is intended for the system administrator.
- The *Messages and Codes Manual,* publication number 900032, lists all StorHouse system and host messages by status code, gives the meaning of each message, and indicates any actions to take as a result of the messages. It is intended for all users.
- The *User Log Format* manual, publication 900028, describes the format of the StorHouse user log. It is intended for programmers who write applications to generate reports from statistical information contained in the user log.

Notational Conventions

This book uses the following conventions for illustrating command formats, presenting examples, and identifying special terms:

Convention	Meaning
Angle brackets (< >)	Enclose optional entries
Braces ({ })	Enclose descriptive terms or a choice of entries
`Courier font`	Code
Italics	New terms and emphasized text
lower case Helvetica font	User entries
UPPER CASE	System responses and StorHouse terms

Chapter

1

StorHouse Introduction

This chapter introduces FileTek's StorHouse data management system and discusses StorHouse:

- Hardware and software components
- Users, accounts, and sessions
- Interfaces
- Storage and file system
- System identification, parameters, and logging.

About StorHouse

FileTek specializes in enterprise data storage and access management solutions that support information lifecycle management processes. Its flagship product, StorHouse, leads the storage industry in scalability for very high volumes of data. StorHouse is ideally suited for large content repositories and archiving initiatives ranging from healthcare and seismic data to enterprise-wide documents and images.

FileTek customers typically use StorHouse in support of long-term records management and retention applications. Systems are often deployed to ensure compliance with regulatory requirements such as those found in the financial and telecommunications market spaces.

Numerous other customer installations benefit from StorHouse's cost and performance features for "active archive" applications. In these cases, individual records that are infrequently accessed in the production system are archived to StorHouse in a relational framework where they can be subsequently queried as if they still resided in their original production databases.

Figure 1-1 illustrates StorHouse application examples.

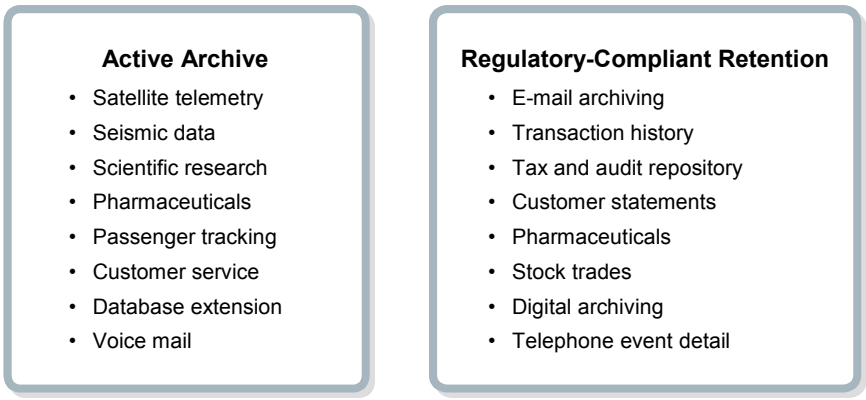

Figure 1-1: StorHouse Application Examples

StorHouse provides the following functions:

- Physical storage
- Data identification and access control
- Storage allocation and management
- Account control
- System management
- Host user access
- Installation and maintenance support.

StorHouse Software and Hardware

StorHouse software and hardware include the StorHouse operating system, a layered hierarchy of storage devices and media, and specialized StorHouse software that executes on the operating system. StorHouse also consists of software and hardware interfaces (network or direct-connect) to one or more host computer systems. This document refers to direct-channel connections used in place of network connections between StorHouse and its hosts as networks unless a distinction must be made between direct connections and network connections.

The StorHouse and network software run under control of the StorHouse operating system. The StorHouse/SM software consists of application programs and drivers. With this special software, the operating system automatically manages storage and data transfer functions. The StorHouse/SM software also includes record-access management software called the Virtual Record Access Manager (VRAM™). The network software, such as Transmission Control Protocol (TCP) or direct-connect support software, comprises a set of drivers and interface routines.

Figure 1-2 shows a typical StorHouse configuration.

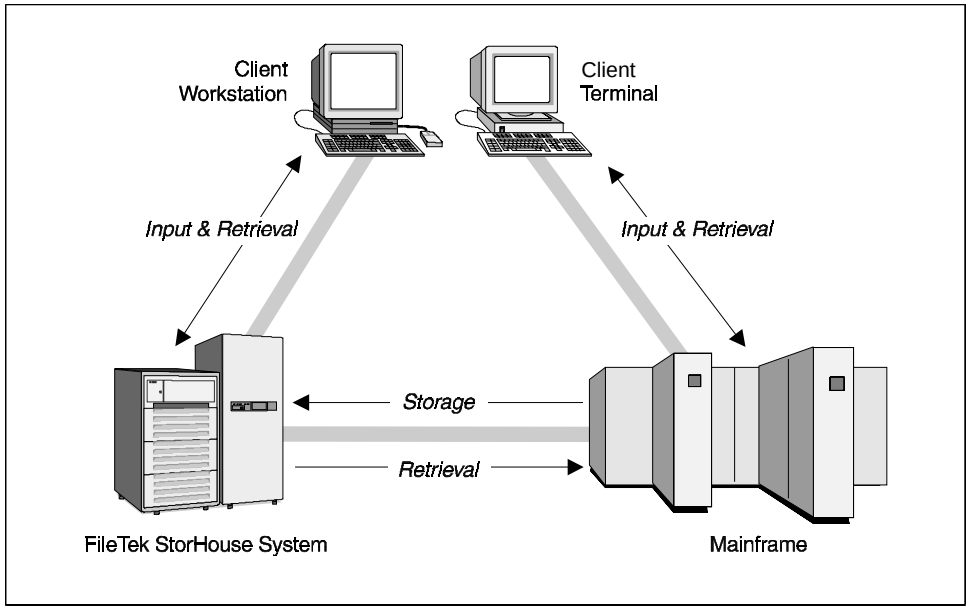

Figure 1-2: Typical StorHouse Configuration

StorHouse Users, Accounts, and Sessions

StorHouse users can be system administrators, system operators, applications, general users of a StorHouse host, or FileTek customer service representatives. StorHouse/SM controls all user access through an *account system*. Each user account has a set of privileges and can access one or more groups of files.

Users can gain access to StorHouse and execute StorHouse functions by starting a StorHouse *session*. A session begins when users sign on to StorHouse and ends when they sign off.

At signon, StorHouse assigns a unique four-digit (0-9) *user identification code* (uid) to each user session. A single user uid may change from session to session. For example, StorHouse may assign a uid of 0002 when a user session begins (at signon). After signoff (when the session for uid 0002 ends), StorHouse may reassign uid 0002 to another user.

StorHouse Interfaces

StorHouse has four interface levels: user, application program, basic StorHouse, and network support. Figure 1-3 shows these interface levels.

Users can access StorHouse from a host through a host interface, which consists of other interface levels. A host interface can use lower levels of interface or can include lower level interface functions. For example, an Interactive Interface may use a Callable Interface to communicate with StorHouse. In other cases, the Interactive Interface may perform Callable Interface and other lower level functions to communicate with StorHouse directly, and may not use a separate Callable Interface.

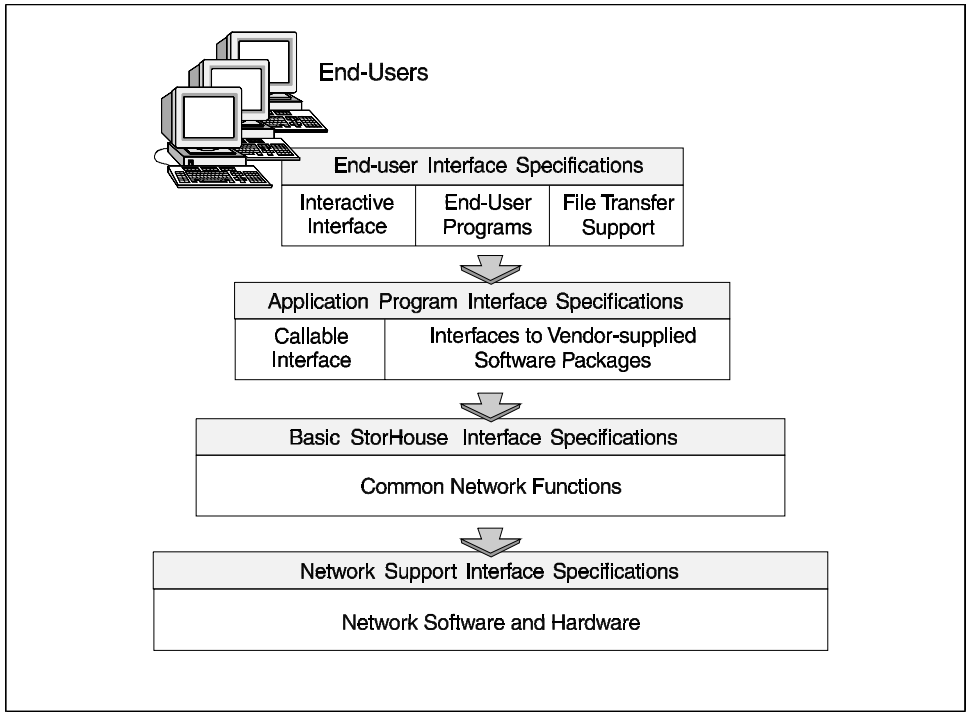

Figure 1-3: StorHouse Interface Levels

User Interfaces

The StorHouse Command Language allows users to execute StorHouse control and file transfer functions. A host Interactive Interface allows host interactive users to enter Command Language commands and receive responses. A host Disk File Transfer Interface provides file transfer functions for user files on host storage devices.

Various utilities and an operator interface enable system administrators, operators, and system engineers to install and maintain a StorHouse system.

Any StorHouse interactive user with appropriate privileges can receive and reply to StorHouse operator messages. IBM hosts require MVS StorHouse subsystem support to allow operators at MVS operator consoles to receive and reply to StorHouse operator messages.

The SET USER, SHOW USER, and CONSOLE commands and the CONSOLE and OPERATOR privileges support these functions. Refer to the *System Operator's Guide* and the *Command Language Reference Manual* for more information.

Application Program Interface

The application program interface consists of a Callable Interface and interfaces to various vendor-provided software (for example, report management packages).

- Host applications programs use a host Callable Interface to execute functions, transfer data, and receive responses. Refer to the *Callable Interface Programmer's Guide* for more information.
- StorHouse interfaces to vendor-supplied report management software allow you to retrieve and view reports that are stored on StorHouse media from these packages.

Basic StorHouse Interface

The Basic StorHouse Interface defines structures and functions that minimize differences between networks that connect hosts to StorHouse.

Network Support Interface

The Network Support Interface specifications describe the network that connects hosts to StorHouse and the functions used to access it. This may be one of the supported Local Area Networks (LANs) or a direct channel connection.

Storage and File System

StorHouse is a large storage system consisting of physical storage components. Physical storage, which includes devices, media, and their characteristics, determines performance and access capabilities for data.

The user interfaces to StorHouse define naming and access rules for various logical structures. Some structures identify and control access to data. Others control

physical storage and its allocation for data storage. Together, these make up the *StorHouse File System*.

The File System allows you to organize, identify, store, protect, and retrieve character and binary data, while insulating them from physical storage requirements. StorHouse stores binary data as 8-bit bytes. It uses 8-bit ASCII codes for character data. The printable ASCII characters and their hexadecimal values are listed in Appendix A, "ASCII Characters."

System Identification

Each StorHouse system has two system-level identifiers: a *system identifier* (sid) and a *site identifier*. FileTek assigns each StorHouse system a unique system identifier when the system is manufactured. The value can range from 1 through 65,535. The StorHouse system parameter SYSTEM_ID contains the value of the system identifier.

FileTek can assign the site identifier before installation or your StorHouse system administrator can request one. A site identifier may consist of up to 14 printable or space characters. Trailing space characters are ignored. The StorHouse system parameter SITE_ID contains the value of the site identifier.

System Parameters

System parameters are named data fields that the system uses to manage resources and provide default information for functions. Each parameter has a name and a value. System parameters allow the system administrator to configure and tune the StorHouse system for maximum performance at your site. The *System Administrator's Guide* defines all StorHouse system parameters.

There are three kinds of system parameters:

- *Static* – cannot change while StorHouse software is operating.
- *Dynamic* – can change while the StorHouse software is operating, and changes take effect immediately.

  Not all dynamic parameters can be altered by the system administrator. StorHouse changes some dynamic parameters automatically to reflect the current system status.
- *Deferred Dynamic* – can change while the StorHouse software is operating, but changes do not take effect until the system is restarted.

System Logging

StorHouse writes to two logs from system startup to system shutdown: the administration log and the user log. System administrators can close current administration and/or user logs and open new ones with the NEWLOG command. In addition, reporting applications, such as FileTek's StorHouse/Performance Monitor product, can read data extracted from log files and generate reports based on that information.

Administration Log

The *administration log* is a system-generated file that contains hardware status, error information, and general data used by FileTek customer service representatives to analyze system operations. This log is not accessible to users. However, users with MESSAGE privilege can record information in the administration log by using the MESSAGE /LOG command.

User Log

The *user log* is a StorHouse file that contains statistical and administrative information about StorHouse operations. StorHouse writes information to this file when a certain event occurs or after a specified amount of time. Logged information helps system administrators monitor StorHouse activity.

User log data records report the following statistical information:

- System startup/shutdown information
- User signon/signoff statistics
- Security and retention violation attempts (for signons, access groups, files, and commands)
- Command execution
- File opens/closes
- Volume mounts, dismounts, and volume movement information
- Device errors and device state changes
- Library device information (over an interval)
- Drive information (over an interval)
- System heartbeat
- Discarded log record count
- Operator messages
- File copies
- Extent transfers
- Error and device usage statistics
- StorHouse/RM session connections and disconnections, and security violations
- SQL statements received
- Completed SQL transactions
- Permanently closed StorHouse/RM files or ended transactions.

StorHouse also provides a message protocol for retrieving the most current user log records in real-time. This protocol communicates message requests and replies between one or more client applications and the StorHouse user log server.

Refer to the *User Log Format* manual for more information about user log record formats and the user log message protocol.

Chapter 2

Physical Storage

Physical storage consists of StorHouse devices, media, and their characteristics. All physical storage (hardware) referenced in this manual may not be used at each site.

Media are volumes that store data (for example, magnetic disks and optical or tape cartridges). Physical units of media are called *physical volumes*.

You must use only FileTek-certified media in your StorHouse systems. FileTek is not responsible for problems caused by using non-standard, non-certified, or non-compatible media. Maintenance, repair, or alteration services for damages resulting from such problems shall be your responsibility and at your cost. Contact your FileTek customer service representative for a list of all FileTek-certified media.

Devices are StorHouse hardware components. They include *units* (magnetic disk drives, automated optical or tape libraries, and shelf storage, among others) and *subunits* (accessors, drives, exchange stations, and slots). Some devices contain one or more physical volumes as integral or removable parts.

The remainder of this chapter discusses:

- Device identification
- Physical volume identification
- Volume and device access.

Device Identification

StorHouse uses *device identification codes* (did) to identify devices, device components, and volume locations.

The format of a device identification code is:

{level}{unit_number}{subunit_type}{subunit_number}

Note the following:

- Device identifiers do not always include all subfields.
- The brackets { } are not part of the did specification. Examples of device identifiers are L00, S00, L00A00, and L00D00.
- All unit and subunit numbers are represented by hexadecimal characters, which, include A-F, 0-9, and a-f.

Certain StorHouse Command Language commands require particular did specifications. Refer to the *Command Language Reference Manual* for information about specifying device identifiers in individual commands. The following sections describe did subfields.

Storage and Device Levels

Storage and *device levels* contain one character that corresponds to a type of storage or use of a device. Valid values for level are:

- F – Fixed (non-removable) storage devices
- L – Library devices with online data devices and mechanical access to removable volumes in slots
- N – Network devices
- S – Storage shelves requiring the operator to transfer removable volumes between shelves and online devices

Offline (exported) removable volumes are not part of the automatically managed library of volumes and are not maintained in the StorHouse directory.

Each device on the library level has a corresponding shelf storage designation.

Unit Number

The *unit number* consists of two hexadecimal characters. Each physically separate device is assigned a number starting from 00. For example, a library device can have a level and unit number of L00.

The unit number represents a single device or a device that contains other devices known as *subunits*. An example of a device that contains subunits is a library device, which contains data devices, volume storage slots, and one or more volume accessor mechanisms. Units that use removable volumes have at least one exchange station for entering and removing volumes.

Subunit Type

The *subunit type* consists of a single character that represents the type of device within a unit. Valid subunit values are:

- A – Accessor (includes bar code reader in tape library devices)
- D – Drive
- E – Exchange station
- S – Slot.

These subunit types are defined as follows:

- An *accessor* is a device that inserts or extracts volumes in drives, slots, or exchange stations within a unit and holds volumes when they are being moved from one location to another within the unit. A *transport arm* or *robotic arm* is a transport mechanism on which one or two accessors are mounted. The robotic arm moves the accessors between locations within a unit.

  In a tape library device, a *bar code reader* is part of the accessor subunit. A bar code reader is a device, generally mounted on a transport arm in a tape unit, which automatically reads bar code labels on tape volumes to identify them.

- A *drive* is a data read and/or write device in a unit.

- An *exchange station* is the location in a unit where an operator can load and unload volumes.

- A *slot* represents the physical location of a volume in a unit. Each unit has its own set of slot identifiers.

Subunit Number

The *subunit number* identifies specific devices or locations within a unit. The number consists of two hexadecimal characters for all subunit types except slots (subunit type S). For slots, the subunit number consists of six hexadecimal characters.

In most cases, subunit numbers are assigned consecutively starting from 0 for each subunit of a particular type in a unit. Some devices require specific numbers for slots.

Physical Volume Identification

Physical volumes are physical units of media where data can be recorded and read. The characteristics of a physical volume determine how the volume is identified and managed.

Volume Identification Code

StorHouse identifies volumes with a *volume identification code* (vid). For a physical volume, the vid contains three concatenated subfields, as follows:

{media_type}{recording_type}{volume_label}

A *logical* volume is one side or surface of a physical volume. The vid of a logical volume consists of its physical volume vid plus a colon and a side indicator, as follows:

{media_type}{recording_type}{volume_label}:{side}

Some physical volumes, such as magnetic tape, have only one usable side and contain only one logical volume. Others, such as most types of optical disks, have two usable sides and contain two logical volumes. Some StorHouse operations, such as moving volumes between devices, require you to specify a physical volume vid. Others operate on specific sides and require you to specify a logical volume vid. The individual StorHouse commands in the *Command Language Reference Manual* will indicate which vid type to specify.

The vid subfields are defined in the following sections.

Media Type and Recording Type

The *media type* consists of two alphanumeric characters that identify the type of drive in which a volume can be formatted and/or processed. A drive can support only one media type. The first character defines the general media type: M for magnetic disk, O for optical disk, P for power-managed disk arrays, or T for magnetic tape. The second character identifies a specific form factor, format, or other defining characteristic within the general media type.

The *recording type* consists of one alphabetic character that identifies the mode of recording used to format the volume and/or the fixed mode for the volume. A single drive may be able to support multiple recording types. Recording types depend on their accompanying media types. Generally, the letter A refers to the natural or standard format of a media type.

A sector-reusable medium allows StorHouse to allocate and return free space on a file-by-file basis. StorHouse uses magnetic disk as a *sector-reusable* medium. A *volume-reusable* medium requires that all file space on a volume be marked as uncataloged space before that space can be reused. StorHouse uses erasable optical disk and magnetic tape as volume-reusable media. A non-erasable or Write-Once-Read-Many (WORM) medium cannot be erased or rewritten after being initially written.

The current media types and recording types used by StorHouse for storage media are listed on the following pages. For the current list of libraries and devices associated with these media and recording types, contact your FileTek Customer Support representative.

- MA – has the following characteristics:

  - Seagate, fixed, magnetic disk
  - Sector-reusable
  - Standard (single) density is 1.3 GB
  - Recording types:
    - A indicates single-sided, single-density

- MB – has the following characteristics:

  - Maxtor, fixed, magnetic disk
  - Sector-reusable
  - Standard (single) density is 669 MB
  - Recording types:
    - A indicates single-sided, single-density

- MC – has the following characteristics:

  - IBM, fixed, magnetic disk
  - Sector-reusable
  - Standard (single) density is 1 GB
  - Recording types:
    - A indicates single-sided, single-density

- MD – has the following characteristics:

  - IBM, fixed, magnetic disk
  - Sector-reusable
  - Standard (single) density is 2 GB
  - Recording types:
    - A indicates single-sided, double-density.

- ME – has the following characteristics:

  - RAID, fixed, magnetic disk
  - Sector-reusable
  - Standard (single) density is 8 GB
  - Recording types:
    - A indicates standard density.

- MF – has the following characteristics:

  - Symbios RAID, fixed, magnetic disk
  - Sector-reusable
  - Recording types:
    - A indicates 16 GB density

- MG – has the following characteristics:

  - Seagate, fixed, magnetic disk
  - Sector-reusable
  - Standard (single) density is 2 GB
  - Recording types:
    - A indicates 2 GB density
    - B indicates 4 GB density

- MI identifies any storage organized as a UNIX filesystem, generally fixed sector-reuseable magnetic disk, either locally or remotely (e.g., NFS) mounted. Recording type A is a normal read/write/delete filesystem device, while recording type B is reserved for devices that have a retention capability. The specific filesystem determines other media characteristics, such as device and sector size.

- MJ specifies a Centera storage device. Recording type A identifies a Centera device operating in non-compliance mode. Recording type B indicates a Centera device operating in Basic Compliance or Compliance Plus mode. Media type MJ identifies the vendor application program interface (API) used to access the storage device. Vendor software/firmware controls all media characteristics.

- OA – has the following characteristics:

  - 12-inch, removable, optical disk cartridges
  - Non-erasable – once written, data cannot be erased or rewritten
  - Standard (single) density volumes are 1.3 GB per side. Double-density volumes are 3.5 GB per side.
  - Volumes are double-sided. Only one side is accessed at a time.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life of approximately 30 years.
  - A volume can be read many times without degrading the medium.
  - Medium supports multiple concurrent reads and one write.
  - Recording types:
    - B indicates double-sided, single-density
    - D indicates double-sided, double-density.

Note A second-generation optical drive for media type OA can support recording types B and D if the hardware is equipped to do so.

- OC – has the following characteristics:

  - 12-inch, removable, optical disk cartridges
  - Non-erasable – once written, cannot be erased or rewritten
  - Volumes are double-sided. Both sides are accessed simultaneously.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life in excess of 20 years.
  - A volume can be read many times without degrading the medium.
  - Medium supports multiple concurrent reads and one write.

- Recording types:
  - A indicates 12 GB, non-erasable volumes
  - B indicates 30 GB, non-erasable volumes

Note The media density nomenclature for 5.25-inch optical disks has changed to follow industry-standard terminology. Media previously documented as 1.0 GB are now called triple-density (3X). The 1.3 GB media are called quadruple-density (4X). The 2.6 GB media are called octal-density (8X).

- OE – has the following characteristics:

  - 5.25-inch, removable, optical disk cartridges
  - Erasable or non-erasable, depending on the recording type
  - Volumes are double-sided. Only one side is accessed at a time.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life in excess of 20 years.
  - A volume can be read many times without degrading the medium.
  - Medium supports multiple concurrent reads and one write.
  - Recording types:
    - A indicates Hitachi-compatible, double-sided, triple-density (3X), 1.0 GB per side, non-erasable volumes
    - B indicates Hitachi-compatible, double-sided, triple-density (3X), 1.0 GB per side, erasable volumes
    - C indicates Hitachi-compatible, double-sided, quadruple-density (4X), 1.3 GB per side, non-erasable volumes
    - D indicates Hitachi-compatible, double-sided, quadruple-density (4X), 1.3 GB per side, erasable volumes.
    - E indicates double-sided, octal-density (8X), 2.6 GB per side, non-erasable volumes.
    - F indicates double-sided, octal-density (8X), 2.6 GB per side, erasable volumes.

- PA – has the following characteristics:

  - MAID device emulating a tape library and tape drives.
  - High-performance, faster access than all except the fastest tape or optical.
  - "Zero time" extent positioning.
  - Erasable (volume reusable).
  - Drives optionally support compression, which can increase average storage capacity.
  - Volumes (Virtual Tape Volumes, or VTCs) are single-sided.
  - VTCs are permanent and cannot be removed.
  - VTCs have a shelf life that is a function of time-powered-on and number of power cycles.
  - A volume can be read many times without degrading the medium.
  - Medium supports one read or one write at a time.
  - Recording type:
    - A indicates COPAN Revolution 200T shelf, 27 VTCs of 750 GB each, maximum of 7 "drives,"that is, 7 volumes concurrently powered on.

- TB – has the following characteristics:

  - Digital linear tape (DLT), removable magnetic tape cartridges
  - Erasable (volume reusable)
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life of approximately 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use.
  - Medium supports one read or one write at a time.
  - Recording types:
    - B indicates CompacTapeIV, 85,937 bpi per track (there are 52 track quads for a total of 208 tracks), 541.9-meter, uncompressed data cartridge (approximately 35 GB)
    - Z indicates a 20-pass cleaning cartridge that can be used up to 20 times to clean tape drive heads.

- TC – has the following characteristics:

  - 3480 standard magnetic tape cartridges
  - Erasable (volume reusable)
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life of approximately 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use.
  - Medium supports one read or one write at a time.
  - Recording types:
    - A indicates uncompressed data cartridge (approximately 405 MB)
    - B indicates enhanced (E-tape) uncompressed data cartridge (approximately 810 MB)
    - Z indicates a 500-pass cleaning cartridge that can be used up to 500 times to clean tape drive heads.

- TD – has the following characteristics:

  - High-performance, 9840 magnetic tape cartridges
  - Erasable (volume reusable)
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life ranging from 15 to 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use.
  - Medium supports one read or one write at a time.
  - Recording types:
    - A indicates data cartridge (approximately 20 GB uncompressed)

    - Z indicates a 100-pass cleaning cartridge that can be used up to 100 times to clean tape drive heads.

- TE – has the following characteristics:

  - High-performance, T9940 magnetic tape cartridges
  - Erasable (volume reusable)
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life ranging from 15 to 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use.
  - Medium supports one read or one write at a time.
  - Recording types:
    - A indicates data cartridge (approximately 60 GB uncompressed)
    - B indicates data cartridge (approximately 200 GB uncompressed)
    - Z indicates a 100-pass cleaning cartridge that can be used up to 100 times to clean tape drive heads.

- TF – has the following characteristics:

  - LTO Ultrium high-performance, magnetic tape cartridges
  - Erasable (volume reusable)
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life ranging from 15 to 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use.
  - Medium supports one read or one write at a time.
  - Recording types:
    - A indicates data cartridge (approximately 100 GB uncompressed)
    - Z indicates a 100-pass cleaning cartridge that can be used up to 100 times to clean tape drive heads.

- TG – has the following characteristics:

  - Sony magnetic tape cartridges
  - Erasable or non-erasable, depending on the recording type
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges and can be stored on shelves.
  - Volumes have a shelf life of up to 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use.
  - Medium supports one read or one write at a time.
  - Recording types:

    - A indicates an AIT-2 erasable data cartridge (approximately 50 GB uncompressed)
    - B indicates an AIT-2 data non-erasable cartridge (approximately 50 GB uncompressed)
    - C indicates an AIT-3 erasable data cartridge (approximately 100 GB uncompressed)
    - D indicates an AIT-3 data non-erasable cartridge (approximately 100 GB uncompressed)
    - Z indicates a 70-pass cleaning cartridge that can be used up to 70 times to clean tape drive heads.

- TH – has the following characteristics:

  - 3599 magnetic tape cartridges (for IBM 3592J1A drives).
  - Erasable (volume reusable).
  - Drives support compression, which can increase average storage capacity.
  - Volumes are single-sided.
  - Volumes are enclosed in plastic cartridges that can withstand a 1-meter drop and can be stored on shelves.
  - Volumes have a shelf life of approximately 30 years.
  - A volume can be read and/or written many times, but the medium degrades with use, including load/unload cycles.
  - Medium supports one read or one write at a time.
  - Recording types:
    - A indicates rewritable data cartridges (approximately 300 GB)
    - B indicates write-once data cartridge (approximately 300 GB)
    - Z indicates a 300-pass cleaning cartridge to clean tape drive heads.

StorHouse uses the following media type and recording type designations for networks, which can be used by the StorHouse software as the source or destination medium of a data transfer:

- NA – has the following characteristics:

  - Transmission Control Protocol (TCP) network.
  - Recording type A indicates normal format.

- NC – has the following characteristics:

  - FileTek Direct Connect network.
  - Recording type A indicates normal format.

Volume Label

The *volume label* is a code similar to a serial number that makes the volume identification code (vid) unique for every volume of a specific media and recording type. The following vid shows the position of a volume label in bold for a *physical* volume:

{media_type}{recording_type}**{volume_label}**

For *logical* volumes, the volume label is positioned as follows within the vid:

{media_type}{recording_type}**{volume_label}**:{side}

StorHouse generates volume labels automatically unless the hardware is equipped with a bar code reader (for tape volumes only). If a tape volume has an external bar code label, StorHouse reads the bar code and uses it as the volume label in the vid when it adds the tape to the pool of empty volumes in the library.

StorHouse assigns labels for removable volumes according to customized formats your system administrator specifies or according to the default time method. The default time method uses an algorithm to generate labels based on the time StorHouse initializes a volume.

Volume labels contain 1 to 8 uppercase alphanumeric characters for all optical media types. Tape labels contain 1 to 6 uppercase alphanumeric characters for all tape media types. For example, in the volume identification code TBB246835, the volume label is 246835.

Side

The *side indicator* specifies one side, or surface, of a physical volume. It consists of one alphabetic character: A for a single-sided volume, or either A or B for a double-sided volume.

If more than one surface of a physical volume can be accessed while in a drive, all the accessible surfaces can be identified by one side indicator. For example, a magnetic disk volume may consist of many surfaces, but all the surfaces are accessible while the drive is mounted. The entire set of surfaces is identified by a single side indicator, A.

Volume and Device Access

StorHouse uses read/write devices (drives), library devices, and operators to access volumes. When you transfer data between a host and StorHouse, StorHouse allocates the necessary devices and volumes to transfer the data. It queues and distributes work to keep the drives busy with data transfers and to minimize the number of physical volume transfers.

Magnetic Disk Drives

Magnetic disk drives read and write data on fixed, magnetic disk volumes. Some StorHouse systems offer magnetic disks in the form of Redundant Array of Independent Disks (RAID) units. While the drives are up (operating), the volumes

are always mounted and ready for use. These devices access blocks of data randomly as well as sequentially. Magnetic disk drives are shared devices. This means that StorHouse can transfer multiple files to or from a single volume concurrently.

The Performance Buffer

The *performance buffer* is a portion of level F magnetic disk space that is used as a holding area for performance copies of user files. These files are temporarily buffered on magnetic disk prior to being copied to their final destination on optical or tape media. The performance buffer is generally invisible to the user.

If needed, a file access group can reserve a portion of the performance buffer for its exclusive use. (See "Files and File Access Groups" on page 3-1 for information on file access groups.)

Optical Disk Library Devices

An *optical disk library device* transfers physical optical disk volumes (cartridges) between drives, a volume exchange station, and storage slots. An optical disk library device can support one removable media type and either one or more than one recording type for that media type. A single StorHouse system can support multiple optical disk library devices that contain the same or mixed media types. Figure 2-1 shows 5.25-inch and 12-inch optical disk library devices.

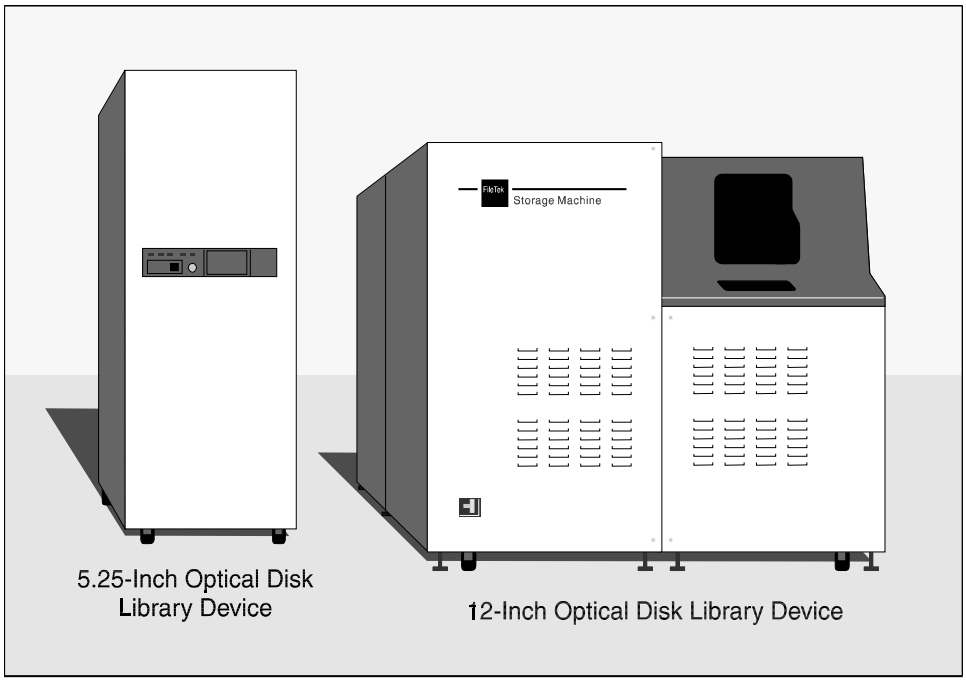

Figure 2-1: 5.25-inch and 12-inch Optical Disk Library Devices

An optical disk library device contains the following subunits:

- A *robotic arm* that moves volumes between storage slots and drives. The part of the arm that holds a volume is called an *accessor*. A robotic arm can have one or two accessors.
- Two or more optical disk drives
- An exchange station where an operator can load or unload optical cartridges from the library device
- Multiple slots for storing optical disk cartridges.

The following sections present a detailed description of optical disk drives and the optical disk library exchange station.

Optical Disk Drives

Optical disk drives read and write data on removable optical disk volumes. These drives can access blocks of data randomly and sequentially.

Optical disk drives may be dedicated or shared devices. If a drive is a dedicated device, it can support only one file read or write at a time. If a drive is a shared device,

it can read multiple files from a drive concurrently, while it may also be writing a single file to the same drive.

In the unlikely event that an optical disk drive experiences a failure, StorHouse may automatically place it in a read-only state (depending on the type of error). The drive will remain in the read-only state until it can be repaired or replaced. The impact of a failed drive on customer operations is significantly reduced by allowing the drive to remain in a read-only state instead of taking it offline. If necessary, a system operator can manually place an optical drive into a read-only state using the SET DEVICE command. The operator can determine whether an optical disk drive is in the read-only state by using the SHOW DEVICE command.

Depending on the library device and its drives, optical drives can access one or both sides of a volume at a time. In library devices that can access one volume side at a time, StorHouse automatically removes the volume from a drive, flips it, and reinserts it in the drive to access the other side.

Some optical disk drives support one recording type, while others support multiple recording types. For example, the drives that support media type OE (5.25-inch optical disk volumes) can read and write volumes using either recording type A or C (non-erasable format), or B or D (erasable format).

Note Once StorHouse assigns a recording type to a volume, it will write the volume only in that recording type.

For media type OA (12-inch, non-erasable, optical disk), first-generation drives can read and write recording type B only. Second-generation drives can read and write recording type D and, if equipped with the requisite hardware, can read (but not write) recording type B.

Optical Disk Library Exchange Station

An optical disk library device has a built-in *exchange station*. The operator can insert or extract an optical cartridge at the station while the accessor is in use.

If StorHouse needs a cartridge, it displays a message on the system console requesting the operator to write a label on a blank cartridge and insert that cartridge into the exchange station, or to insert a previously used cartridge in the station. The system may direct the operator to remove a cartridge from the station at the same time.

After the operator loads the cartridge into the station and enters a reply at the console, the StorHouse software directs the accessor to transfer the cartridge from the station to a drive. If the cartridge is a blank, the system records a label record on it and adds the cartridge to the pool of empty cartridges in the library device.

When the drive has finished using the cartridge, the accessor returns the cartridge to the exchange station or to a slot. If the cartridge is placed in the exchange station, the system directs the operator to remove the cartridge and place it on a shelf or in a separate storage area.

Automated Magnetic Tape Library Devices

Automated magnetic tape library devices transfer physical tape volumes (cartridges) between drives, a volume exchange station, and storage slots. Some magnetic tape library devices support one removable media type/recording type combination; others support multiple recording types; and some of the newer libraries support multiple media types. A single StorHouse system can support multiple library devices that contain the same or mixed media types. Figure 2-2 shows an automated magnetic tape library device.

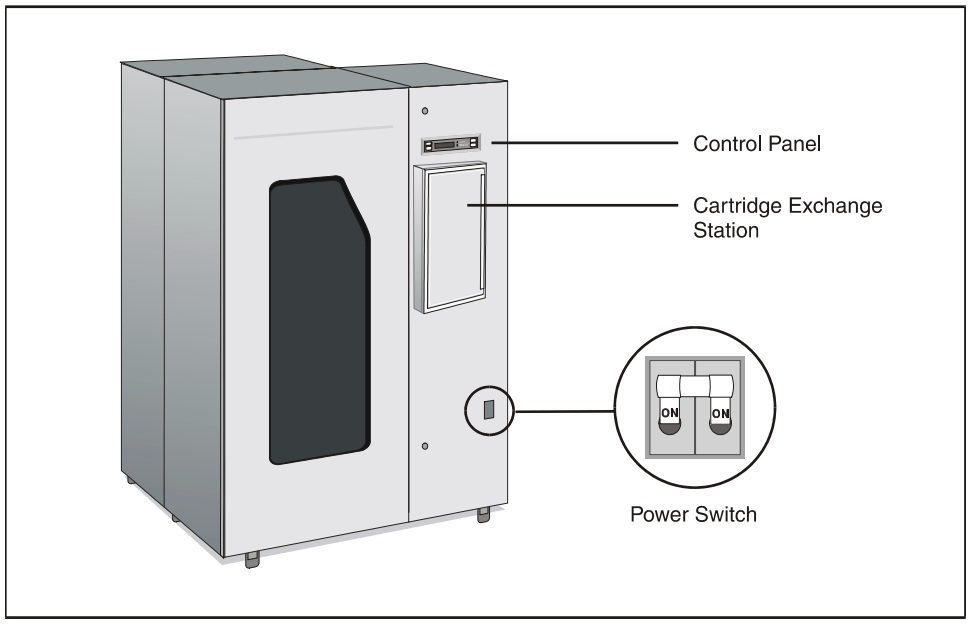

Figure 2-2: Magnetic Tape Library Device

Magnetic tape library devices contain the following subunits:

- A *robotic arm* that moves volumes between storage slots and drives. The part of the transport mechanism that holds a volume is called an *accessor*. The robotic arm has one accessor.

  A *bar code reader* is also mounted on the robotic arm. A bar code reader can automatically read bar code labels on tape cartridges to identify cartridges without having to load them into a drive.

- Multiple tape drives.

- An exchange station where an operator can load or unload tape cartridges from the library device.

- Multiple slots for storing tape cartridges.

Bar Code Readers

A robotic arm in a magnetic tape library device is equipped with a *bar code reader* that can read bar code labels on tape cartridges. StorHouse uses this feature to identify a volume without having to load it in a drive and perform a read operation. The bar code reader is part of the robotic arm subunit in a magnetic tape library device.

Magnetic Tape Drives

Magnetic tape drives read and write data on removable magnetic tape volumes. These drives can access blocks of data *sequentially* on tape. However, using the StorHouse software, you can *randomly* locate and extract individual blocks of data.

Magnetic tape drives are dedicated devices. A *dedicated* device can support only one file read or write at a time. However, the StorHouse software provides priority processing for tape drives, which allows high-priority reads to be queued and processed before lower priority operations.

In the unlikely event that a tape drive experiences a failure, StorHouse may automatically place it in a read-only state (depending of the type of error). The drive will remain in the read-only state until it can be repaired or replaced. The impact of a failed drive on customer operations is significantly reduced by allowing the drive to remain in a read-only state instead of taking it offline. If necessary, a system operator can manually place a tape drive into a read-only state using the SET DEVICE command. The operator can determine whether a tape drive is in the read-only state by using the SHOW DEVICE command.

The read/write heads in a magnetic tape drive require occasional cleaning to prevent errors due to dirty heads and to prevent damage to the drives. (The cleaning interval, which is determined by the drives, is completely dependent on usage and will vary.) StorHouse supports the use of special head-cleaning tape cartridges for this purpose. Typically, cleaning tapes are permanently stored in tape slots. The system tracks drive usage and automatically inserts a cleaning tape cartridge in a drive when needed. A cleaning cartridge can be used a specific number of times before it must be replaced. The system tracks cleaning cartridge usage and automatically directs the operator to unload an old cleaning cartridge and load a new one the next time the old cartridge is requested after the limit is reached. (StorHouse automatically retires cleaning cartridges when they reach their lifecycle limit if you have scheduled the RETIRE VOLUME command.)

Magnetic tape drives that support media types TB, TC, and TD are equipped with compression hardware that can read and write tapes of specific recording types. Magnetic tape drives can be cleaned using recording type Z cleaning tapes.

Note Once StorHouse assigns a recording type to a tape, it will write the tape only in that recording type. Also, if StorHouse is unable to clean a tape drive (for example, there are no usable cleaning cartridges available in the system), StorHouse will bring down the drive until you load a new tape cleaning cartridge.

Magnetic Tape Library Exchange Station

Magnetic tape library devices have a built-in *exchange station*. See Figure 2-2 on page 2-15 for the location of the exchange station in a magnetic tape library device. StorHouse Releases 5.1 and above support multiple-slot exchange stations.

If StorHouse needs a cartridge(s), it displays a message on the system console directing the operator to write an external label (or place a bar code label) on one or more blank cartridges and insert the cartridge(s) in the exchange station, or to insert one or more previously used cartridges in the station. The system may also direct the operator to remove one or more cartridges from the station.

After the operator loads the cartridge(s) into the station and enters a reply at the console, the StorHouse software directs the accessor to transfer the cartridge(s) from the station to a drive. If a cartridge is blank, the system records an internal label record on it and adds the cartridge to the pool of empty cartridges in the library device.

When the drive has finished using the cartridge(s), the accessor returns the cartridge(s) to the exchange station or to a slot(s). If one or more cartridges are placed in the exchange station, the system directs the operator to remove the cartridge(s) and place it (them) on a shelf or in a separate storage area.

Library Device Malfunction

If the robotic arm breaks down, cartridges cannot be transferred to and from drives. However, if one drive goes down, operation of other library device components is not affected as long as the other drive(s) in the library device are operational.

Performance Enhancements for Library Devices

The following performance enhancements are available for your site:

- Improved performance for volume operations in library devices that cannot exchange volumes at the drive, either because they have only one accessor or because only one accessor is available.
- Ability to minimize volume mounts within a library device.

Improved Performance for Volume Operations

To improve volume operations for library devices that cannot exchange volumes at the drive, StorHouse will automatically dismount a volume from one drive and move it to a slot if no other drives in the library device are empty and the volume in that drive has not been accessed for a period of time (in other words, the drive has been

idle). This action allows one drive to remain free at all times for the next volume that has to be mounted. This capability is transparent to users.

To improve performance for library devices that cannot exchange volumes at the drive, the system administrator can set the TMD_DISMNT_DELAY system parameter. For information on this feature and the system parameter, see the *System Administrator's Guide*.

Ability to Minimize Volume Mounts

The system administrator can minimize volume mounts within a library device to improve system performance. To do so, the system administrator can set two system parameters: TMD_HOLD_HIGH and TMD_HOLD_LOW. For information on this feature and these system parameters, see the *System Administrator's Guide*.

Support for ACSLS and LibraryStation Software

FileTek supports Automated Cartridge System Library Software (ACSLS) for UNIX-based systems and LibraryStation software for MVS-based systems. These two software solutions centrally manage library requests from applications running on different host platforms. This centralized data management enables libraries to be shared across the enterprise. For more information on ACSLS or LibraryStation, contact your FileTek customer support representative.

Chapter

3

Data Identification and Access Control

Data identification and access control rules define structures for organizing, identifying, and accessing user data. This chapter discusses:

- Files and file access groups
- File versions, structures, organization, format identification, extent types, directories, access, and control.

Files and File Access Groups

A *file* is a collection of logically related data, located on a medium, and treated as a unit. A file is uniquely identified by a file identifier. Revisions of files are identified by revision numbers. Files can also be named.

A *file access group* is a set of named files. Each named file in a StorHouse system is a member of one access group. A named file is uniquely identified by an access group name, a file name, and a version number.

File Identifier and File Number

StorHouse assigns a unique *file identifier* (fid) to each file that is created. A file identifier consists of two numbers: the *system identifier* (sid) and a *file number* (fno). All files created in a single StorHouse system have the same system identifier. The file number makes the file identifier unique within the StorHouse system.

For example, all files created in one StorHouse system share a system identifier, such as 234. However, their file numbers are unique. These files might have file identifiers of 234.77, 234.108, and 234.8211.

File Names

A *file name* is a unique name that identifies a file within an access group. StorHouse file names must contain printable ASCII characters. File names must contain 1 to 56 characters and at least one character must be non-blank. Lowercase characters are distinct from uppercase characters. In a command, you must enclose a file name in quotes if it contains lowercase or special characters, or spaces. Examples of valid StorHouse file names are BANKFILE and "BankFile".

Access Group Names

An *access group name* is a unique name that identifies a group in a StorHouse system. Access group names can contain 1 to 8 characters and consist of the following ASCII characters: A-Z (uppercase), 0-9, _ (underscore), and $ (dollar sign). An example of a valid StorHouse group name is BANKGRP.

File Versions

StorHouse allows you to create a new *version* of a file with the same group name and file name as an existing file. StorHouse uses relative version numbers to distinguish between different files with the same group and file names. Each version has a unique file identifier. StorHouse can maintain up to 32,768 versions of a file for one file name.

Relative version numbers range from 0, the current version, through -32,767, the oldest version. The relative version number of a file changes each time a new version is added and may change when an existing version is deleted.

If a new version is added, the new version becomes the current version, or relative version 0. If there was a previous version 0, it becomes version -1. If there was a previous version -1, it becomes version -2, and so on until the maximum number of versions is exceeded. If the maximum number of versions is exceeded, the last one is deleted. If you were to delete relative version -1, version -2 would become version -1, and so on.

Note Files with STORHOUSE organization do not have file versions. For more information, see "File Organization" on page 3-5.

File Revisions

Whenever you change the contents of a file version, StorHouse identifies the revised contents with a unique *revision number*. You can change the contents of a file version by opening it; changing, deleting, or adding records; and closing it.

Revision numbers range from 1 through 65,535. StorHouse assigns revision number 1 to a file version when it is created and increments the number by one each time the contents of the file version are changed.

You can also identify revisions using *relative revision numbers*. Relative revision numbers range from 0, the current revision, through -65,534, the oldest revision.

Note Files with SEQUENTIAL or STORHOUSE organization cannot be revised. See the section "File Organization" on page 3-5.

File Structures

StorHouse files consist of extents, frames, and records. The following sections describe these structures.

File Extents

A *file extent* is a collection of file data that StorHouse treats as a unit. A StorHouse file consists of one or more extents. StorHouse creates and uses extents as needed to support user file operations.

When a file version is created, written, or updated, extents are added to the file. Each extent is identified by an *extent sequence number*. The system assigns sequence number 1,000,000 to the first extent of a file version and increments the sequence number by one for each extent that is added.

StorHouse also assigns its system identifier to each extent that it creates. This and the extent creation date/time help identify differences between files modified at different StorHouse systems or at different times in the same StorHouse system.

StorHouse specifically identifies the last extent of a revision. If one or more extents are missing before the last extent, the system identifies the file version as a *partial* file. If the last known extent of a file does not have the last extent flag, the system identifies the file as a *truncated* file.

Each extent must be no larger than 2 GB and fit on a single volume side. Thus, the maximum size of an extent is limited by the size of a volume side. The minimum size of an extent is 0 bytes.

File extents consist of a series of structures called frames.

Frames

A *frame* is a unit of data and/or control information that can be transferred between processors and storage devices without reformatting. Each frame can have an error detection code and has a header identifying the frame and its internal structure. StorHouse creates and uses frames as needed to support user file operations.

Special access control frames are used only to communicate access control information across the network and are not stored in the StorHouse system.

A frame must be entirely contained within a single file extent. The minimum size of a frame is the minimum size of a frame header plus the trailer, which may contain an error detection code. The maximum size of a frame is 31,744 bytes. All but the last frame in an extent must be 31,744 bytes in length. The last frame can be shorter.

A frame can contain one or more records or segments of records.

Records

A *record* is a unit of user data or file control information. A file's creator determines the contents of the file's data records. The file organization determines what control information, if any, is stored in control records.

Record lengths can vary. A record can:

- Begin and end in one frame
- Begin in one frame and end in the next sequential frame
- Begin in one frame, completely fill one or more of the following frames, and end in the last frame in a sequence.

The part of a record contained within a single frame is called a record *segment*. Each record segment has a *header*, which defines the segment.

A record must be contained within a single extent. Thus, the maximum size of a record is limited by the maximum size of an extent. The minimum size of a record's data is zero bytes, but a zero-length record still has a record header.

Control Record Formats

Control records contain information that supports revision control and data access. StorHouse creates and uses all control records that are stored as part of a file.

Data Record Formats

Some data record formats are specific to one host type, while others are transportable between unlike hosts. *Transportable ASCII data records* consist of 8-bit ASCII character data. A *transportable binary data record* consists of a bit stream; it does not preserve characteristics of the data, such as word alignment and data field formats. The host file transfer interfaces define *host-dependent data record* formats.

IBM MVS hosts require the IBM Disk File Transfer Option and IBM's Data Facility Data Set Services (DFDSS) utility to transfer datasets with host-dependent record formats. See the *Command Language Reference Manual* for more information about DFDSS.

File Organization

A StorHouse file has one of five organizations:

- SEQUENTIAL
- RECORD
- KEYED
- KEYSEQUENTIAL
- STORHOUSE.

The StorHouse software supports SEQUENTIAL files as the basic file organization.

RECORD, KEYED, and KEYSEQUENTIAL files are Virtual Record Access Manager (VRAM) files supported by the StorHouse VRAM software. VRAM Relative Record Access (RRA) software supports RECORD files. VRAM Keyed Record Access (KRA) software supports KEYED and KEYSEQUENTIAL files.

STORHOUSE files are created by the StorHouse/RM software, which is available as an option on StorHouse.

SEQUENTIAL Files

For a SEQUENTIAL file, StorHouse stores records in the order in which you write them. StorHouse does not maintain control information that allows random access of records.

You must read or write records sequentially, from the beginning of the file to the end. You cannot append or update individual records after the file is closed.

RECORD Files

For RECORD files, StorHouse stores records in the order in which you write them initially, but stores updated records separately.

Each record is identified by a *relative record number*. Relative record numbers begin at 1 and are assigned sequentially by StorHouse as you write records into the file. StorHouse assigns update records the same relative record number as the record being updated.

StorHouse saves control information that allows random access of records by relative record number. You can read and update records in sequential or random order. You can append new records to the end of the file but cannot insert records between existing records.

KEYED Files

A KEYED file has all of the characteristics and capabilities of a RECORD file. In addition, a KEYED file allows you to read records by key values. A *key* is a named data field. You can define up to 31 keys for a KEYED file.

The VRAM_KEYED system parameter controls the creation of VRAM KEYED files by users. If this parameter is TRUE, users can create VRAM KEYED files; otherwise, users are not allowed to create this type of file. (The VRAM KEYED system parameter does not control the creation of VRAM KEYSEQUENTIAL files.)

If the VRAM_KEYED system parameter is TRUE, you can define internal or external keys:

- For *internal* keys, each data record can have a value for each key. Internal keys are fields that are contained within each data record. A key definition specifies which bytes of each record form the value of the key.

- For *external* keys, you specify key values in a key data record, which you supply along with a regular data record when writing records into the file. External keys are located in special key records that are associated with separate data records. A key definition specifies which bytes of each key data record form the value of the key.

StorHouse maintains a key data base that maps keys to relative record numbers for a file. Internal key values can be changed. External key values cannot be changed because the record containing the key is not accessible.

KEYSEQUENTIAL Files

A KEYSEQUENTIAL file has the characteristics and capabilities of a KEYED file, with the following restrictions:

- You can define only one key.
- The key must be an internal key.
- When updating a record, you cannot change the key value.
- You cannot specify duplicate key values.
- When writing records into the file, you must write the records in ascending key value order.

In addition, the system does not create a key data base for the file. It stores key values with relative record numbers.

STORHOUSE Files

The optional FileTek StorHouse/RM software creates and uses STORHOUSE files to support relational database capabilities on the StorHouse system. STORHOUSE files have standard frame and record structures, but users cannot access STORHOUSE files directly through the Interactive or Callable Interfaces. Users must submit Structured Query Language (SQL) statements to StorHouse/RM to load and access database data. The StorHouse/RM software determines the contents of STORHOUSE files and file records. For more information on StorHouse/RM and STORHOUSE files, refer to the *StorHouse/RM Administration Guide*.

File and Record Format Identification

StorHouse identifies file and data record formats using the file type, the file system type, the host type, and file organization. The file organization shows that a file is either a SEQUENTIAL, RECORD, KEYED, KEYSEQUENTIAL, or STORHOUSE file. RECORD, KEYED, and KEYSEQUENTIAL file organizations are also called VRAM files and support random record access.

The interpretation of file system type and host type depends on the value of file type. The use of host type depends on the value of file system type. The values for file type, file system type, and host type are shown in Table 3-1. The values are displayed in decimal format.

Table 3-1: File, File System, and Host Type Information

File Type	File System Type	Host Type	Description
01			Standard, StorHouse Framed File
	03	17	IBM MVS DFDSS Unload
	04	05	DEC VAX/VMS RMS Block Mode
	65	n/a	Transportable ASCII Character-stream
	66	n/a	Transportable Binary Records
	67	33	UNIX Binary Byte Stream

Transportable Formats

All VRAM files and some SEQUENTIAL files have either transportable ASCII character-stream data records or transportable binary data record formats. STORHOUSE files have transportable binary data record formats. Files copied to StorHouse in SEQUENTIAL transportable format are stored as sequential records. The only file characteristics information preserved is:

- The length of the longest record
- Whether the file had fixed-length records
- The print control characteristics (none, ANSI carriage control, printer machine-level control).

File Extent Types

Files consist of *extents*, which contain user data records and/or control records. StorHouse supports the following types of extents:

Table 3-2: File Extent Types

Extent Type	For File Organization	Contains
Data (D)	SEQUENTIAL	User data records and sometimes a control record as the first record.
	VRAM (All types)	User data records and a Frame Pointer Table (FPT). The FPT is a set of control records that map user record numbers to frames within the data extent. The frames contain the user data records.
	STORHOUSE (All types)	User and/or control data for a STORHOUSE database file.

Table 3-2: File Extent Types (continued)

Extent Type	For File Organization	Contains
Definitions (DF)	VRAM (All types) STORHOUSE (All types)	Information that defines a revision of a VRAM file version. Information that is necessary to retrieve data from a STORHOUSE file.
Change (C)	VRAM (All types)	Updated user data records for one or more data extents. A change extent contains only those records that were updated to produce the new revision. Previously updated records are not duplicated in each C extent.
Key Data Base (K)	VRAM KEYED	Indexes that map key values to relative record numbers.
Map (M)	STORHOUSE (Hash and value index files)	The high-level index for STORHOUSE hash index files and STORHOUSE value index files. There is one map extent for each hash index file and one map extent for each value index file. StorHouse/RM always reads these extents first when performing lookups that use hash indexes or value indexes.
Update (U)	VRAM (All types)	All updated user data records for a VRAM data extent for all revisions of a file.

Relation of File Extents to Revisions

StorHouse creates file extents for all file organizations. Some file organizations have different extents than other file organizations. This section explains the extents that compose a file revision for each file organization.

StorHouse supports three types of file organizations:

- SEQUENTIAL
- VRAM (which includes RECORD, KEYED, and KEYSEQUENTIAL)
- STORHOUSE.

These file organizations can have the following types of file extents:

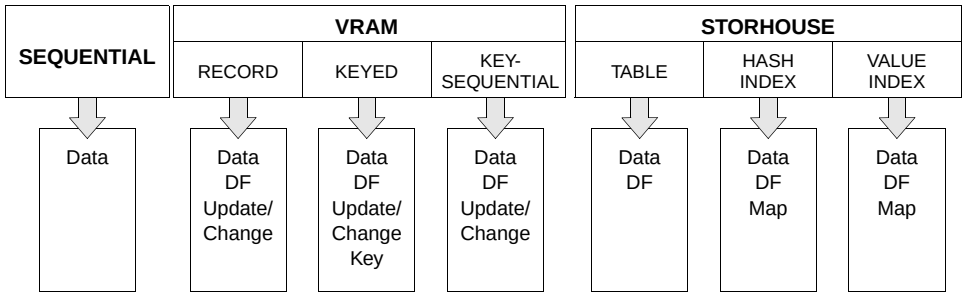

Figure 3-1: File Organizations and Their File Extents

SEQUENTIAL Files

Each version of a SEQUENTIAL file consists of one data extent. The data extent is created when you write data records into a SEQUENTIAL file version for the first and only time. After you close a file version (complete writing records to the file), you cannot reopen that file to write or append additional records. Therefore, each SEQUENTIAL file version contains only one revision.

VRAM Files

Each time a VRAM (RECORD, KEYED, or KEYSEQUENTIAL) file version is opened, modified, and successfully checkpointed or successfully closed, the system creates at least two file extents (a data and a DF extent). The extents created between the open and close become the current revision for the file version.

For VRAM files, StorHouse creates extents for file revisions as follows:

- Data extent – Created when you open a VRAM file in APPEND mode, write new data records, and close or checkpoint the file successfully. This extent is also created when you open a file with the CREATE-OPEN or LSMCO function and close the file successfully.

- DF extent – Created when you open a VRAM file in APPEND or UPDATE mode, write new data records, change data records, or delete data records, and then close or checkpoint the file successfully. This extent is also created when you open a file with the CREATE-OPEN or LSMCO function and close the file successfully.

- K extent – Created when StorHouse stores or updates key values for a KEYED file. StorHouse stores new key values when you append records. StorHouse may have to update key values when you update records. A K extent is also produced when you issue a CHECKPOINT function during an append operation for a KEYED file.

- U or C extent – Created when you update data records in a VRAM file. If the VRAM_UPDATE system parameter is 0, StorHouse creates U extents. If the parameter is 1, StorHouse creates C extents.

A RECORD file contains one DF extent for each revision or checkpoint of the file. If you have written data into the file, it contains one or more data extents. If you have updated records in the file, it contains one or more U or C extents. Two or more revisions of a RECORD file may share some or all of the same data extents and/or U/C extents.

A KEYED file contains the same extents as a RECORD file plus a K extent for each revision of the file where key values were added or changed. Two or more revisions of a KEYED file may share some or all of the same data, U/C, and/or K extents.

A KEYSEQUENTIAL file contains the same types of extents as a RECORD file.

STORHOUSE Files

Before you load data to a StorHouse database, you must issue the appropriate SQL statements to create tables and their associated indexes. Then, when you start your load, StorHouse/RM creates the necessary files to satisfy your table, hash, and value index definitions.

Unlike VRAM file processing, you cannot issue StorHouse commands or Callable Interface functions to open, access, and close STORHOUSE files. Once StorHouse/RM closes a file, you cannot reopen it to append new records or update existing ones.

StorHouse/RM files contain the following types of extents:

- Table files have one or more Data and DF extent pairs per table file.
- Value index files have one or more Data and DF extent pairs plus one Map extent per value index file. The Map extent has an associated DF extent.
- Hash index files have one or more Data and DF extent pairs plus one Map extent per hash index file. The Map extent has an associated DF extent.

Initially, StorHouse/RM creates a Data, DF, and Map extent (when you have defined hash or value indexes for your table) when it begins loading data.

You control when StorHouse/RM creates new data extents (and their corresponding DF extents) by setting values for the following StorHouse system parameters:

System Parameter	Definition
SQL_MAX_EXT_DATA	For table files, specifies the maximum data extent size (in megabytes) that StorHouse/RM will write to StorHouse. StorHouse/RM checkpoints each data extent when it reaches the maximum size and then creates a new one. StorHouse/RM can recover data extents to the last successful checkpoint.
SQL_MAX_EXT_HASH	For hash index files, specifies the maximum data extent size (in megabytes) that StorHouse/RM will write to StorHouse. StorHouse/RM checkpoints each hash index data extent when it reaches the maximum size and then creates a new one. StorHouse/RM can recover index data extents to the last successful checkpoint. FileTek recommends that you try to contain a hash index in one data extent. (Keep in mind, however, that the index must fit on the target device.)
SQL_MAX_EXT_VALUE	For value index files, specifies the maximum data extent size (in megabytes) that StorHouse/RM will write to StorHouse. StorHouse/RM checkpoints each value index data extent when it reaches the maximum size and then creates a new one. StorHouse/RM can recover index data extents to the last successful checkpoint. FileTek recommends that you try to contain a value index in one data extent. (Keep in mind, however, that the index must fit on the target device.)

For example, if you set SQL_MAX_EXT_DATA to 300, StorHouse/RM writes a new data extent each time the current data extent reaches a size of 300 MB.

The settings for these three system parameters are system-wide. That is, you cannot set them for individual table, hash index, or value index files.

File Directories

The StorHouse file directories identify, locate, and maintain statistics for files stored in StorHouse. A directory that is accessible to users can contain only one copy of a file version. Three directories are accessible to users:

- Primary
- Backup
- Archive.

In addition, StorHouse maintains a deleted file directory that contains file versions that have been deleted from the other directories and are awaiting removal.

Primary Directory

The *primary* directory contains information for primary copies of files. You normally access the primary directory. Users on hosts can read and write to primary files directly, using StorHouse host interfaces.

Backup Directory

The *backup* directory contains information on backups of primary files. A *backup* is a duplicate copy of a primary and is identified by the same group name, file name, file identifier, and revision number as the primary copy. Note that a primary is considered backed up only if the backup has the same *revision number* as the primary. The *relative version number* of a backup may be different from that of its primary, however.

The system limits access to files in this directory to a few commands and to data duplexing operations (see "User File Duplexing" on page 5-4 for information on data duplexing). For the most part, these commands create backups from primaries or primaries from backups.

Archive Directory

The *archive* directory contains information for archive copies of primary files. Normally, these archive copies are exported from the StorHouse system for storage at a different site.

The system limits access to files in this directory to a few commands and to data duplexing operations (see "User File Duplexing" on page 5-4 for information on data duplexing). For the most part, these commands create archives from primaries or primaries from archives.

Directory Information

StorHouse records information for file access groups, files, versions, and extents that are located in each directory. The following sections describe StorHouse directory information.

Group Directory Information

StorHouse records the following information for a file access group:

- Group name
- Group passwords.

Users with the appropriate authority can display and change directory information for groups using StorHouse Command Language commands. Refer to the *Command Language Reference Manual* for more information on individual commands.

File Directory Information

StorHouse records the following information for a file name in the primary directory:

- Group name
- File name
- File passwords
- LIMIT attribute.

Users with the appropriate authority can display and change directory information for files using StorHouse Command Language commands. Refer to the *Command Language Reference Manual* for more information on individual commands.

Version Directory Information

StorHouse records the following information for a file version in the primary directory:

- Relative version number
- File identifier (system identifier and file number)
- Revision number
- Directory record number
- Size of file
- Access count
- Date/time information
- ATF attribute
- VTF attribute
- Volume set
- File set
- File descriptor flags
- File status
- File type (see Table 3-1 on page 3-8)
- Host file system (see Table 3-1 on page 3-8)
- Host type (see Table 3-1 on page 3-8)
- File organization
- Number of extents in file
- Framing information
- Maximum record length
- Number of records in file
- Data record attributes
- Retention
- Replication class.

The date and time information includes the following:

- Primary's creation
- When file data was last modified
- When directory entry was last modified
- Last access
- Latest backup or NONE.

The file descriptor flags provide the backup and accessibility status of the file version:

- Archived
- Backed up
- Being cataloged
- Named file
- Copy/transfer pending
- NOBACKUP attribute
- Hardware disabled
- Software disabled
- Deleted
- Replicated.

The file status indicates if the file version has a complete sequence of extents:

- Complete
- Partial
- Truncated.

The file organization must be one of the following:

- SEQUENTIAL
- RECORD (VRAM)
- KEYSEQUENTIAL (VRAM)
- KEYED (VRAM)
- VRAM (RECORD, KEYSEQUENTIAL, or KEYED file created before Release 3.0)
- STORHOUSE.

The framing information includes:

- Frame structure version
- EDC type
- Bytes per frame
- Frame header size (in bytes)
- Record header size
- Host data unit size.

The data record attributes (for transportable format files) can be:

- Print format with machine carriage control
- Print format with ANSI carriage control

- Fixed record length.

Extent Directory Information

StorHouse records the following information for a file version extent:

- Extent sequence number
- Extent system identifier
- Revision number of the file to which the extent belongs
- Extent creation date
- Date extent written
- Extent size
- Extent location (vid)
- Extent level (F, L, or S)
- Extent migration factor (NONE if not in performance buffer)
- Extent status
- Extent retention date (“none” if not in performance buffer).

The extent status indicates the current type and accessibility of the extent:

- Last extent of revision or checkpoint
- New extent
- Has copy in performance buffer
- Not copied to performance buffer yet
- Extent created as part of update
- Disabled
- Write-back disabled.

Directory Functions

StorHouse supports the following directory functions:

- CHECKPOINT – copies all critical system files to a backup medium.
- CREATE – creates a new file version entry in the directory.
- DELETE – marks a file version as deleted in the directory.
- EXTRACT – copies StorHouse directory information to directory extraction files.
- RECOVER – recovers system information on user files from a backup medium.
- REMOVE – removes the directory entry for a file version.
- RESTORE – reads extraction files containing directory information from one or more source StorHouse systems and inserts the information into a destination StorHouse system.
- SET – modifies a file version’s directory entry.
- SHOW – displays information from a file version’s directory entry.

File and Record Access

StorHouse provides file and record access capabilities for users to access SEQUENTIAL and VRAM files. These include basic file functions, access methods, access modes, and record functions. StorHouse provides file transfer functions for SEQUENTIAL, VRAM, and STORHOUSE files.

Basic File Functions

StorHouse provides you with the basic file access functions described in Table 3-3:

Table 3-3: File Access Functions

Function	Description
CREATE-OPEN	Creates a VRAM file and then obtains system and file resources to make the file available for access.
OPEN-SEQ (also OPEN)	Obtains system and file resources to make a SEQUENTIAL file available for access.
OPEN-VRAM	Obtains system and file resources to make a VRAM file available for access.
CHECKPOINT	Saves new data written to a VRAM file for possible use in restart after abnormal termination of append.
CLOSE	Frees system and file resources, saves new or changed data, and makes a previously opened file unavailable to a user for access.

Access Methods

The *access method* or methods that are used to open a file determine the order in which you can read records. StorHouse supports three file access methods, described in Table 3-4:

Table 3-4: File Access Methods

Method	Description
SEQUENTIAL	Reads records sequentially from a SEQUENTIAL, RECORD, KEYED, or KEYSEQUENTIAL file.
RECORD	Reads records randomly by relative record number from a RECORD, KEYED, or KEYSEQUENTIAL file.
KEYED	Reads records randomly by key value from a KEYED or KEYSEQUENTIAL file.

You can open a file for any one method of access, any combination of two methods, or all three methods based on the file organization.

Access Modes

The *access mode* that is used to open a file determines the record functions you can perform on that file. You can open a file using one access mode in the OPEN function.

StorHouse supports four file access modes, described in Table 3-5:

Table 3-5: File Access Modes

Mode	Description
READ	Reads records from a SEQUENTIAL, RECORD, KEYED, or KEYSEQUENTIAL file.
WRITE	Writes records to a SEQUENTIAL file.
APPEND	Writes new records to a RECORD, KEYED, or KEYSEQUENTIAL file.
UPDATE	Reads, changes, and deletes records in a RECORD, KEYED, or KEYSEQUENTIAL file.

If you open a file using WRITE, APPEND, or UPDATE mode and write data into that file or otherwise modify it, the function creates a new revision of the file.

Record Functions

StorHouse provides you with the basic record access functions described in Table 3-6:

Table 3-6: Record Access Functions

Function	Description
READ	Reads the next sequential record from a SEQUENTIAL file. Requires READ mode and SEQUENTIAL access method.
READ-SEQUENTIAL	Reads the next sequential record from a RECORD, KEYED, or KEYSEQUENTIAL file. Requires READ or UPDATE mode and SEQUENTIAL access method.
READ-RECORD	Reads a random record by relative record number from a RECORD, KEYED, or KEYSEQUENTIAL file. Requires READ or UPDATE mode and RECORD access method.
READ-KEYED	Reads a random record by key value from a KEYED or KEYSEQUENTIAL file. The key must be one of those defined by you (not the relative record number). Requires READ or UPDATE mode and KEYED access method.

Table 3-6: Record Access Functions (continued)

Function	Description
READ-NEXT-KEY	Reads the next record in a key sequence from a KEYED or KEYSEQUENTIAL file. This function must be preceded by a successful READ-KEYED or READ-NEXT-KEY function. Requires READ or UPDATE mode and KEYED access method.
WRITE	Writes the next sequential record to a SEQUENTIAL, RECORD, KEYED, or KEYSEQUENTIAL file. Requires WRITE or APPEND mode and any access method.
WRITE-KEY	Writes the next sequential record to a KEYED file and may also supply a key data record containing external key values. Requires WRITE or APPEND mode and any access method.
DELETE	Marks the last record read from a RECORD, KEYED, or KEYSEQUENTIAL file as deleted. The function must be preceded by a successful READ-SEQUENTIAL, READ-RECORD, READ-KEYED, or READ-NEXT-KEY function. Requires UPDATE mode and any access method.
CHANGE	Writes an updated record for the last record read from a RECORD, KEYED, or KEYSEQUENTIAL file. The function must be preceded by a successful READ-SEQUENTIAL, READ-RECORD, READ-KEYED, or READ-NEXT-KEY function. Requires UPDATE mode and any access method.

File Transfer Functions

File transfer functions combine user and StorHouse internal file and record functions to transfer an entire file version from one place to another. StorHouse provides you with the file transfer functions described in Table 3-7:

Table 3-7: File Transfer Functions

Function	Description
PUT	Creates a SEQUENTIAL file (if it does not exist), accepts an OPEN-SEQ in WRITE mode from the host, accepts WRITE functions from a host, and accepts a CLOSE function from the host. This function always creates a new version of the file.
GET	Accepts from the host an OPEN-SEQ of a SEQUENTIAL file in READ mode, accepts READ functions from a host, accepts a CLOSE function from the host.
XFER	Opens a SEQUENTIAL, VRAM, or STORHOUSE file in READ mode (uses internal StorHouse file open), duplicates each extent in a new location (uses internal StorHouse read and write functions), and closes the file. No new version is created because the file version is not modified. This function is incorporated in a number of user commands; it is not accessible through the user interface.

Software Disabled File Version

When a VRAM file version is opened in mode APPEND or UPDATE, StorHouse marks it as software disabled. When the file version is successfully closed, StorHouse marks it as software enabled. Opening a VRAM file in READ mode or a STORHOUSE file for reading does not software disable a file.

StorHouse leaves an open file in the software disabled state whenever:

- You close the VRAM file with the abort flag set.
- A user application terminates while appending or updating data in a VRAM file.
- The network disconnects.
- The host crashes.
- StorHouse crashes.
- StorHouse forces the file close as a result of an internal error.

The revision that was being created when the file was software disabled is the current revision of a software-disabled file version. A user application can access a checkpointed, software-disabled VRAM file by opening the current revision and specifying a valid checkpoint number to restart an append operation. An application can also open an older revision of the file.

File Access Control

Users with the appropriate authority can control access to file access groups by assigning access group passwords. Also, authorized users can control access to files by assigning file passwords and by locking file versions.

Access Group Passwords and Access Types

Each access group can have a *read* password, a *write* password, and a *delete* password. An access group password can be null or contain up to eight characters, and can consist of the following ASCII characters: A-Z (uppercase), 0-9, _ (underscore), and $ (dollar sign).

You must specify the correct read password to have read access to the group. Read access allows you to obtain read access to files in the group and to display group information. A null read password allows any user to have read access without specifying a read password.

You must specify the correct write password to have write access to the group. Write access allows you to obtain write access to files in the group. A null write password allows any user to have write access without specifying a write password.

You must specify the correct delete password to have delete access to the group. Delete access allows you to obtain delete access to files in the group, to change the group's passwords, and to delete the group. A null delete password allows any user to have delete access without specifying a delete password.

If you specify a non-null read, write, or delete password to gain access to a group, and the group itself has a null password in the corresponding password position, StorHouse returns an invalid password error and does not give you access to the group.

File Passwords and Access Types

File passwords control user access to individual files. A *read* password, a *write* password, and a *delete* password can be assigned to each file. One set of passwords controls access to all versions of a file.

A file password can be null or contain up to eight characters, and can consist of the following characters: A-Z (uppercase), 0-9, _ (underscore), and $ (dollar sign).

You must specify the correct read password to have read access to a file. Read access allows you to get a copy of the file from StorHouse and display file directory information. A null read password allows any user to have read access without specifying a read password.

You must specify the correct write password to have write access to a file. Write access allows you to put a new version of the file into the StorHouse system. A null write password allows any user to have write access without specifying a write password.

You must specify the correct delete password to have delete access to a file. Delete access allows you to delete the file from StorHouse, to change file passwords, and to change file attributes. A null delete password allows any user to have delete access without specifying a delete password.

If you specify a non-null read, write, or delete password to gain access to a file, and the file itself has a null password in the corresponding password position, StorHouse returns an invalid password error and does not give you access to the file.

File Locks

Users with the appropriate authority can lock a file version to prevent users of other accounts from reading, writing, or deleting it until they unlock it. This is an *explicit* lock.

An explicit lock locks a file across all directories. In other words, if you specify a primary file version in a LOCK command, any archive or backup copies are also locked.

StorHouse locks and unlocks a file automatically during update, append, copy, or various directory operations. This is an *implicit* lock. There are two kinds of implicit locks:

- *Read-lock* – If StorHouse reads a file only, it *read-locks* that file. All users can read a read-locked file.
- *Write-lock* – If StorHouse either writes or deletes a file, it *write-locks* that file. Users cannot access a write-locked file. Users cannot unlock implicitly locked files.

Chapter

4

Storage Allocation

This chapter discusses logical structures for physical storage allocation and control. It also defines naming conventions, attributes, and access characteristics for volumes, volume sets, file sets, and file copies.

Volumes

A *logical* volume is the portion of a *physical* volume that can be accessed while it is mounted in a drive and is treated as a mountable unit of storage. A logical volume consists of one or more physical volume surfaces and is identified by a volume identification code, which may include a side indicator.

For example, one media type could represent optical volumes that are double-sided but only one side is accessed at a time. To StorHouse, these volumes have two logical sides. In contrast, another media type could represent optical volumes that are double-sided but both sides can be accessed simultaneously. To StorHouse, these volumes have one logical side.

Each volume used by StorHouse must have a physical format and a structure and labeling format for recording volume and file data on the volume. The physical format determines where and how data is recorded on the physical medium. Volume labels, file labels, and file data contain the data to be recorded. Different media and recording types can have different formats.

Volume Label Records

A *volume label record* is an internal block of data that StorHouse uses to identify removable volumes across customer sites. This record includes the volume_label subfield from the *volume identification code* (vid), the StorHouse *system identifier* (the value of the SYSTEM_ID system parameter), and the *site identifier* (the value of the SITE_ID system parameter).

Before StorHouse encodes a volume label record on a removable volume, it directs the operator to write the vid on the volume's protective casing (cartridge) or, for tape volumes, to ensure that the tape has an external bar code label. (For tapes, StorHouse uses the bar code as the volume label in the vid.)

Volume Table of Contents

Each volume contains a *Volume Table of Contents* (VTOC) or a logical equivalent to the VTOC. This is a directory of the files stored on the volume. The directory includes StorHouse file labels, one for each file extent on the volume. Each file label contains StorHouse directory information for the file and pointers to the file extent data on the volume.

Volume Formats

This section discusses volume formats for magnetic disk, optical disk, and magnetic tape media.

Magnetic Disk

Magnetic disk volume formats are defined by StorHouse. Although StorHouse records StorHouse-defined volume label records on magnetic disk volumes, it does not maintain entries for magnetic disk volumes in its volume directories. Thus, StorHouse cannot locate magnetic disk volumes if you attempt to use one in a command.

Optical Disk

Manufacturers physically format optical disk volumes. If two volumes have the same media type and recording type, they have the same physical format and can be used in the same drive. For media type OA, second-generation (recording type D) optical drives can read first-generation (recording type B) optical media only if the hardware is specially equipped with this option. The system requires special boards to use this feature. Some drives support multiple recording types within a media type. Call your FileTek customer support representative for information.

StorHouse labels the volumes when they are installed in the system, and records file labels on a volume when it writes file data on the volume.

Magnetic Tape

Magnetic tape drives define the physical format of magnetic tape volumes. If two volumes have the same media type and recording type, they have the same physical format and can be used in the same drive. Some drives support multiple recording types but may be limited to read-only functions for selected recording types.

StorHouse internally labels the volumes when they are installed in the system, and records file labels on a volume when it writes file data on the volume. The system also saves tape usage information on the volume for use in long-term tape maintenance.

Residence

Normally, the system maintains each physical volume in a single level of storage (fixed, removable, library, or shelf) until it is migrated to another level or removed from the StorHouse system. The level of storage in which the system maintains a volume is the *residence* of that volume.

Volume HOLD Attribute

The *HOLD attribute* for a volume helps determine the order in which volumes are migrated from a library device to shelf storage. The attribute can have a value of HOLD or NOHOLD. For any one library device, StorHouse migrates volumes with a HOLD value of NOHOLD before migrating volumes with a HOLD value of HOLD. StorHouse migrates the least recently accessed volume when selecting a volume from a set of volumes that all have the same HOLD value.

The system uses the HOLD attribute only for migrations of volumes out of library devices. If a volume is stored on shelf, the system ignores the HOLD attribute; it does not attempt to migrate volumes back into the library based on the HOLD attribute. However, if the system directs the operator to load a shelf volume back into the library to satisfy a user or operator request, the system will use the volume HOLD attribute when considering that volume for any subsequent migrations out of the library.

If you change the HOLD attribute for a volume, StorHouse uses the new value when considering the volume for any subsequent migrations of volumes out of the library device in which the volume resides.

Deactivation Time and Deactivated Volumes

The *deactivation time* for a volume side indicates when StorHouse can mark the volume side as *deactivated*. The system allocates no additional space for files on a deactivated side, but it can read files from the side.

The deactivation time for a volume side is specified as a number of days. The deactivation time is ignored by the system until file space is allocated on the volume side. The system marks a volume side as deactivated after the specified number of days has passed since the *first* file space was allocated on the side. The deactivation time has no effect if it is given a value of zero days.

Cycle Time and Cycled Volumes

The *cycle time* for a volume determines the side of the volume on which the system can allocate space for files. The cycle time is specified as a number of days.

For double-sided logical volumes (such as most optical disks), when a volume is assigned a non-zero cycle time (either when it is added to a volume set or at a later time through the SET VOLUME command), the system immediately deactivates the second side of the volume. The system allocates no space for files on the second side while it is deactivated. The system activates the second side and deactivates the first side when the specified number of days has passed since the *last* (most recent) file space was allocated on the first side. The cycle time has no effect if it is given a value of zero days. In this case, both sides of a volume are available for use when the cycle time is assigned to the volume.

For single-sided logical volumes (such as magnetic tape), the system takes no immediate action when the volume is assigned a non-zero cycle time. The system deactivates the volume side when the specified number of days has passed since the *last* (most recent) file space was allocated on the side. The cycle time has no effect if it is given a value of zero days.

Expiration Time and Expired Volumes

The *expiration time* for a volume side indicates when StorHouse can mark a volume side as *expired*. Generally, when a volume side is marked expired, the files on that side are no longer needed in the system. If the only side of a single-sided volume or both sides of a double-sided volume are marked expired, the volume is a candidate for removal from the StorHouse system.

The expiration time is specified as a number of days. The expiration time for a volume side is ignored by the system until the system allocates file space on the volume side. The system marks a volume side as deactivated and expired when the specified number of days has passed since the *last* (most recent) file space was allocated on the side. The system no longer allocates space for files on that side but will continue to read files on the side. The expiration time has no effect if it is given a value of zero days.

Writelocked Volumes

StorHouse can mark a volume side as *writelocked*. For example, the system writelocks a volume because of excessive errors on the volume or because it encountered the end of a magnetic tape volume before it was expected. A premature end-of-tape may be caused by an unusually short tape, excessive media errors, or an extremely large number of very small files.

The system administrator can writelock a volume, if necessary, by using the SET VOLUME /WRITELOCK command. Writelocking prevents any more files from being written to the volume. This modifier is useful if the administrator wants to move a partially full volume offsite and does not want the system to attempt to write any more information to it.

The system reads file extents on a writelocked volume side, but it will not write file extents onto the side. A file with an extent on a writelocked volume side can be deleted and removed from the directory, but deleted file labels cannot be written on that volume side and deleted data cannot be erased on that volume side.

The system operator can unwritelock an optical disk using either the SET VOLUME /CLEANED command or the SET VOLUME /UNWRITELOCK command, or a tape using the SET VOLUME /UNWRITELOCK command. However, the system can allocate space on the volume only if unused space remains.

Disabled Volumes

A *disabled volume* is a volume that a system operator has marked as disabled using the StorHouse Command Language SET VOLUME /DISABLED command. An operator might disable a volume because it is broken, unreadable, or misplaced.

If the primary copy of a file resides on a disabled volume and you have set up the system for file duplexing, StorHouse can automatically read the duplex (backup or archive) copy of the file. For more information on file duplexing, see “User File Duplexing” on page 5-4.

A disabled volume can be recovered; in other words, a replacement volume can be created from backup or archive copies of the primary files. The system administrator can recover a disabled volume using the RECOVER VOLUME command. For more information on recovering volumes, see “Volume Recovery” on page 5-26.

Memos for Volumes

A user with proper authorization can specify a comment for a volume (for example, information about the volume contents or the location of a volume that has been moved out of the library device) by using the /MEMO parameter modifier on the MOVE VOLUME and SET VOLUME commands. The SHOW VOLUME /MEMO command displays the comment. The SHOW VOLUME /MEMO command can also display all volumes that have a specific memo assigned.

A user can update the comment anytime using the SET VOLUME command. The comment applies to the entire volume, not to a volume side. The information specified on the /MEMO parameter modifier is included in the StorHouse system files, but is not written to the optical volume or tape. The comment also displays in selected operator messages.

Volume Sets

A *volume set* identifies one or more physical volumes that are treated as a logical unit of storage. Volume sets allow you to control the physical grouping of files. Each StorHouse volume, including every level F magnetic disk, must be a member of only one volume set. A volume set is associated with one directory, either the primary, backup, or archive directory. Files stored on a volume set have entries only in the volume set's associated directory.

Volume Set Name

The *volume set name* uniquely identifies a volume set within a StorHouse system. Volume set names can contain 1 to 8 characters and consist of the following ASCII characters: A-Z (uppercase), 0-9, _ (underscore), and $ (dollar sign).

Volume Set Attributes

Volume set attributes determine how the volume set is managed. All volumes in a volume set have the same media type and recording type. The media type and recording type for a volume set are the same as those of its member volumes. The other volume set attributes are described in the following sections.

Size

The size of a volume set is the total of the sizes of its member volumes. The minimum size is 0 bytes (no volumes). You can create a volume set with a size of 0, but the set is not usable unless it is extendible, that is, unless the volume set's maximum size (LIMIT) is greater than 0. If you execute a release function that causes the size to shrink to 0, the system removes the volume set from the directory.

Some volumes may be slightly smaller than the physical volume's maximum size. If a volume set contains one or more of these volumes, the volume set size may not be an integer multiple of the maximum size of a physical volume of the medium.

LIMIT Attribute

The *LIMIT attribute* specifies the maximum size of a volume set. StorHouse does not extend a volume set beyond the value of its LIMIT.

The LIMIT must be greater than or equal to the total size (in other words, the storage capacity) of all volumes in the volume set, and it must be an integer multiple of the physical volume's maximum size.

- If the difference between the LIMIT and the volume set's size is less than the maximum size of a physical volume of the medium, the volume set cannot extend.
- If the LIMIT is greater than the volume set's size, and the difference between the LIMIT and the volume set's size is greater than or equal to the maximum size of a physical volume of the medium, the volume set is extendible. StorHouse adds volumes to the volume set automatically when the volume set needs more storage space.

Residence

The *residence* of a volume set is the residence of all of its member volumes. Removable volumes can be on different levels of storage and different devices; thus, such a volume set can be resident on more than one device and level at the same time.

Library Device and Media Attributes

The *library device attribute* specifies a library device or level F device (a device identification code) that contains the pool of empty volumes (the *free pool*) from which new allocations to the volume set will be taken. Some library devices support multiple recording types and have a free pool for each recording type. The *media attribute* specifies the media type and recording type of the free pool to be used.

Once you have created a volume set, you cannot change its media type or recording type. However, if you can remove a volume set from a StorHouse system by releasing all of its volumes back to the free pool, you can re-create it with a different media specification.

You can change the value of the library device attribute for a volume set to any other library device that supports the same media type and recording type (in other words, the library device has a free pool of volumes with the same media). Removable volumes that are members of the volume set do not have to reside in the library device specified by the library device attribute. They may reside on level S or even in another library device.

Volume Set HOLD Attribute

The *HOLD attribute* for a volume set specifies the value for a volume's HOLD attribute when the volume is added to the volume set. The volume set HOLD attribute can have a value of HOLD or NOHOLD. The volume HOLD attribute helps to determine the order in which volumes are migrated from a library device to shelf storage. For any one library device, StorHouse migrates volumes with a HOLD value of NOHOLD before migrating volumes with a HOLD value of HOLD. StorHouse migrates the least recently accessed volume when selecting a volume from a set of volumes that all have the same HOLD value.

The system parameter VSET_HOLD provides the default value for the HOLD attribute for a volume set when the volume set is created. For more information on the VSET_HOLD system parameter, see the *Command Language Reference Manual*.

The CREATE VSET, SET VSET, and SET VOLUME commands allow you to assign or change the volume set and volume HOLD attribute values. StorHouse displays the value of the HOLD attribute when you enter the SHOW VSET /FULL or SHOW VOLUME /FULL command. The SHOW VOLUME command allows you to select volumes for display based on the value of the HOLD attribute. For more information on these features, see the appropriate command descriptions in the *Command Language Reference Manual*.

If you change the value of the HOLD attribute for a volume set, StorHouse assigns the new value to any volumes added to the volume set after the change, but does not change the attribute for any volumes already in the set. If you change the attribute for a volume, StorHouse uses the new value when considering the volume for any subsequent volume migrations out of the library device in which the volume resides.

Deactivation Time

The volume set *deactivation time* specifies the deactivation time for volume sides in the volume set. StorHouse assigns the volume set deactivation time to a volume side when it is added to the volume set, but the system will ignore the time until space is allocated on the volume side.

Cycle Time

The volume set *cycle time* specifies the cycle time for volumes in the volume set. StorHouse assigns the volume set cycle time to a volume when it is added to the volume set. The cycle time applies only to double-sided logical volumes. It has no effect on single-sided logical volumes. If a non-zero cycle time is set for a single-sided logical volume, StorHouse takes no immediate action. However, the system will deactivate the volume side when the specified number of days has passed since the last (most recent) file space was allocated on the side. The cycle time has no effect if it is given a value of zero days.

Expiration Time

The volume set *expiration time* specifies the expiration time for volume sides in the volume set. StorHouse assigns the volume set expiration time to a volume side when it is added to the volume set, but the system will ignore the time until file space is allocated on the volume side.

Level F Volume Set

At installation, StorHouse is configured with one level F volume set called *MAGDISK*. The name MAGDISK is given by the default value of the StorHouse system parameter PERF_BUF_VSET. MAGDISK initially encompasses all level F storage space.

Level F volume sets are similar to volume sets on other levels of storage, with the following exceptions. Because level F volume sets do not extend automatically, the size of a level F volume set must be increased or decreased using StorHouse Command Language. In addition, level F volume sets cannot be imported, exported, cataloged, or uncataloged.

Note Do not remove (release) MAGDISK when it is empty.

Memos for Volume Sets

A user with proper authorization can specify a comment for each volume in a volume set (for example, information about the volume set's contents or the location of a volume set that has been moved out of the library device) by using the /MEMO parameter modifier on the MOVE VSET and SET VSET commands. The SHOW VSET /MEMO command displays the comment for the first volume in the volume set. The SHOW VSET /MEMO command can also display all volume sets whose first volumes have a specific memo assigned.

A system administrator can update the comment anytime using the SET VSET command. The information specified on the /MEMO parameter modifier is included in the StorHouse system files, but is not written to the optical volume or tape. The comment also displays in selected operator messages.

File Sets

A *file set* is an area of storage within a volume set. Files are stored in file sets. All StorHouse files must be part of a file set.

A file set may span volumes within a volume set; thus, a file set consists of all or part of one or more volume sides in a volume set. The part of a file set that is located on a single volume side is called a *partition*.

A file set is identified by its volume set name and file set name. Figure 4-1 shows an optical volume set with three file sets.

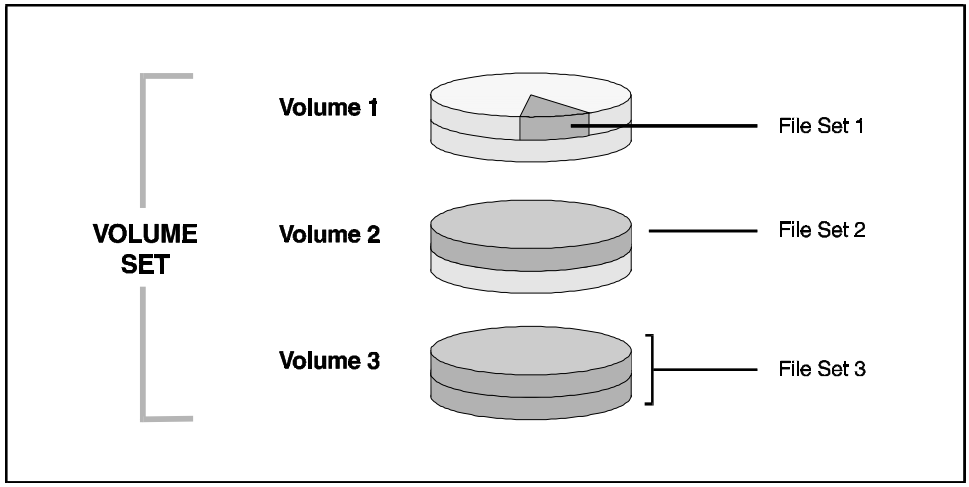

Figure 4-1: Optical Volume Set with Three File Sets

File Set Name

The *file set name* identifies a file set within a volume set. Each file set name must be unique within a volume set. File set names can contain 1 to 8 characters and consist of the following ASCII characters: A-Z (uppercase), 0-9, _ (underscore), and $ (dollar sign).

File Set Attributes

The following *file set attributes* determine how a file set is managed.

- The size, limit, and storage allocation attributes determine how StorHouse allocates file set space on volume sides.
- The staging attribute determines whether files in the file set will be staged to the performance buffer when they are accessed.
- Under certain conditions, the retention attribute specifies the retention period for files in the file set. (See /FORCE_RETENTION on page 4-13.)
- The replication class name attribute specifies the name of the replication class associated with files in the file set.

Size

The size of a file set is the number of bytes the system has allocated to the file set for file storage. A file set's maximum size is the maximum size of its volume set. The minimum size is 0 bytes. You can create a file set with a size of 0, but the set is not usable unless it is extendible, that is, unless the file set's maximum size (LIMIT) is greater than 0. If you execute a release function such that the size shrinks to 0, the system removes the file set from the directory.

LIMIT Attribute

The *LIMIT attribute* specifies the maximum size of a file set. StorHouse will not extend a file set beyond the value of its LIMIT. The LIMIT must be greater than or equal to the size of the file set.

If the LIMIT is equal to the size, the file set cannot extend. If the LIMIT is greater than the size, the file set is extendible. StorHouse can increase the size of the file set automatically, when the file set needs more space. The size of the extension is determined by the size of the needed space.

Storage Allocation Attribute

You can specify a *storage allocation attribute*, either *contiguous* or *noncontiguous*, for the file set. The attribute controls the way the system allocates storage for the file set for initial size allocations and extensions. In all cases, StorHouse rejects an allocation request if the resulting size of the file set would exceed the LIMIT attribute of the file set.

Contiguous Allocation. For file sets, a *contiguous* allocation is an allocation that is contained entirely on one volume side or that consists of entire volume sides with any remainder on one side of a volume. Thus, it uses only as many sides as are required to accommodate the file set size.

When allocating space for a file set with the contiguous attribute, StorHouse allocates entire volume sides unless the LIMIT attribute for the file set restricts an allocation to less than a full side.

Note The term contiguous refers to the relationship of file set data stored on multiple volume sides. It does not refer to the relationship of individual file set bytes across volume sides or on one volume side. StorHouse writes file set data (bytes) in the most efficient way possible, and may not write file set bytes stored on one volume side contiguously (that is, right next to one another).

Noncontiguous Allocation. For a *noncontiguous* allocation, StorHouse allocates the first free space it finds and continues to allocate space on other volume sides until it has allocated the entire size of the file set. Thus, a noncontiguous allocation can be partitioned across more volume sides than are required to accommodate the file set size.

For an extension to a file set, the noncontiguous attribute indicates that the system is to extend to contiguous space (on the same volume side) if it is available, but any available space can be used if contiguous space is not available.

Update Space

File set *update space* is the portion of a file set that is reserved for VRAM file updates (mode=UPDATE). *General space* is storage space to be allocated to SEQUENTIAL files and the portions of VRAM files that are not associated with updates. StorHouse also stores VRAM updates in general space if no update space is available.

The *update attribute* indicates how much of the file set space is to be reserved for update space. The attribute is specified as a percentage of the general space allocated to files. If the value of this attribute is 0 percent, the system reserves no space for updates and uses general space to store any updates.

Staging Attribute

The staging attribute, AUTO_STAGE, controls whether files in a specified level L file set (or their archive or backup copies) are subject to automatic staging when they are accessed. Two system parameters, MIG_REPOP_LOAD and MIG_REPOP_MAX, control whether the auto-staging feature is enabled. MIG_REPOP_LOAD determines the number of stage requests that can be queued for transfer. MIG_REPOP_MAX specifies the maximum file extent size (in bytes) that can be considered for staging. Refer to the *Command Language Reference Manual* for more information on these parameters.

When auto-staging is enabled, StorHouse stages file extents with the AUTO_STAGE attribute to the performance buffer when they are accessed. If a file extent already resides in the performance buffer, it is not recopied. If the performance buffer contains insufficient space, StorHouse does not stage the file nor does it initiate a regular migration to free up space.

If you display file set attributes and AUTO_STAGE is not set, the AUTO_STAGE state does not display. For more information on file staging, see “User File Staging” on page 5-20.

File Set Retention Attribute

File retention can be set at the file and file set level. This section describes file set-level retention. Refer to “User File Retention,” on page 5-7 for more information about file-level retention.

The *file set retention attribute* specifies the time span that files in the file set may not be deleted from StorHouse. It becomes effective at file create time when:

- A user or an application specifies no file retention value or a file retention value of DEFAULT. When this occurs, the file inherits its file set retention attribute.

- The /FORCE_RETENTION modifier on the CREATE FSET or SET FSET command is in effect on the file set. In this case, the file set retention value overrides the file retention value.

Administrators or users with the proper authority can set the file set retention attribute with the /RETENTION modifier on the CREATE FSET and SET FSET commands. CREATE FSET /RETENTION applies to all files in the file set, while SET FSET /RETENTION applies only to new files in the file set.

Valid values for /RETENTION are:

- DEFAULT (the retention is not specified and assumes the default value).
- FOREVER (never allow the file to be deleted).
- ZERO (do not retain the file).
- Number of days (retain the file for the specified number of days).

The maximum retention period is 65,000 days, or approximately, 178 years.

Replication Class Name Attribute

A file is eligible for application-independent replication from a source StorHouse system to a target, or destination, StorHouse system only when it has an assigned replication class. The file set *replication class name attribute* is one way to assign a replication class to a file. A *replication class* provides a named set of information about the StorHouse system that will contain the file replicas.

System administrators or users with the proper authority can associate a file with a replication class by assigning a *replication class name* to the file's resident file set (SET FSET or CREATE FSET command). In this case, at create time, the file inherits its replication class from its file set.

Refer to the section, "User File Replication," on page 5-10 for more information about file replication.

Level F File Sets

At installation, StorHouse is configured with a *performance buffer*, which is the holding area on level F that contains performance copies of files waiting to be backed up to their primary file sets. The performance buffer is configured as the file set $$BUFFER on the level F volume set MAGDISK. The name $$BUFFER is given by the default value of the system parameter PERF_BUF_FSET.

While $$BUFFER can consist of multiple level F volumes, the largest file extent that can be written is limited by the size of one surface (partition). Each level F volume is a discrete entity, just as with optical volumes.

At installation, the performance buffer encompasses the entire available user disk area. Before you can create additional files sets on level F, your system administrator must reduce the size of the performance buffer file set using the StorHouse Command Language. The *System Administrator's Guide* describes how to change the size of the performance buffer.

If you write data to a file in a level F file set other than the performance buffer, StorHouse always writes the data to the destination file set directly and does not use the performance buffer.

File Storage and Statuses

A StorHouse file has one or more versions. StorHouse manages storage for each version independently of any other versions.

File Residence

StorHouse indicates the *residence* of a file version by identifying the volume set and file set on which it is stored or has space allocated. (A file version consists of one or more extents located in a single file set.)

A file version may span multiple volumes in a volume set, but each extent of a file version is resident on a single volume. StorHouse indicates the residence of an extent by identifying the volume on which it is stored.

File Extent Statuses

Table 4-1 lists the StorHouse file extent statuses:

Table 4-1: File Extent Statuses

File Extent Status	Indicates:
BUFFERED	The extent has a copy in the performance buffer.
LAST	The last extent of a revision or checkpoint.
NEW	The extent has not been copied from the performance buffer to its primary file set yet.
NONE	None of the other options.
UPDATE	The extent was created in an update operation.
WRITEBACK_DISABLED	The extent cannot be backed up to its destination file set.

When you write file data to an extent in the performance buffer, the system marks the extent as NEW and as BUFFERED. It also preallocates storage space for the extent in its primary file set. The system removes the NEW file extent status when the new extent is copied to its primary file set. The system removes the BUFFERED status when the extent is removed from the performance buffer.

If the system does not have enough pre-allocated space to write a file extent from the performance buffer to its primary file set (because of media, drive, or system problems during a previous attempt), the system marks the extent as WRITEBACK_DISABLED. The system administrator can use the StorHouse Command Language ENABLE command to enable the extent. This command removes the writeback-disabled status and preallocates additional space for the extent.

File Size

A file version's total size is the sum of the sizes of its extents. Each file version must be contained by its file set.

For append operations, each *extent set* of a file must be no larger than 2 GB or a single volume side, whichever is smaller. For a description of extent sets, see the section "Relation of File Extents to Revisions" on page 3-9.

For update operations, because StorHouse allocates space for each extent separately, the size limitation applies to individual extents rather than to the extent set. If the file is written to level F, each extent must fit on the largest available level F partition in the destination file set.

The following sections describe how to estimate extent sizes and total file sizes. The asterisk (*) is used as the multiplication symbol. All sizes are assumed to be in units of bytes.

SEQUENTIAL File

A SEQUENTIAL file (a non-VRAM file) consists of a single data extent. For the Callable Interface, you must specify the estimated size of the file as an argument of the open function. The PUT command estimates the size of a file and allocates space for it automatically.

To estimate the total size of a SEQUENTIAL file written using the Callable Interface, assume N is the number of records written and A is the average record size. Then use the following formula:

file size = 1.001 * N * (A+5)

The size of a SEQUENTIAL file must be no larger than the largest available partition in the destination file set, which can be no larger than a level F volume or one side of a removable volume.

If you PUT a file into the StorHouse system, the above formula can be used to obtain a rough estimate of the size of the resulting StorHouse file. For host-dependent file formats, StorHouse selects a convenient record size, calculates the number of records based on the host file size, and adds a small header record containing host file information.

For files PUT with the /ASCII or /BINARY option, StorHouse may have to estimate the number of records and average record size. It allocates storage space based on an estimate of the file size, which is almost always larger than the actual file size. Any unused space is released after the file is written. Thus, the maximum file size may be limited to a size that is smaller than the size of a volume side.

VRAM File Size

The total size of a VRAM file is the sum of the sizes of all its extents. The following indicates when extents are created:

- After the CREATE FILE command (or CREATE OPEN or LSMCO callable function), the file consists of a small DF extent.
- For any append operation (mode=APPEND), add a DF extent, a data extent, and (for KEYED files) a K extent whenever a checkpoint is issued or the file is closed successfully.
- For an update operation (mode=UPDATE), add a DF extent. If any records were changed (not just deleted), add an update extent (a U or C extent). For KEYED files where the value of a key was changed in at least one updated record, add a K extent. (K extents always contain all key data for a KEYED file, not just the changed keys.)

The following sections describe how to estimate extent sizes.

VRAM Data Extent

The size of a data extent written during an append operation for a KEYED or RECORD file is given by the following formula, where D is the largest number of bytes of user data that will be written in one extent set, and nrecs_e is the largest number of records in one extent set:

$$\text{data size} = 32{,}000 + (1.006 * (D + (\text{nrecs_e} * 7)))$$

If there are no checkpoints, nrecs_e is the same as the number of records written during the entire append operation.

The size of a data extent written during an append operation for a KEYSEQUENTIAL file is given by the following formula, where K1 is the size of the key:

$$\text{data size} = 32{,}000 + (1.006 * (D + \text{nrecs_e} * (7 + K1)))$$

VRAM DF Extents

For VRAM files, DF extents normally range from a minimum of 1.2 KB (1,200 bytes) to a maximum of 10 KB (10,000 bytes) with the average being about 6 KB (6,000 bytes) or less. Files with keys, checkpoints, and/or a large number of updates have larger DF extents. In addition, whenever a checkpoint is issued during an append to a VRAM file, StorHouse produces another DF extent. The subsequent DF extents are usually larger because they contain additional entries.

The DF extents include various tables. Calculate the size of these tables, as applicable, as described in the following paragraphs. To compute DF size, always add the minimum DF size of 1.2 KB to the sum of the estimated table sizes.

All Files. For all files, estimate the size of the data table as follows, where data_xtnts is the total number of data extents in the file:

data table size= 13 + (12 ∗ data_xtnts)

A new data extent is created when a file is opened in mode=APPEND and either a checkpoint is performed or the file is closed normally.

Checkpointed Files. For files that will be checkpointed, estimate the size of the checkpoint table as follows, where max_cpts is the maximum number of checkpoints in one append operation:

checkpoint table size= 13 + (8 ∗ max_cpts)

Files with Keys. For files with keys, estimate the size of the key name and key segment location tables.

Estimate the size of the key name table as follows, where nkeys is the total number of keys defined for the file:

key name table size= 13 + (58 ∗ nkeys)

Estimate the size of the key segment location table as follows, where nsegs is the total number of key segments defined for the file:

key segment location table size= 13 + (6 ∗ nsegs)

Keysequential Files. For keysequential files, estimate the size of the key index table as follows, where K1 is the size of the key in bytes; data_xtnts is the total number of data extents in the file (a new data extent is created when the file has been opened in APPEND mode and either a checkpoint is performed or the file is closed normally); and nrecs_t is the total number of records in the file:

key index table size= 13 + (K1 + 6) ∗ (data_xtnts + (nrecs_t ∗ (K1 + 2) /31711))

A key index table is created only for keysequential files.

Updated Files. For files that will be updated (records changed or deleted), estimate the size of the change and record modification tables.

Note These tables need to be included in your estimate only if you plan to write additional records after updates have been performed.

Estimate the size of the change table as follows, where chg_xtnts is the total number of change extents:

change table size= 13 + (8 ∗ chg_xtnts)

Change extents exist only for files created with the StorHouse system parameter VRAM_UPDATE set to 1. A new change extent is created each time a file is opened with mode=UPDATE, records are changed (not just deleted), and the file is closed normally.

Estimate the size of the record modification table as follows, where nupds is the number of update entries:

record modification table size= 13 + (9 ∗ nupds)

An entry represents either an update of a single record or a group of consecutive records updated (all changed or all deleted) in respective order during one update operation.

VRAM K Extents

For KEYED files, the K (key data base) extent consists of an area for key data plus one index area for each user-defined key. The size of a K extent is the sum of the allocations made for all areas. The amount of data stored in each area can be estimated, but the actual amount may vary due to storage overhead, index compression, and the distribution of key values.

Note Whenever CHECKPOINT is issued during an append to a KEYED file, StorHouse produces another K extent. Your estimate for each checkpoint should be based on the size of the last (largest) K extent.

Estimate the size of the key data area as follows, where nrecs_t is the total number of records with keys that are written to the file in all extent sets, and K is the sum of the sizes of all user-defined keys:

key data area size = 4096 + (K + 12) ∗ nrecs_t ∗ 1.03

Estimate the size of each key index area for user-defined keys as follows, where Kn is the size of key number n:

Kn index area size = (Kn + 12) ∗ nrecs_t ∗ 2.06

VRAM Update Extents

If the VRAM_UPDATE system parameter is 1, the size of each update extent (C extent) can be calculated using the formula for a data extent, where N is the number of records updated in the update operation.

If the VRAM_UPDATE system parameter is 0, the size of the data in the first update extent (a U extent) can be calculated as follows, where N is the number of records updated and A is the average size of the new records:

first update extent = 32,000 + (1.006 ∗ N ∗ (A+16))

However, the minimum size of the extent is 64 KB, and it extends by 32 KB at a time. Thus, if the data occupied 65 KB, the extent would contain 96 KB.

The data size for any subsequent U extent is the data size calculated for the previous U extent plus the following, where N is the number of records and A is the average size of the records changed in the update operation:

addition to extent = 1.006 ∗ N ∗ (A+16)

Again, the minimum size of the extent is 64 KB, and it extends by 32 KB at a time. Thus, if the first U extent was 96 KB, of which 65 KB was data, and the update operation added 20 KB more data, the resulting U extent would still be 96 KB (of which 85 KB would be data).

Storage Allocation and Deallocation

StorHouse keeps track of available space on volumes, chooses volumes on which to record files, and allocates space for labels and data as needed. It allocates and deallocates storage space for volume sets and file sets upon your request or, in some cases, automatically. It allocates and deallocates space for files automatically based on user-specified guidelines.

When allocating space to write a file, StorHouse uses the general algorithm discussed below. The following sections describe additional preferences and limitations used in the search as well as other space allocation and deallocation procedures.

If the file is being written to a level F file set:

1. Search each partition in the file's destination file set for sufficient free storage.
2. If not found, then try to extend each partition in the file set to obtain sufficient free storage.
3. If unable to obtain enough space, reject the file operation.

If the file is being written to a file set on removable volumes:

1. If the performance buffer is being used, search each partition of the performance buffer's file set for sufficient free storage and allocate the space; otherwise, proceed to Step 4.

2. If sufficient free storage cannot be found in the performance buffer, initiate a migration and wait until sufficient storage becomes available; then allocate the space.

3. If unable to allocate enough space in the performance buffer, reject the file operation; otherwise, proceed to the next step.

4. Search each partition in the file's destination file set for sufficient free storage.

5. If not found, then try to extend each partition in the file set to obtain sufficient free storage.

6. If unable to obtain sufficient storage, try to extend the file set onto a surface in the file set's volume set (but not already in the file set).

7. If unable to obtain sufficient storage, try to extend the volume set by adding an empty volume from the volume set's free pool, then extend the file set onto the new volume.

8. If unable to obtain an empty volume, reject the file operation.

Blank Volumes, Empty Volumes, and Free Pool Volume Sets

A *blank* volume is not initialized; no volume or file labels have been written on the volume. An *empty* volume is initialized with volume labels but contains no file data or file labels.

A *free pool* is a special type of volume set that consists entirely of empty volumes of a specific media type and recording type. A free pool volume set name includes a period (.) followed by the device identification code (did) of the device in which the free pool resides, and the media type and recording type of the free pool volumes. A library device has only one free pool of empty volumes for each kind of media (in other words, media type and recording type combination) that it supports.

Allocation of Blank Volumes

Every library device has a corresponding FREE_POOL_didmmr system parameter. The value of FREE_POOL_didmmr determines the minimum number of empty volumes in the free pool volume set for the specified library device (did) and media (where mm is the media type and r is the recording type). When the number of volumes in any free pool volume set falls below its minimum value, StorHouse automatically asks the operator to load a blank volume of the appropriate type into

the exchange station of the respective library device. StorHouse initializes the blank volume and adds it to the free pool volume set.

If FREE_POOL_didmmr has a value of 0, the system does not automatically request blank volumes when a library device runs out of empty volumes of the corresponding media type.

Allocation of Empty Volumes

StorHouse adds empty volumes to a volume set when the volume set is created with a size greater than zero or when it is extended. The system extends a volume set when it cannot find enough free space in the volume set for an initial allocation or extension to a file set, or when the system administrator increases the size of the volume set. If the volume set cannot be extended to satisfy a request, the system deallocates any space allocated for the request and rejects the request.

StorHouse allocates empty volumes from the library device's free pool of empty volumes for the appropriate medium. It always adds an entire physical volume (in other words, both sides of a two-sided volume). It rejects any request that cannot be satisfied using volumes currently in the free pool.

Figure 4-2 illustrates allocating optical blank and empty volumes to a volume set.

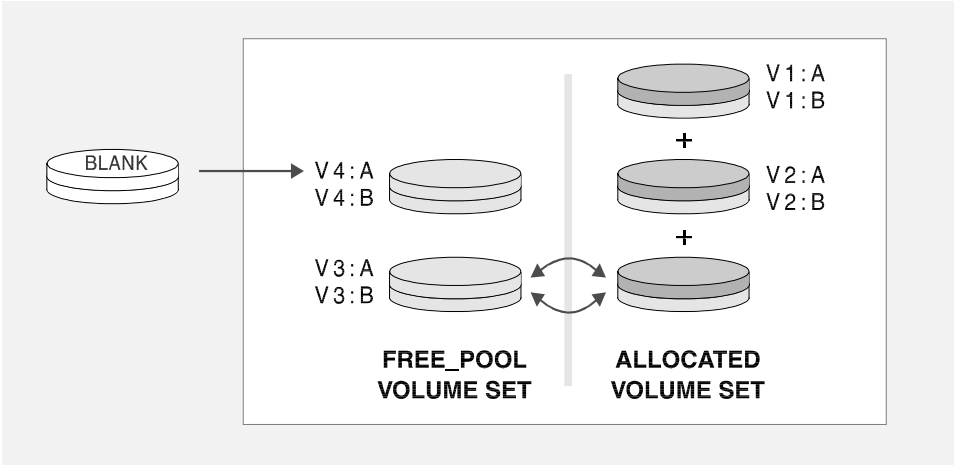

Figure 4-2: Volume Allocation

Placement of File Sets in Volume Sets

StorHouse allocates space for a file set in a volume set under one of the following conditions:

- Initial size allocation, contiguous space
- Initial size allocation, noncontiguous space
- Size extension, contiguous space
- Size extension, noncontiguous space.

Initial Size Allocations

The system allocates space for the initial size of the file set when the file set is created. In general, the system searches for space starting with the first available space in the volume set and continues until the size specified for the file set is satisfied. It starts by checking the first side of the first volume, then the second side of the first volume (if the media is logically double-sided), and so on. If the file set's update attribute is not zero, the system sets aside a percentage of the allocated size for file updates.

Contiguous Space. When allocating contiguous space to a file set, the system allocates space on the minimum number of sides required for the file set size. It allocates entire volume sides. It also allocates all remaining free space on the side up to the value of the limit attribute for the file set.

Noncontiguous Space. When allocating noncontiguous space to a file set, the system may allocate space on more than the minimum number of sides required for the file set size. It allocates all free space it locates in the volume set until the size specified for the file set is satisfied.

Size Extensions

The system extends the file set (allocates additional space) when the system administrator increases the size or when there is not enough general free space in an extendible file set for one or more new file extents. The system searches for free space on volume sides that already contain a portion of the file set before it searches other sides.

If the system must extend the file set for non-update extents, it allocates the additional size required plus any that must be set aside for updates (as determined by the update percentage). If it must extend the file set for update extents, it allocates only the additional size required.

The system will not extend a file set beyond the value of its LIMIT attribute and will reject any request that requires such an extension.

Contiguous Space. If a file set has the contiguous attribute, the system first tries to obtain the required size by allocating all free space on a volume side that already contains a partition of the file set. If this size requirement cannot be satisfied, the system then tries to extend the file set by entire volume sides. In either case, the file set is not extended beyond its LIMIT.

Noncontiguous Space. If a file set has the noncontiguous attribute, the system allocates the required amount of space for the extension on the first side it finds that has at least the required amount.

Placement of Files in File Sets

Each file extent is stored in a file set. You can specify a volume set and file set when creating a file; if you omit them, some commands use your default volume set and file set. If the system cannot determine which volume set or file set to use, it rejects the operation.

For file extents written to the performance buffer, the system reserves space for the extents in their primary file sets, and writes them back to primary file sets when appropriate. If the system cannot use the reserved space in a write-back, perhaps because a previous operation used part of it before aborting or the primary storage is disabled, it logs an error message to the operator.

The system allocates free space in a file set to store an entire file extent. If the extent does not use all of the allocated space, the system deallocates the remainder to free space. If there is not enough free space in the file set on any single volume side to hold the entire file extent and the file set is extendible, the system extends the file set. Otherwise, it rejects the request.

The system allocates update free space in a file set to store an entire file update. It allocates the space on the same volume side that contains the data being updated. If the update does not use all of the allocated space, the system deallocates the remainder to update free space. If there is not enough update free space in the file set on the desired volume side to hold the entire file update, the system allocates the additional space from the file set's general free space on the volume side. If there still is not enough free space, the system extends the file set or uses other file set free space as described for file set extensions.

Volume Set Storage Deallocation

StorHouse deallocates storage space from a volume set by removing empty volumes from the volume set and adding them to the appropriate pool of empty volumes.

There are two ways to deallocate volume set space: reduce the size attribute of the volume set or perform a release function.

When the system administrator reduces the size of a volume set, StorHouse deallocates empty volumes from the set until the set is reduced to the specified size or there are no more empty volumes. If the system cannot reduce the volume set to the specified size, it sets the final size equal to the actual size of the set (without the empty volumes) and returns an error response. If the system deallocates all volumes from the set (the size is zero), it still keeps the volume set in the directory.

The release function works much like size reduction, but with two differences. First, the function always attempts to reduce the size to zero. Second, if it reduces the size to zero, it removes the volume set from the directory.

File Set Storage Deallocation

StorHouse deallocates storage space from a file set by removing free space from the file set and adding it to the free space for the volume set.

There are two ways to deallocate file set space: reduce the size attribute of the file set or perform a release function.

When the system administrator reduces the size of a file set, StorHouse deallocates all unnecessary free space from the set and allocates any space it needs to obtain the specified size. For a noncontiguous set, all free space is unnecessary; for a contiguous set, if all space allocated to a file set on a volume side is free space, then that space is unnecessary. If the system cannot reduce the file set to the specified size, it returns an error response and does not reduce the size of the set. If the system deallocates all space from the set (the size is zero), it still keeps the file set in the directory.

The release function works much like size reduction, but with two differences. First, the function always attempts to reduce the size to zero. Second, if it reduces the size to zero, it removes the file set from the directory.

File Storage Deallocation

After StorHouse writes a file on a medium, it deallocates any unused space back to free space within the file set. Thus, once written, a file version has no free space associated with it. If a file is removed from a rewritable medium, StorHouse changes the space it occupied to free space in the file's file set. If a file is removed from a write-once or erasable medium, StorHouse reduces the size of its file set by the size of the file and accounts for the deleted space as part of the volume's space.

Chapter

5

Storage and User File Management

StorHouse provides a comprehensive set of features for managing user files and volumes. These features include:

- Backing up, archiving, duplexing, recovering, removing, and staging user files
- Retaining user files for compliance purposes
- Replicating user files from a source StorHouse system to a destination StorHouse
- Migrating user files between the performance buffer and their resident file sets
- Migrating user files between volume sets
- Migrating, exporting, importing, erasing, retiring, and recovering volumes.

This chapter discusses these features and the file attributes that affect them.

User File Backup

StorHouse supports the following user file backup operations:

- Creating a backup copy of a primary file in a backup volume set (CREATE BACKUP command)

- Writing back, or copying, a file from the performance buffer to its resident file set (BACKUP command)

- Creating a separate archive copy of a primary file in an archive volume set (ARCHIVE command).

Figure 5-1 illustrates these operations using the Command Language BACKUP, CREATE BACKUP, and ARCHIVE commands.

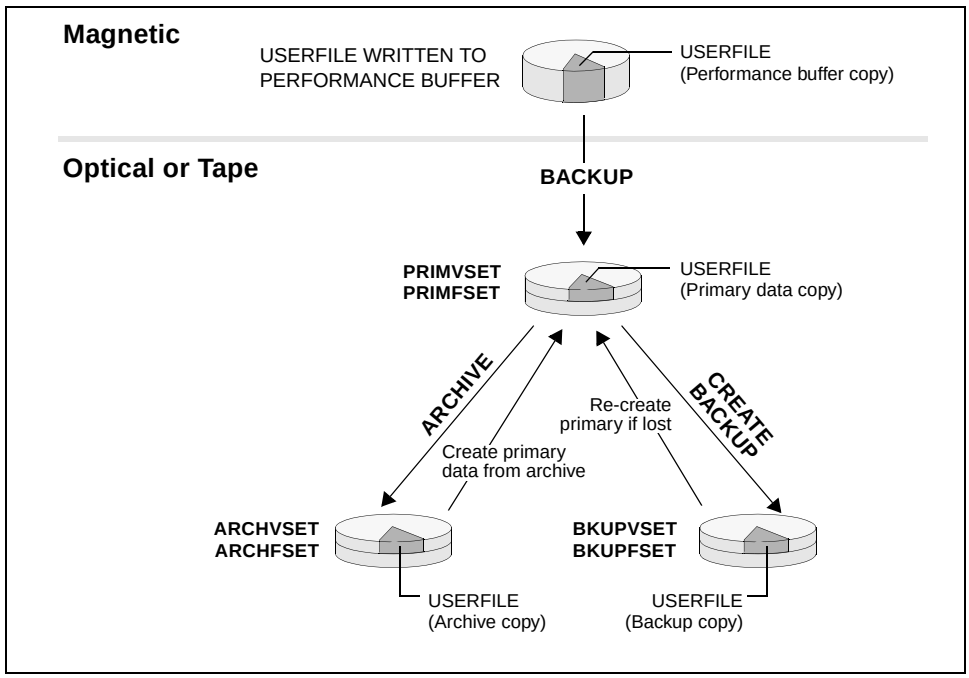

Figure 5-1: BACKUP, CREATE BACKUP, and ARCHIVE Commands

File Attributes

File attributes are characteristics assigned to files at creation time. StorHouse derives file attribute values from system parameters or user specifications. The file attributes that affect backup operations are:

- BACKUP
- VTF

BACKUP Attribute

The BACKUP file attribute indicates whether a file version is a candidate for backing up and archiving (CREATE BACKUP and ARCHIVE commands).

Valid values for the BACKUP attribute are:

- BACKUP – the file version is a candidate.
- NOBACKUP – the file version is not a candidate.

If the BACKUP attribute is omitted at file creation, the BACKUP system parameter provides the default attribute value for new file versions.

- If the BACKUP system parameter is set to TRUE, the default BACKUP attribute for the file is BACKUP.
- If the BACKUP system parameter is set to FALSE, the default BACKUP attribute for the file is NOBACKUP.

Once a file version exists, its BACKUP attribute can be changed using the StorHouse Command Language SET FILE command.

VTF Attribute

The VTF, or *Vulnerability Time Factor*, attribute specifies how long StorHouse can leave new extents of a file version in the performance buffer before copying them (perform a *write-back* with the BACKUP command) to their primary file set.

Valid values for the VTF attribute are:

- DIRECT – Bypass the performance buffer, and write the file directly to its primary file set.
- NOW – Write the file version to the performance buffer, and then copy it to its primary file set immediately.
- NEXT – Write the file version to the performance buffer and copy the version to its primary file set the next time the system performs the BACKUP command.

The system parameter VTF provides the default value (DIRECT, NOW, or NEXT) for the VTF file attribute for new files.

If you write a file to a level F file set other than the performance buffer, StorHouse always writes the data directly to the destination file set and does not use the performance buffer, (just as if the VTF attribute were set to DIRECT). In this case, the actual value of the VTF is ignored.

CREATE BACKUP Command

Users with SYSTEM privilege can use the CREATE BACKUP command to create backup copies of user file versions based on the files' BACKUP attribute, current backup status, and location. The command backs up primary file versions that are not already backed up and that have a BACKUP attribute of BACKUP to a user-specified file set and volume set in the backup directory. The system considers a file version to be backed up when there is a current copy of the most recent revision of the version in the backup directory.

CREATE BACKUP can perform full or incremental backups. Incremental is the default.

BACKUP Command

Users with SYSTEM privilege can use the BACKUP command to perform a write-back operation, which copies new file extents from the performance buffer to their primary file sets based on their VTF file attribute. A file version's primary file set is the file set assigned to the version at creation.

When the system writes back an extent, it also writes back all other new extents of the file version to keep the extents together on the destination volume. It copies the DF extents first so that they will appear before the rest of the file on the volume. This order improves read performance for sequential media (for example, magnetic tape) because the system reads the DF extent first when opening a file.

ARCHIVE Command

Users with SYSTEM privilege can use the ARCHIVE command to copy files from their primary volume sets to archive volume sets, leaving the source (primary) file intact. ARCHIVE copies all of the specified file(s), including any extents located in the performance buffer. For a file to qualify for archiving, it must not be marked as previously archived, and it must have a BACKUP attribute of BACKUP.

The ARCHIVE command can perform full or incremental archiving. Incremental is the default. If you specify /NOINCREMENTAL and an archive already exists for a selected file, the old archive is deleted and replaced by the new one.

User File Duplexing

The *duplex* feature allows StorHouse to read the backup or archive (duplex) copy of a file extent when the primary copy is unavailable or for load balancing. If the primary file copy resides on a volume that is in an offline library device or if you want to perform load balancing, you must maintain the primary and duplex file copies in separate online library devices. By using file duplexing, you greatly increase the availability of your user files at all times.

In addition to increasing the availability of user files, you can improve your system performance by *load balancing* across library devices. In this case, StorHouse automatically determines when to access the duplex file copy instead of the primary copy for better overall system performance. The StorHouse system administrator can select the degree of similarity (the number of media code characters that must match) between the primary and duplex media to be used in duplex load balancing operations. For example, load balancing can be performed across erasable and non-erasable optical devices, different types of tape devices, or even across optical and tape devices.

Duplex feature operations are transparent to StorHouse users (including applications). The system administrator is responsible for setting up the StorHouse system for duplex operations, and maintaining and monitoring StorHouse operations as they relate to the duplex feature. The system administrator can manage duplex operations by setting specified system parameters.

As long as a file remains in the performance buffer, StorHouse will always access the file's performance buffer copy. In this case, the duplex file copy is not accessed.

StorHouse file read statistics displayed by the SHOW FILE command always report the last access as an access to the primary file copy.

FileTek recommends the following for StorHouse systems that use tape:

- Always keep two copies of your data (for example, the primary copy and a backup or archive copy).
- Create the backup or archive copy from the file copy on the performance buffer. In other words, run the CREATE BACKUP command before you migrate data from the performance buffer.
- Set the DUPLEX_ENABLE system parameter to ALL. With this setting, StorHouse will always use the duplex copy of a file that resides on tape when the primary copy is unavailable or busy.

The following sections explain the conditions necessary for StorHouse to access duplex file copies, and how StorHouse will process those requests.

Accessing the Duplex Copy of a File Extent in Place of a Disabled Primary Volume

Assuming the primary copy of a file extent is on a disabled volume and the duplex copy of the requested file extent resides in an online library or on a shelf, StorHouse will use the duplex feature to access the duplex copy of a file extent for reading under the following conditions:

- The DUPLEX_ENABLE system parameter is set to DISABLED, OFFLINE, or ALL
- The file extent is backed up or archived (in other words, an entry for the file extent resides in the directory indicated by the DUPLEX_DIRECTORY system parameter) in an online library device or on a shelf.

If these conditions are met, StorHouse will process the request using the extent from the BACKUP or ARCHIVE directory instead of the copy on the disabled primary volume.

If the duplex copy is on a shelf, StorHouse will instruct the operator to load the specified volume into a library device. If the duplex copy does not exist, is unreadable, or is inaccessible, StorHouse will return an error response.

The system administrator can issue the SET VOLUME /DISABLE command to disable a volume and the SET VOLUME /ENABLE command to enable a disabled volume.

Accessing the Duplex Copy of a File Extent in Place of an Offline Primary Volume

Assuming the requested primary file extent resides on a volume in either an offline library device or on a shelf whose associated library device is offline, StorHouse will use the duplex feature to access the duplex copy of the file extent for reading under the following conditions:

- The DUPLEX_ENABLE system parameter is set to OFFLINE or ALL.
- The file extent is backed up or archived (in other words, an entry for the file extent resides in the directory indicated by the DUPLEX_DIRECTORY system parameter) in a separate online library device or on a shelf.

If these conditions are met, StorHouse will process the request using the extent from the indicated directory instead of returning an “offline” error status. (You will receive an offline error status if the file extent is backed up or archived in the same library device.)

Accessing the Duplex Copy of a File Extent for Load Balancing

Assuming the primary and duplex copies of the requested file extent reside in different online libraries, StorHouse will use the duplex feature to load balance under the following conditions:

- The DUPLEX_ENABLE system parameter is set to ALL to instruct StorHouse to automatically perform load balancing between primary and duplex copies of files
- The DUPLEX_BALANCE system parameter is set to one of the following values:
    - 0 indicates that any two media can be used for load balancing (for example, erasable optical and tape)
    - 1 indicates that the primary and duplex media type codes have the same first character (for example, erasable optical and WORM optical)

  - 2 indicates that the primary and duplex media type codes are the same (for example, single-density WORM optical and double-density WORM optical)
  - 3 indicates that the primary and duplex media type and recording type codes are the same (for example, 541.0-meter, compressed tape for both)

- StorHouse determines that using the duplex copy would result in better overall system performance. (This factor is determined by a number of criteria, such as the number of online drives in the libraries and the volumes currently mounted.)

If these conditions are met, StorHouse will process the request using the extent from the BACKUP or ARCHIVE directory.

If the duplex or primary copy does not exist or is inaccessible, StorHouse will select the other copy. If there is an unrecoverable read error or other failure during the data transfer for the selected copy, StorHouse will terminate the transfer and return an error response.

For more information on the duplex feature and load balancing, see the *System Administrator's Guide*.

User File Retention

StorHouse provides features for setting and enforcing user file retention. These features ensure StorHouse archiving adequately supports industry-based compliance rules and data remains accessible throughout its required life span.

File Retention Attribute

Each file written to StorHouse has a *retention attribute*, which determines a file's retention period. A *retention period* indicates the time span that a file may not be deleted from StorHouse. A file is considered *retained* when it has a non-zero retention period and that retention period has not expired.

Copies of primary files (for example, backup, archive, replica, and relocated files) inherit the retention attribute of the primary. The exception is the performance buffer copy. It is treated as temporary cache and has no retention attribute.

Calculating File Retention Periods

StorHouse calculates file retention periods by adding the number of days in the retention period to a file's last modified date (the modified_file date as displayed by the StorHouse Command Language SHOW FILE command). A file retention period ends when the current date is beyond the last modified date plus the retention period.

For example, if a file with a 3-day retention period was last modified at 11 p.m. on December 12, it would expire at 11 p.m. on December 15.

StorHouse does not automatically delete files after their retention period expires. It simply allows applications or users to delete them.

Setting Retention Periods

Retention can be set at the file and file set level. File-level retention takes precedence over file set-level retention unless the /FORCE_RETENTION modifier is in effect on the file's resident file set.

File-level Retention

You can set file-level retention when writing a file to StorHouse through the Interactive Interface or the Generic Callable Interface by specifying:

- The /RETENTION modifier on the StorHouse Command Language CREATE FILE or PUT command.
- The retention_interval list member on the Generic Callable Interface LSMCO or LSMOS (MODE=WRITE) function or the FATTR-RETENTION-INTERVAL on the mainframe Callable Interface CREATE-OPEN or OPEN-SEQ (MODE=WRITE) function.

A user or an application can specify one of four retention values at file create time:

- DEFAULT (the retention is not specified and assumes the default value).
- FOREVER (never allow the file to be deleted).
- ZERO (do not apply any retention period to the file).
- Number of days (retain the file for the specified number of days).

The maximum retention period is 65,000 days, or approximately 178 years.

StorHouse records a file's retention setting on the media where the file resides. If the retention period is modified, the system updates the media to preserve the change.

File Set-Level Retention

Refer to the section, "File Set Retention Attribute," on page 4-12 for information about specifying file set-level retention.

Determining File Retention Values When DEFAULT is Specified

When StorHouse creates a primary file, it checks several places to determine the file's retention value. As soon as StorHouse finds a retention value other than DEFAULT, it assigns that value to the file.

- First, StorHouse checks the Interactive Interface command or Callable Interface function that creates the file. If it finds a value other than DEFAULT, StorHouse assigns that value to the file unless /FORCE_RETENTION is in effect on the file set. In this case, StorHouse overrides the file-level specification with the file set-level specification.

- If the file retention value is DEFAULT, StorHouse checks the file set retention attribute for a value other than DEFAULT. If found, it assigns that retention value to the file.

- If the file set retention value is DEFAULT, StorHouse uses the value of the RETENTION_MODE system parameter to determine file retention. If RETENTION_MODE is set to BASIC, StorHouse sets the file-level retention to ZERO, which indicates no retention. If set to STRICT, StorHouse sets the file - level retention to FOREVER, which indicates infinite retention.

Refer to the *Command Language Reference Manual* for more information about the RETENTION_MODE system parameter.

Effect of Retention on Select StorHouse Commands

Retention affects the operation of several StorHouse commands. For example, commands like RELOCATE contain internal delete operations. When RELOCATE moves a retained file to a different storage location, it cannot delete the original file copy from the source location because its retention period has not expired. The original file essentially becomes orphaned on the media.

Retention also affects the operation of the following commands, which, like RELOCATE, perform internal delete operations:

- DELETE
- ERASE VOLUME
- ERASE VSET
- EXPORT
- INITIALIZE DEVICE
- PURGE
- PUT
- RECOVER VOLUME
- RELOCATE
- RETIRE VOLUME

- UNCATALOG VOLUME
- UNCATALOG VSET
- CREATE FILE/REPLACE.

Release-Dependent Retention Information

Note the following:

- To specify retention on a StorHouse 5.6 system, you can use any version of the StorHouse Host Application Programming Interface (API) with the Interactive Interface. You must use version 2.5 (for Windows/UNIX) or version 1.8 (for MVS) of the StorHouse Host API with the Callable Interface. If you use prior versions of the Host API, StorHouse assumes that the retention period is the default.

- The new Host API releases do not pass retention information from Callable Interface functions to a StorHouse system running any release prior to StorHouse 5.6. Therefore, the Callable Interface can be used with older StorHouse systems. However, interactive commands with retention parameters/modifiers will fail if executed on a StorHouse system running a release prior to 5.6.

- When an older StorHouse release is upgraded to Release 5.6, files and file sets created with previous StorHouse releases will be assigned a retention attribute of ZERO after the conversion.

User File Replication

StorHouse supports application-independent replication of one or more eligible primary files from a source StorHouse system to a destination, or target, StorHouse system. Replication is typically used to provide redundancy for disaster recovery. A file is considered *replicated* when at least one copy resides on the destination system, including on the performance buffer. Any StorHouse file type can be replicated (VRAM, sequential, or STORHOUSE).

Replication Class Name Attribute

A primary file is eligible for replication when it has an assigned replication class, or a named set of information about the destination StorHouse system. A file's *replication name attribute* specifies the name of the assigned replication class or the value "none" to indicate that the file is not eligible for replication.

Table 5-1 lists the components of a replication class.

Table 5-1: Replication Class Components

Component	Specifies the
RPL_CLASS_NAME	Name of the replication class. A replication class name can consist of from 1-8 of the following ASCII characters: A_Z, 0-9, _ and $. StorHouse always forces replication class names to uppercase, even when enclosed in quotes.
DISABLED_FLAG	Indicator specifying whether the replication class is disabled. A non-blank character indicates disabled.
NETWORK_DEVICE	Network device used to connect to the destination StorHouse system. This value must be set to N00 to indicate TCP/IP.
SYSNAME	Network system name used to connect to the destination StorHouse system. This value must be the same as the SM_HOSTID in the SMCONFIG file.
LINKNAME	Network link name used to connect to the destination StorHouse system. This value must be the same as the SM_LINKNAME in the SMCONFIG file.
VSET	Volume set that will contain the replicated files on the destination StorHouse. Valid values are: • VSET=volume_set_name • VSET=* (indicates to use the same volume set name as the source file) If VSET is omitted, StorHouse uses the account's default volume set on the destination StorHouse.
FSET	File set name that will contain the replicated files on the destination StorHouse system. Valid values are: • FSET=file_set_name • FSET=* (indicates to use the same file set name as the source file) If FSET is omitted, StorHouse uses the account's default file set on the destination StorHouse system.

StorHouse associates a file with a replication class by:

- Assigning a replication class to the file's resident file set (SET FSET or CREATE FSET command). In this case, at create time, the file inherits its replication class from its file set.

- Explicitly specifying a default replication class for the file version (SET FILE command).

Currently, FileTek is responsible for creating and maintaining replication classes. For more information about these tasks, contact your FileTek customer support representative.

Implementing Replication

System administrators implement replication with the StorHouse Command Language REPLICATE command. For convenience, this command can be scheduled to run periodically.

A replication operation works as follows:

- When an application creates or modifies a replication-eligible file, StorHouse queues the same action to occur on the destination StorHouse system the next time the REPLICATE command executes on the source system and selects that file.

- When a file is deleted (and removed) from the source system, StorHouse queues the same action to occur on the destination StorHouse the next time the REPLICATE command executes on the destination system and selects any file.

REPLICATE requires SYSTEM privilege.

StorHouse does not copy file and file set replication class attributes from one system to another. A replica inherits the replication class attribute of its target file set on the remote system. In addition, the system does not write queued files to the destination system in any particular order or within a set time interval.

REPLICATE uses the same account, password, and file access group on the source and target systems. The system administrator must manually create the respective accounts and volume sets on the target location. If the required groups and file sets do not already exist on the destination system, StorHouse creates them automatically. System-created file sets are noncontiguous and use the other CREATE FSET command defaults.

For more information about the REPLICATE command, refer to the *Command Language Reference Manual*.

User File Recovery

If the primary copy of a user file is unreadable, users with COPY privilege can execute the CREATE PRIMARY command to create a new primary copy from the backup or

archive copy. If the most recent backup or archive copy is not available but an older version of the file exists, users may be able to update the older version.

User File Removal

StorHouse enables users with the proper privilege to remove deleted files from the deleted file directory. It also enables users to control the number of primary file versions to retain.

Removing Deleted Files

Users with SYSTEM privilege can execute the REMOVE FILE command to remove information for files that are marked as deleted in the deleted file directory. If extents of the file version to be deleted reside on fixed erasable storage, StorHouse also frees that storage when it removes the file.

The remove operation queues remote delete requests to the target systems for all replicated files that are removed. Note that a REPLICATE operation must be run to forward the delete requests to the target system.

If the DEL_FILE_PERM system parameter is TRUE, REMOVE FILE writes deleted file labels for each extent of a deleted file version removed from an erasable or non-erasable removable volume; otherwise, the command does not write deleted file labels.

LIMIT Attribute

The *LIMIT file attribute* specifies the number of primary file versions that StorHouse will retain. It applies *only* to primary files. If you exceed the LIMIT value for a file, the system automatically deletes the oldest version(s) so that the total number of versions equals the LIMIT value. (Older file versions have the most negative relative version numbers.)

The exception is for retained files. Consider the following:

- You add a new version to an existing file, and the number of file versions exceeds the LIMIT value.
- The oldest file version is retained.

In this case, StorHouse keeps all retained versions. It does not delete the oldest one.

Valid LIMIT values range from 1 to 32,768. If LIMIT is not specified at file creation, the LIMIT system parameter provides the default. (StorHouse always maintains a

limit value of 65,653 for backup and archive file copies to protect them from unintentional deletion. You cannot change this value.)

User File Migration From the Performance Buffer

StorHouse automatically migrates, or moves, file extents off the performance buffer based on extent access histories, extent size, and ATF file attributes.

ATF Attribute

The ATF, or *Access Time Factor*, attribute indicates the importance of access time for a file. The ATF must be a numeric value from 1 to 3. A value of 1 indicates that a short access time is very important; 2 indicates that it is moderately important, and 3 indicates that it is minimally important.

The ATF attribute works in conjunction with the MIGRATE function. The MIGRATE function migrates files with larger ATF values off the performance buffer before files with smaller ATF values. The system parameter ATF provides the default value for the ATF file attribute for new file versions.

MIGRATE Function

The goal of the MIGRATE function is to keep file extents in the performance buffer that are most likely to be accessed, while maintaining a supply of free space in the buffer. To maintain free space in the buffer, the function removes extents from the performance buffer. The MIGRATE function cannot migrate an extent that has not yet been written back to its primary file set. Also, it does not migrate file extents from performance buffer space that is reserved by a group for its exclusive use.

Migration Factor

System parameters determine when to start a migration and how many bytes of data to migrate. The *migration factor* determines which extents to migrate. StorHouse maintains a migration factor for each extent in the performance buffer. The migration factor is a 32-bit integer that is derived from the file's ATF attribute, size, and access history. Extents with the smallest migration factor values are migrated off the performance buffer first.

StorHouse subtracts the file's ATF value from the largest ATF value (3) and places the result in the most significant portion of the migration factor. Thus, file extents with an ATF of 3 are migrated before extents with an ATF of 2, which are migrated before extents with an ATF of 1.

The value of the rest of the migration factor is based on the number of times the extent has been accessed, when the accesses occurred, and the size of the extent. This portion of the migration factor is called the *usage factor*.

Each time a file extent is accessed, the system divides the number 2^{24} (2 raised to the 24th power) by the number of units of storage the extent occupies and adds it to the migration factor. The size of a unit of storage is specified by the MIG_FAC_UNIT system parameter. The value of the usage factor cannot exceed 2^{29} - 1. Thus, extents with many accesses tend to have larger migration factors than extents with few accesses, and extents with larger sizes tend to have smaller migration factors than extents with smaller file sizes.

Periodically, the system divides the usage factor for each extent in half. The period is specified by the MIG_FAC_PERIOD system parameter. Thus, extents with older accesses tend to have smaller migration factors than extents with more recent accesses.

Migration Function Operation

The MIGRATE function can be started automatically, or the system administrator can initiate it interactively or schedule it by command.

StorHouse starts a migration automatically when the amount of free storage in the performance buffer drops below the amount indicated by the MIG_MIN system parameter. The system migrates enough extents off the buffer to bring the amount of free storage to the amount indicated by the MIG_MAX system parameter. If the function cannot free enough storage, it sends a warning message to the operator and log.

If the available space in the performance buffer equals or exceeds the amount indicated by the MIG_MAX system parameter, no extents are migrated. If available space in the performance buffer is less than the amount indicated by the MIG_MAX system parameter, the system migrates file extents until the available space equals or exceeds MIG_MAX. If MIG_MAX was not reached and there are new extents, the system initiates a BACKUP command to copy new file extents from the performance buffer to their primary file sets. Once the backup is complete, the migrate continues.

The system administrator can migrate files off the performance buffer by invoking the MIGRATE command. The command initiates a MIGRATE function, as described above. The MIGRATE function can run while the system is available for normal operations.

Figure 5-2 illustrates file write-back and migration between the performance buffer and level L storage.

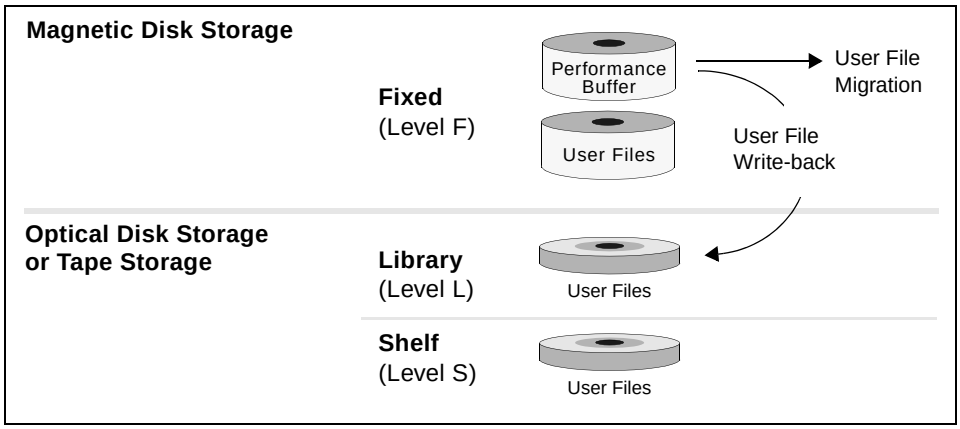

Figure 5-2: File Write-back and Migration

User File Migration Between Volume Sets

A system administrator with the appropriate authorization can use the MIGRATE /BY_VSET command to move data from one medium to another within StorHouse to obtain the best trade-off between performance and cost of storage. For example, you can migrate data from erasable optical disk to lower cost magnetic tape as the data ages.

StorHouse allows multiple MIGRATE /BY_VSET commands to be executed at one time.

The MIGRATE /BY_VSET command migrates files from a source volume set to a destination volume set based on user-specified migration criteria to make space available in the source volume set for new data. StorHouse migrates files only if available space is needed in the source volume set to satisfy a user-designated minimum amount. The source volume set must be erasable media and cannot be a level F volume set. Also, the source and destination volume sets must be different, but must both reside in the same directory. The source volume set must have a non-zero LIMIT attribute, or the command will not migrate files.

The MIGRATE /BY_VSET command ensures that a volume set has a minimum amount of available space for new data allocations by using one or both of the following operations:

- It verifies that the size of the volume set can be increased (that is, empty volumes can be added, although the command does not actually add them).

- It migrates all files off source volumes to the destination volume set so that it can erase the source volumes.

Note If a copy of a file extent exists in the performance buffer when the MIGRATE /BY_VSET command migrates the file, StorHouse retains the performance buffer copy of the extent.

StorHouse does not perform volume set migrations automatically. Volume set migration is driven entirely by the MIGRATE /BY_VSET command, which can be entered interactively or scheduled. Each execution of a MIGRATE /BY_VSET command is independent of any other MIGRATE /BY_VSET command.

A system administrator can change a scheduled MIGRATE /BY_VSET command at any time by removing it from the schedule and scheduling it again using different parameter and/or modifier values. For example, the administrator may change the criteria for selecting files to migrate, the minimum amount of space to have available after each time the command is invoked, or the destination volume set.

How Files are Selected for Volume Set File Migration

The MIGRATE /BY_VSET command selects files for migration based on characteristics of the volume on which the files reside. The command selects whole physical volumes (both sides of two-sided volumes) of files for migration.

The system administrator can specify the amount of space to make available and the order in which volumes are selected using the /MINIMUM and /ORDER_BY modifiers, respectively. These modifiers are described as follows:

- The /MINIMUM command modifier specifies the minimum number of empty volume sides that must be available to the source volume set for new data after the MIGRATE /BY_VSET command completes. The volume sides can be empty volume sides in the volume set, or they can be volume sides that might be allocated to the volume set from the free pool without exceeding the volume set LIMIT attribute.

- The /ORDER_BY command modifier specifies the type of timestamp field to be used by StorHouse to sort volume sides (in ascending order) before selecting them for file migrations. The timestamp field can be a logical volume's (volume side's) first space allocation time, last space allocation time, or last modification time. The system administrator can use existing volume attributes for deactivating and cycling volumes to manage storage allocations for new data for appropriate access and aging characteristics.

A volume is considered to be empty if the following conditions are true:

- It has no active files or files that have been deleted or removed on it.

- The volume state is CATALOGED.
- The volume side is not WRITELOCKED, DEACTIVATED, or DISABLED.

For a source volume set with a non-zero CYCLE attribute, the MIGRATE /BY_VSET command assumes that volumes allocated from the free pool and erased volumes will provide only one empty side because the second side of two-sided media will automatically be deactivated. If the source volume's CYCLE attribute is not zero and the volume set consists of two-sided media, then the maximum number of empty volume sides is only half of the sides in the volume set because the other half will be deactivated and/or used.

When selecting volumes for migration, the MIGRATE /BY_VSET command processes volume sides with CATALOGED, ERASING, and WRITELOCKED states but skips empty volume sides. The command does not select physical volumes with one or more DISABLED, CATALOGING, UNCATALOGING, or UNCATALOGED sides. The command selects the volume side with an appropriate volume state and attribute, and the lowest (oldest) time value indicated by /ORDER_BY. If the physical volume containing the selected logical volume has two sides, both sides are migrated regardless of the time value for the opposite side.

The command will complete successfully without migrating files if either of the following two conditions is true when the command is invoked:

- The number of empty volume sides in the source volume set is greater than or equal to the /MINIMUM.

- Extending the source volume set (in other words, adding volumes from the free pool) would satisfy the /MINIMUM without exceeding the source volume set's LIMIT attribute. (However, the command does not actually extend the volume set.)

The system administrator must limit the size of the source volume set using the volume set LIMIT attribute. Otherwise, StorHouse will obtain empty volumes for that volume set by allocating them from the free pool rather than by migrating and erasing files.

How Files are Migrated for Volume Set File Migration

The MIGRATE /BY_VSET command performs the equivalent of a RELOCATE command to migrate each file from its source file set to a file set with the same name in the destination volume set. The destination volume set must already exist. (See the *Command Language Reference Manual* for more information on RELOCATE and all other StorHouse Command Language commands.) Before migrating files, however, the command checks to see if the destination file set already exists. If the destination file set does not exist, StorHouse creates one with the same name as the source file set using the NONCONTIGUOUS attribute and the CREATE FSET command default

values for other attributes. After copying each file to its destination file set, the MIGRATE /BY_VSET command updates the directory to use the file's new destination copy and removes all information on the file's source copy. In effect, this changes the *residence* (the primary file set) of the file.

The MIGRATE /BY_VSET command migrates all files off one or more whole physical volumes in the source volume set to one or more volumes in the destination volume set. When the command selects a volume for file migration, it writelocks and deactivates all sides of the volume to prevent any further space allocations or writes to the volume. After migrating all files off a source volume, the command performs the equivalent of an UNCATALOG VOLUME command and an ERASE VOLUME command for the volume. When the erase completes, the volume is an empty volume in the source volume set that can be reused.

Volume Set File Migration Operation

This section describes the steps that occur during a volume set file migration.

During a volume set file migration, the following events take place:

1. The MIGRATE /BY_VSET command first verifies the command parameters. If the destination volume set does not exist or the /MINIMUM value is larger than the source volume set's LIMIT attribute value, the command returns an error.

2. The MIGRATE /BY_VSET command counts the number of empty volume sides that are in the volume set. If the number of empty volume sides is greater than or equal to the /MINIMUM value, it returns a successful status.

3. If the number of empty volume sides is less than the /MINIMUM value, it determines if the volume set LIMIT attribute will allow the system to allocate enough empty volumes from the free pool to satisfy the /MINIMUM. If the LIMIT is large enough, the MIGRATE /BY_VSET command returns a successful completion without actually allocating volumes from the free pool.

4. If the LIMIT is not large enough or the volume set is already at its LIMIT, the command uses the /ORDER_BY parameter value to select enough used volumes from the source volume set to reach the /MINIMUM level. Then, it migrates and erases each selected volume. If the command is unable to complete the processing for any volume, it returns an error status for that volume, leaves the volume writelocked and deactivated, and continues with the next volume.

5. If the MIGRATE /BY_VSET command is able to satisfy the /MINIMUM level, it return a successful status; otherwise, it returns an error status. When the command completes, it returns the final status to the user.

User File Staging

You can *stage*, or copy, primary files (file extents) from their resident file sets on Level L storage to the performance buffer. Files cannot be staged from a Level F file set. StorHouse supports both manual and automatic file staging. This process is also known as *upward migration*.

To activate manual file staging, your system administrator must set the MIG_STAGE_LOAD system parameter to a non-zero value. Then, if you have the proper privilege, you can use the STAGE command to manually copy a specified file to the performance buffer.

To activate automatic file staging, your system administrator must set the system parameters MIG_REPOP_LOAD and MIG_REPOP_MAX to non-zero values. Also, the system administrator or a properly privileged user must set the AUTO_STAGE attribute for a file set using either the CREATE FSET /AUTO_STAGE or SET FSET /AUTO_STAGE command. (The system administrator can set the system parameters before or after setting the attribute.) Once automatic staging is enabled and a file in that file set is accessed for a host read (user request), StorHouse automatically stages the file extents to the performance buffer.

To see whether a file set is assigned the AUTO_STAGE attribute, you can submit the SHOW FSET command. If the attribute is not set for a file set, the state of the attribute does not display.

When manually staging a file, StorHouse copies all extents (or all special extents if you specified /SPECIAL) of the specified file that are not already in the performance buffer. When automatically staging files in a file set, StorHouse copies only the referenced file extents that are not already in the performance buffer. In both cases, source files are left intact, and files can be staged from any media. If sufficient performance buffer space is not available for a staging operation, StorHouse will not stage the file(s), nor will StorHouse initiate a general file migration to make space available.

The following figure describes file staging:

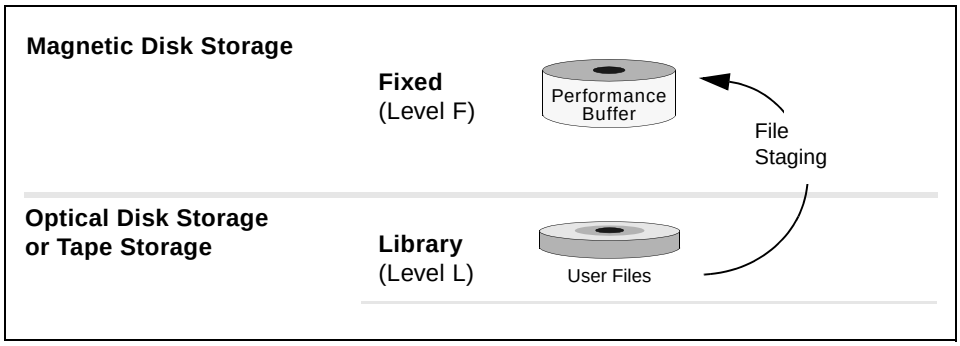

Figure 5-3: File Staging

Volume Migration

StorHouse automatically migrates volumes between level L and level S storage devices. The system migrates blank volumes into a library device (level L) as described in the section "Blank Volumes, Empty Volumes, and Free Pool Volume Sets" on page 4-20.

If the library device has no empty slots for a new volume, the system first migrates a volume from the library to level S. It selects a volume to migrate to level S that is not in use, has the oldest access time, and has a HOLD attribute value of NOHOLD (or a value of HOLD if all volumes in the library have a value of HOLD). The system sends a message to the system console and all console-enabled terminals, requesting that the operator remove the old volume from the device exchange station and store it on a shelf prior to requesting a blank volume.

If the system needs a volume currently stored on level S, it requests the operator to load that volume into the exchange station. If the library device has no empty slots for the needed volume, the system migrates the volume that is not in use, has the oldest access time, and has a HOLD attribute value of NOHOLD (or a value of HOLD if all volumes in the library have a value of HOLD) to level S prior to requesting the needed volume. Once loaded, the system uses the volume and then places it in a slot. The system keeps the volume in the device until it is migrated again.

The system operator can move specific volumes between library devices or between a library device and shelf storage (level S) using the MOVE VOLUME or MOVE VSET command. Or, the operator can specify a wildcard in the volume identifier parameter of the MOVE VOLUME command to move all volumes in a specified volume set or location to a new location. Both commands update the last access time for the moved volumes, making them the least likely volumes (in their HOLD class) to be migrated if they are moved into a library device.

Volume Export and Import

Volumes can be classified as follows:

- *Exported* or *offline* – StorHouse has no knowledge of the volume.
- *Uncataloged* – StorHouse maintains volume and volume set information for the volume, but has no knowledge of files and file sets.
- *Cataloged* – StorHouse maintains file and file set information for the volume.

The system administrator can execute StorHouse Command Language commands to change the classification of a single volume, several volumes, or all volumes in a volume set. The EXPORT, IMPORT, CATALOG VSET, UNCATALOG VOLUME, and

UNCATALOG VSET commands allow the system administrator to import and export, catalog, or uncatalog volume sets or individual volumes.

These commands are described in the following sections. For more information on these commands, see the *Command Language Reference Manual*.

Volume Export and Uncatalog

The system administrator can remove uncataloged volumes and volume sets from the system by using the EXPORT command. The volume or volume set must be in the uncataloged state. The UNCATALOG VSET command removes directory information for files and file sets on a volume set. The UNCATALOG VOLUME command removes directory information for files and reduces file sets on a volume.

EXPORT removes all directory information for the volumes from the system directories and directs the operator to remove the physical volumes from the level L exchange station or level S shelves. EXPORT releases all empty volumes from a volume set to the free pool so that it does not export empty volumes as part of a user volume set.

Exported volumes or volume sets can be imported into another StorHouse system.

Special Considerations for Uncataloging Volumes with Retained Files. The retention feature affects the operation of the UNCATALOG command. Therefore, customers who currently use UNCATALOG and EXPORT to remove archive volumes from StorHouse may want to rethink this strategy if they also plan to use the StorHouse retention feature.

StorHouse cannot uncatalog a volume that contains retained files because those files cannot be deleted until their retention period expires. When StorHouse attempts to uncatalog such a volume, it ignores the retained files, deletes and removes all non-retained files on the volume, and leaves the volume in the UNCATALOGING state. A volume in the UNCATALOGING state cannot be exported. To circumvent this restriction, FileTek recommends that instead of using UNCATALOG and EXPORT to remove archive volumes from StorHouse, you simply write lock the volumes to be exported and use the MOVE VOLUME command to transfer them from level L to shelf storage.

Volume Import and Catalog

The system administrator can merge one or more volumes with an existing volume set, import an entire volume set into a specific directory, or import one or more volumes into a specific library device using the IMPORT command. For example, the system administrator can import a previously exported volume into an existing volume set or the administrator can import a previously exported volume set into a StorHouse system that does not contain a volume set with the same name. IMPORT

requests the operator to load a volume or a volume set's volumes into a library and then adds them to the volume directory.

After the volumes have been imported, they are left in the uncataloged state. The system administrator must use the CATALOG VSET command to add file and file set information into the StorHouse directory for all uncataloged volumes. After cataloging the file and file set information, the command marks the volumes as cataloged, which makes them available for normal use. Figure 5-4 illustrates the steps required to import and export volumes and volume sets.

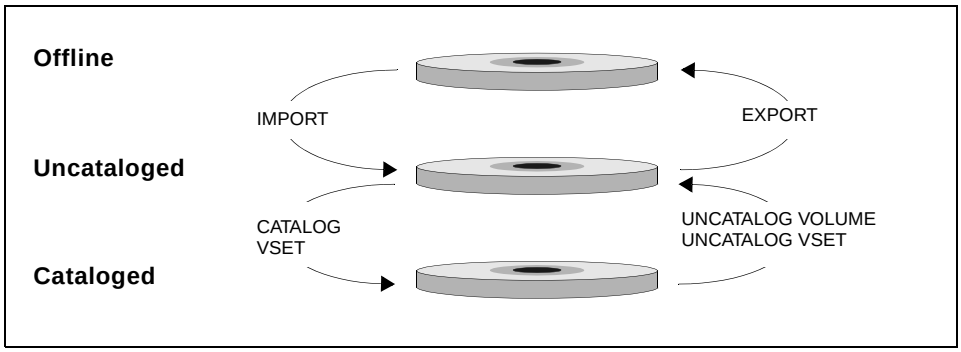

Figure 5-4: IMPORT/CATALOG and UNCATALOG/EXPORT Operations

Volume Erasure

StorHouse allows the system administrator to erase entire volumes and volume sets of volume-erasable media (in other words, erasable optical or tape media). Once erased, that storage space can be reallocated for use by the system. The administrator must perform a separate erase operation to erase existing data before new data can be written in its place. Once data has been erased, it cannot be recovered. The system administrator can use the following commands to erase data:

- ERASE VOLUME – Erases file data on a specific volume
- ERASE VSET – Erases file data on a specific volume set.

The volumes to be erased must be in the uncataloged state, either just imported or just uncataloged. (See the IMPORT and UNCATALOG VSET command descriptions in the "Volume Export and Import" section on page 5-21.)

When these commands first begin to erase a volume, they set the volume state to ERASING. If the command successfully erases the volume, the command places the erased volume in the CATALOGED state.

For erasable optical disk, the command physically erases and initializes (relabels) the volume. Any files that resided on the erased volume are completely and permanently removed. For magnetic tape, the command initializes (relabels) the tape volume to make the existing data inaccessible; subsequent writes of new data will overwrite the old data. All space on the erased volume (in other words, both sides of a two-sided volume, including space previously used for the erased files) is made available for allocation in the volume set. Erased volumes resemble volumes that have just been moved from the free pool into the volume set.

If either the ERASE VOLUME or ERASE VSET command is interrupted by a system failure and one or more volumes are left in the erasing state, the administrator must enter the command again.

To simplify the management of expired data, you should allocate files that expire at the same time to the same volume or volume set.

Volume Retirement

System administrators can use the RETIRE VOLUME command to:

- Manually retire removable volumes that are damaged, degraded, or have reached their lifecycle limit. This command is typically used for magnetic tape volumes, which degrade with use.
- With the /ERASE modifier, to reinitialize and reuse erasable, removable volumes. This command is typically used for erasable optical volumes.

StorHouse uses a mount limit for each media and recording type combination and a mount count for each volume to determine lifecycle limits.

- The *mount limit* indicates the average number of mounts that a volume of the media can undergo before it might encounter unrecoverable errors due to media degradation. The mount limit is zero for all optical media (non applicable) and non-zero for magnetic tape media.
- The *mount count* is the number of times a volume side has been used (mounted and dismounted).

The mount limit and mount count are recommended by the media manufacturer and initially set at StorHouse installation for each media type and recording type combination configured in the system.

Table 5-2 lists the RETIRE VOLUME modifiers.

Table 5-2: RETIRE VOLUME Command Modifiers

Modifier	Description	Default
/CONFIRM	Controls whether StorHouse asks you to confirm the command.	-
/DEADLINE=...	Specifies the date and time after the RETIRE VOLUME command is invoked when the command can start moving extents off selected volumes.	No start-time limit
/ERASE	Indicates that the command is to erase each volume instead of exporting it after moving all extents off the volume and uncataloging it.	If not specified, the command exports volumes instead of erasing them.
/REPORT	Controls the generation of special text responses for the completion of significant actions.	-
/WAIT	Instructs StorHouse to wait for a locked file to be unlocked before attempting to use it in the command execution.	-
/LIMIT=...	Specifies the maximum number of volumes to be selected by the command.	/LIMIT=1
/MAX_ERRORS	Controls the maximum number of errors StorHouse allows when it finds file extents that are not accessible on a volume. If the number of errors reaches this limit, StorHouse stops processing the current volume.	/NOMAX_ ERRORS
/MOUNTS=...	Controls whether volumes with a mount count greater than or equal to the number indicated by /MOUNTS are to be selected by the command.	/MOUNTS
/VSET=...	Indicates that the command will consider for selection only volumes in the volume set specified by this modifier.	-

RETIRE VOLUME works as follows. It:

- Deactivates one or more specified volumes
- Copies all non-deleted, accessible file extents from the specified volume(s) (both sides of a two-sided volume) to one or more other volumes in the same volume set
- Changes directory entries to point to the new file copies

- Uncatalogs the volume(s)
- Either exports or erases the volume(s).

During the retire volume process, the command removes rather than copies files marked for deletion and then proceeds to uncatalog and export the volume.

StorHouse selects a destination volume or volumes based on the availability of storage space in the appropriate file sets. StorHouse always deactivates the source volume (both sides of a two-sided volume) before attempting to allocate new space for the extents. This prevents the command from moving extents from one side to the other on a two-sided volume. StorHouse allocates space for the extents in appropriate file sets as necessary. StorHouse attempts to keep extents from the same file version on the same volume side.

If one or more extents have space allocated on the source volume, are on the performance buffer, and have not yet been written back to the source volume, the command allocates space for these extents on another volume and frees the space on the source volume.

If StorHouse cannot move all of the extents off the source volume, it returns an appropriate error condition to the user and moves as many of the extents as it can. Individual extent moves may fail due to read errors, lack of available space, locked files, or other similar conditions.

Volume Recovery

System administrators can use the RECOVER VOLUME command to create a replacement volume for one or more other volumes. RECOVER VOLUME selects volumes for recovery, creates a replacement volume for each selected volume, and uncatalogs and exports the volumes that have been replaced.

Although the administrator can recover enabled or disabled volumes, the primary purpose of volume recovery is to create a replacement volume for a volume that you cannot access, such as a broken, misplaced, or otherwise unreadable volume. A volume is unreadable when it is disabled. A *disabled volume* is a volume that your system administrator has marked as disabled using the StorHouse Command Language SET VOLUME /DISABLED command. Your system administrator must disable an unreadable volume before recovering it. Also, your administrator must have copies of the files on the volume you want to replace.

The copies of the files on a disabled volume may be located on a number of other volumes. The other volumes are likely stored offsite or on shelves. The /PREVIEW command modifier lets your system administrator identify the volumes needed to restore the selected original volume and display information about them without performing the recovery operation. This modifier displays information on the

selected original volume and on each other volume needed to recover the original. This information can be useful for disaster recovery planning.

Your system administrator can select specific volumes to be recovered using the RECOVER VOLUME modifiers shown in Table 5-3:

Table 5-3: RECOVER VOLUME Modifiers

Parameter Modifier	Description
/DEADLINE	Limits the length of time the command runs
/DIRECTORY=...	Indicates that the command will consider for selection only volumes in the directory specified by this modifier
/DISABLED	Determines whether the command uses the volume side state DISABLED to select volumes
/LAST_ALLOCATION	Limits the selected volumes to those with a last allocation date and time in the specified range
/LIMIT	Limits the number of volumes to be selected
/ORDER	Processes selected volumes in the specified order
/VSET	Limits the selected volumes to one particular volume set

In addition to recovering individual volumes, RECOVER VOLUME can be used in conjunction with the CHECKPOINT or EXTRACT DIRECTORY commands to restore an entire StorHouse system after a disaster. After using one of these two commands to restore directory information on a replacement or backup StorHouse system, your system administrator can use RECOVER VOLUME to create replacement volumes for all or selected volumes. StorHouse directory recovery and other more extensive recovery procedures are documented in *StorHouse Recovery Strategies*, publication number 900117. Call your FileTek customer support representative for a copy.

Chapter

6

The Account System

A StorHouse *account* is a collection of administrative data that is used to monitor and control the use of the StorHouse system by one or more users. StorHouse accounts are separate from the host system accounts.

One StorHouse account can be dedicated to a single user or shared by many users. Multiple users with the same or different accounts can sign on to and use StorHouse concurrently.

StorHouse account information is kept in the account file. Each account has an account identification code (aid) and a password to protect the account from unauthorized use. The aid and password must be entered by a user and are validated by the system as part of the sign-on procedure. Various usage statistics and default values for file access group, volume set, file set, and privileges are also kept for each account.

A user account can be disabled to prevent users from signing on to the account. File access controls and privileges can be changed and statistics gathered for an account while it is disabled or enabled. A disabled account can be enabled to allow users to sign on.

Account Identification Codes

An *account identification code* (aid) contains 1 to 12 characters and consists of the following characters: A-Z (uppercase), 0-9, $ (dollar sign), and _(underscore). StorHouse always translates account identification codes to uppercase characters.

Account Passwords

Account passwords can be null or contain up to 32 characters, and consist of the following ASCII characters: A-Z (uppercase), 0-9, $ (dollar sign), and _ (underscore). StorHouse always translates account passwords to uppercase characters.

Account Privileges

Each account has a set of privileges. There are two types of account privileges:

- *Access* privileges allow an account user to bypass various security checks.
- *Command* privileges allow an account user to perform specific commands and functions, or groups of commands and functions.

Access privileges and command privileges do not override each other. That is, you must have both the required command and access privileges (or pass all required security checks) to execute commands.

The access and command privileges are listed and defined in Appendix B.

Account Access Levels

The account information that you can display or modify depends on your level of access. There are three levels of access to StorHouse accounts:

- *Account owner* access, which all users have by default, allows you to display information and modify some parameters for your own account.
- *Show account* access, which is provided by the SHOACCOUNT privilege, the ANYACCOUNT privilege, or the ALLPRIVILEGE privilege, allows you to display account information for any account.
- *Any account* access, which is provided by the ANYACCOUNT privilege or the ALLPRIVILEGE privilege, allows you to display and modify account information for any account.

Default Environment

When you sign on to StorHouse, the system assigns a *default environment* to your session. The default environment defines default values for the access group name, access rights to the default file access group, volume set, and file set associated with your account.

StorHouse uses the default names when processing a user command if you do not specify alternate names in the command. Access rights specify the default access type(s) that you have to the default group upon signon. Note that the default access rights do not prevent access to the default group. They allow you to gain access without specifying passwords.

You can temporarily change the account's default file access group, volume set, or file set by using the SET USER command. Any changes become effective immediately and remain in effect until the end of your current session or until you change them again.

Chapter

7

System Backup, Recovery, and Error Reporting

This chapter discusses system file backups, system recovery, and the Call Home error reporting feature.

System File Backup

System files are StorHouse operating system files that contain StorHouse account information, directory information, system parameters, and other data used to control the system. *System file backups* are copies of the current system files. The backup copies are used to reconstruct the current files in the unlikely event the current files become damaged. StorHouse backs up system files automatically. It also allows system administrators to manage backups manually.

Automatic Backup

StorHouse supports two types of automatic system file backup. The system automatically maintains *shadow copies* of all StorHouse system files (including directory files) on a different magnetic disk. Whenever StorHouse updates the primary system files, it also updates the shadowed copies. In addition to shadow copies, the system records primary directory information on removable volumes whenever it writes a file to a volume.

Manual System File Backup

StorHouse supports checkpoints and directory extractions to backup up system files manually. The following sections explain these features.

System File Checkpoints

The *checkpoint* feature enables your system administrator to use the CHECKPOINT command to take a snapshot of StorHouse system files and write them to a checkpoint volume set. Normal user operations can continue during a checkpoint operation.

Table 7-1 lists the system parameters that control checkpointing.

Table 7-1: Checkpoint System Parameters

Parameter	Description
CHKP_ACCOUNT	Specifies the StorHouse account that will be used to write checkpoint information.
CHKP_FSET	Specifies the checkpoint file set.
CHKP_GROUP	Specifies the file access group that will be used for checkpoint files.
CHKP_LIMIT	Specifies the maximum number of checkpoints to maintain on the checkpoint volume set.
CHKP_ON	Enables or disables the system's ability to perform a checkpoint.
CHKP_TAKEN	Specifies the date and time of the last checkpoint.
CHKP_UPD_NOW	Specifies whether StorHouse will write any changed account or group information directly to the checkpoint volume set as it is updated.
CHKP_VSET	Specifies the checkpoint volume set.

For more information on the system parameters that control checkpointing, see the *System Administrator's Guide*.

StorHouse ensures that the latest checkpoint volume or volumes (individual checkpoint operations may span volumes) are always in the library device and not on a shelf because system checkpoint files may need to be written at any time. The system administrator should not use the MOVE VOLUME or MOVE VSET command to move the checkpoint volumes out of the library device except to replace them with new volumes.

System File Extractions

The *extraction* feature enables your system administrator to use the EXTRACT DIRECTORY command to create StorHouse *extraction files*. These files contain information from the PRIMARY, BACKUP, and ARCHIVE directories of one or more

source StorHouse systems. StorHouse writes extraction files to removable volumes in a StorHouse volume set.

StorHouse supports full and incremental directory extractions. A *full extraction*, which is the default, saves account, group, file set, volume set, removable volume, and removable volume file extent information for all volume sides. An *incremental extraction* adds to the information saved by previous full extractions by creating additional extraction files with significant changes (for example, new or newly removed extents) since the last extraction. (A modification to a file's last access date and time is an example of a change included in a full extraction but not in an incremental extraction.)

Each successful directory extraction creates one *extraction control file* and one *extraction VTOC file* for each volume side for which file information is extracted. The contents of the control file includes the groups, volume sets, file sets, volumes, accounts, and the names of all VTOC files associated with the extraction. The VTOC file identifies the file extents stored on the volume and their associated file version, group, file set, and volume set information.

The EXTRACT DIRECTORY command is used with the RESTORE DIRECTORY command to implement directory recovery from extraction files. FileTek must perform this type of directory recovery for you.

System Recovery

Normally, StorHouse attempts to recover from error conditions automatically. However, if the system does not recover during a restart after a crash, your system administrator should contact FileTek customer support to diagnose the problem and discuss available system recovery options and procedures.

StorHouse supports two types of system recovery:

- Normal
- Extended.

The following sections provide more information about these recovery types.

Normal Recovery

A *normal* recovery occurs when the StorHouse system goes down in an uncontrolled manner but no directory information is lost. In such a case, the system will automatically restart and recover on its own.

After an uncontrolled shutdown, all volumes to be recovered must be accessible to StorHouse. If the system cannot access a volume (for example, the volume is in an offline drive), StorHouse shuts down after displaying an error message.

If a power outage occurs or the software experiences a failure while a system file is being updated, StorHouse attempts to log an error message and display it at the system console before shutting down. The operator must restart the system to initiate recovery. If the corrupted system file has a shadow copy, the system uses the shadow copy to recover during the next startup. If there is no shadow copy or the system is unable to recover, it attempts to log an error message and display it at the system console again before shutting down. In this case, your system administrator should call FileTek customer support for assistance.

Extended Recovery

Extended recovery restores one or more system files that were not recovered during normal recovery. It can be used to restore directory information (*directory recovery*) at a site that is otherwise operational (media, data, and hardware are intact) or to restore a site that was completely destroyed or made inaccessible because of a disaster (*disaster recovery*).

StorHouse supports four extended recovery methods as shown in Table 7-2. FileTek must assist you with each of these methods.

Table 7-2: Extended Recovery Methods

Method	Recovers Information	For files on
Checkpoint recovery	From checkpoint files on a StorHouse volume set	Levels L and S
Directory recovery from extraction files	From extraction files on a StorHouse volume set	Levels L and S
Directory recovery from physical volumes	From the volume table of contents (VTOC) on level L and level S volumes	Levels L and S
Device scan recovery	By scanning the Centera device to be recovered	Level F

FileTek recommends that you create an extended recovery plan that best meets the needs of your organization. This plan will detail how you prepare for recovery, including which type(s) of recovery you will perform, which commands you will execute to take a snapshot of system files, how frequently you will execute them, and where you will store your checkpointed or extracted information. FileTek is available to help you develop this recovery plan.

Checkpoint Recovery

Checkpoint recovery uses “checkpoint” files created with the CHECKPOINT command prior to the data loss to recover one or more system files that were not restored during normal recovery. To perform checkpoint recovery, your system administrator must have enabled the checkpoint capability on your system and issued CHECKPOINT commands at regular intervals.

Figure 7-1 illustrates the checkpoint recovery process.

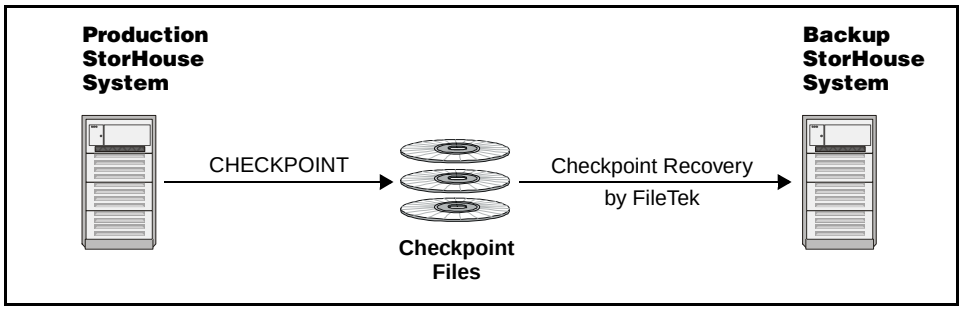

Figure 7-1: Checkpoint Recovery

Directory Recovery From Extraction Files

Directory recovery from extraction files restores StorHouse directory information from extraction files that your system administrator created prior to the loss. *Extraction files* contain volume set, file set, volume, file access group, file version, file extent, and account information from the directories of each source StorHouse system. Your system administrator creates extraction files by periodically issuing the EXTRACT DIRECTORY command on one or more StorHouse systems. Should the need arise, FileTek will execute the RESTORE DIRECTORY command, which uses information in the directory extraction files to restore lost directories from one or more source StorHouse systems to a backup StorHouse system.

Figure 7-2 illustrates directory recovery from extraction files.

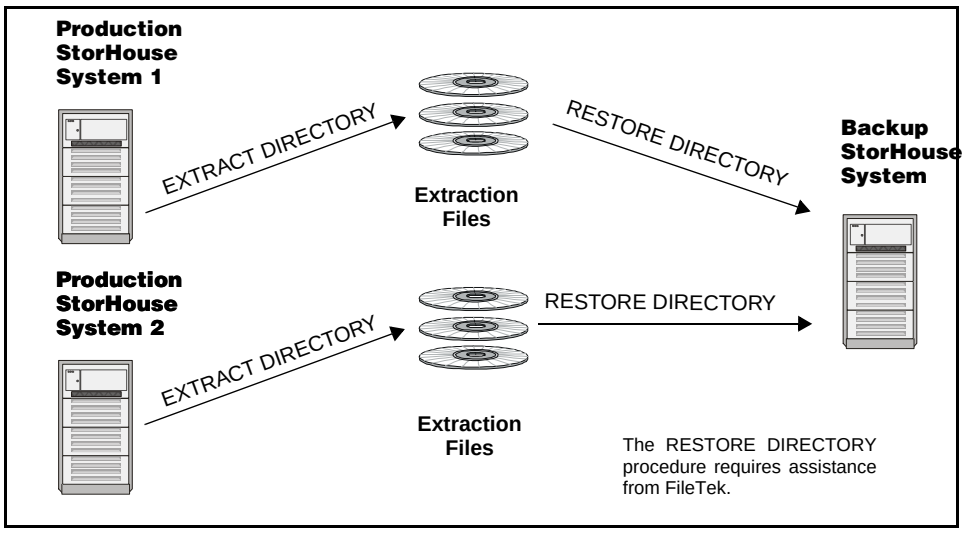

Figure 7-2: Directory Recovery from Extraction Files

Should your site need to restore directory information, you need to restore only the volumes containing the extraction files. You do not need to subsequently import and catalog other primary volumes or backup or archive copies of all the volumes that were in the affected system. Because extraction files contain any changes made to the directories after the data volumes were written, they may provide more up-to-date information than the directory information that would be recovered from physical volumes.

Directory Recovery From Physical Volumes

Directory recovery from physical volumes restores directory information for level L and/or level S volumes from the volume table of contents (VTOCs) of your removable volumes. This recovery procedure is typically used only in the rare event that directory files cannot be recovered using any other method.

It is not necessary to recover the contents of all optical or tape volumes in the system (level L and level S) at one time. You can regain operational use of StorHouse by recovering and cataloging the volumes you need to access first. The remaining volumes can be added to the system as necessary to complete the total recovery.

During this type of recovery, FileTek customer support personnel reinitialize (clear all files from) the magnetic disks or RAID containing system files. If any of your user files on magnetic disks were not backed up from the performance buffer to their resident file sets, they cannot be recovered. Once the magnetic disks are reinitialized, FileTek personnel rebuild system files and set up empty directories.

Device Scan Recovery

Device scan recovery supports directory and disaster recovery of Centera level F devices. It scans the level F device to be recovered to obtain information about the StorHouse files, extents, and file sets on the device. Then it catalogs those files, extents, and file sets. FileTek must assist customers with device scan recovery, which is initiated with the CATALOG DEVICE command.

Using Device Scan Recovery for Disaster Recovery. To prepare for disaster recovery of a level F device, the device at the primary site must replicate data to a similar device at the backup site. This requires a network connection of sufficient bandwidth. Because level F devices that support replication write files to a backup site almost continuously, device scan recovery restores files almost to the point of the disaster.

Device scan recovery is independent of existing disaster recovery procedures for level L optical and tape volumes. StorHouse systems with level F and level L devices must perform disaster recovery procedures for both storage levels. If checkpoint recovery is used to recover level L, it must be completed before level F device scan recovery begins.

Call Home Error Reporting

Call Home error reporting automatically identifies error conditions at customer sites, reports them to the FileTek Support Center at headquarters, and notifies FileTek customer support personnel, as appropriate. Call Home increases StorHouse system availability by improving FileTek customer support response time, which in turn reduces unscheduled downtime.

Upon detection of a problem, the Call Home procedure at a customer site transmits a message containing the error event information to the FileTek Support Center. When the message arrives at FileTek, the StorHouse software analyzes the information and automatically pages the appropriate customer support representative, depending on the severity of the error.

Call Home responds to the following types of events and conditions:

- Predictive maintenance
- Critical disk shortages
- Termination of a critical StorHouse program
- Unrecoverable device errors
- Excessive (recovered) device errors
- Disabled volumes.

The system administrator must set the CALL_HOME system parameter to TRUE to enable the Call Home feature. If CALL_HOME is set to FALSE, the system does not generate automatic calls upon identifying error conditions.

Appendix

A

ASCII Characters

StorHouse uses 8-bit ASCII characters to code character data. The printable ASCII characters are the subset of ASCII characters that normally appear as printed characters. The printable characters are defined in Table A-1. The ASCII characters and their hexadecimal values are listed in Table A-2 on the following pages.

Table A-1: Printable ASCII Characters

Printable ASCII Characters					
A-Z	uppercase letters	+	plus sign	( )	parentheses
a-z	lowercase letters	~	tilde	< >	angle brackets
0-9	digits	,	comma	[ ]	square brackets
!	exclamation point	-	hyphen	{ }	braces
“	quote	.	period	\	backslash
#	number sign	/	slash	^	circumflex
$	dollar sign	:	colon	_	underscore
%	percent sign	;	semicolon	|	vertical bar
&	ampersand	=	equal sign	‘	reverse apostrophe
’	apostrophe	?	question mark		space
*	asterisk	@	at sign		

Table A-2: ASCII Characters and Hexadecimal Values

Value	Character	Value	Character
00	NUL	20	(space)
01	SOH	21	! (exclamation point)
02	STX	22	“ (quote)
03	ETX	23	# (number sign)
04	EOT	24	$ (dollar sign)
05	ENQ	25	% (percent sign)
06	ACK	26	& (ampersand)
07	BEL	27	’ (apostrophe)
08	BS	28	((parenthesis)
09	HT	29	) (parenthesis)
0A	LF	2A	* (asterisk)
0B	VT	2B	+ (plus sign)
0C	FF	2C	, (comma)
0D	CR	2D	- (hyphen)
0E	SO	2E	. (period)
0F	SI	2F	/ (slash)

Table A-2: ASCII Characters and Hexadecimal Values (continued)

Value	Character	Value	Character
10	DLE	30	0
11	DC1	31	1
12	DC2	32	2
13	DC3	33	3
14	DC4	34	4
15	NAK	35	5
16	SYN	36	6
17	ETB	37	7
18	CAN	38	8
19	EM	39	9
1A	SUB	3A	: (colon)
1B	ESC	3B	; (semicolon)
1C	FS	3C	< (angle bracket)
1D	GS	3D	= (equal sign)
1E	RS	3E	> (angle bracket)
1F	US	3F	? (question mark)

Table A-2: ASCII Characters and Hexadecimal Values (continued)

Value	Character	Value	Character
40	@ (at sign)	60	‘ (reverse apostrophe)
41	A	61	a
42	B	62	b
43	C	63	c
44	D	64	d
45	E	65	e
46	F	66	f
47	G	67	g
48	H	68	h
49	I	69	i
4A	J	6A	j
4B	K	6B	k
4C	L	6C	l
4D	M	6D	m
4E	N	6E	n
4F	O	6F	o

Table A-2: ASCII Characters and Hexadecimal Values (continued)

Value	Character	Value	Character
50	P	70	p
51	Q	71	q
52	R	72	r
53	S	73	s
54	T	74	t
55	U	75	u
56	V	76	v
57	W	77	w
58	X	78	x
59	Y	79	y
5A	Z	7A	z
5B	[(square bracket)	7B	{ (brace)
5C	\ (backslash)	7C	| (vertical bar)
5D	] (square bracket)	7D	} (brace)
5E	^ (circumflex)	7E	~ (tilde)
5F	_ (underscore)	7F	DEL

A ASCII Characters

Appendix

B

Access and Command Privileges

Each StorHouse Command Language command requires you to have certain types of access and command privileges. *Access privileges* allow you to bypass security checks. *Command privileges* allow you to perform specific commands and functions or groups of commands and functions. The commands are described in the *Command Language Reference Manual*.

The following tables describe access and command privileges. Table B-1 describes the available access privileges. Table B-2 describes the available command privileges. Having the privilege ALLPRIVILEGE is equivalent to having all access and command privileges; therefore, it appears in both tables.

Table B-1: Access Privileges

Privilege	Gives the account user access to:
ALLPRIVILEGE	All access and command privileges. (This is also a command privilege.)
ANYACCOUNT	Use any account in order to create, remove, modify, or show account information.
ANYFILE	Any file in an accessible group without having to specify a file password. This privilege does not bypass group passwords. The user's types of access to a file are limited to the types of access that the user has to the file's group.
ANYGROUP	Any file access group without having to specify a group password. This privilege does not bypass file passwords.
SHOACCOUNT	Any account for the purpose of showing account information. It does not allow the user to change anything.
SHOFILE	Any file in an accessible group for the purpose of showing file directory information. This privilege does not bypass access group passwords.
SHOGROUP	Any file access group for the purpose of showing group information.
SQLADMIN	Database administration (DBA) privilege to access all StorHouse databases. Any account assigned SQLADMIN access privilege is automatically granted DBA privilege and can grant SCAN privilege to other accounts.

Table B-2: Command Privileges

Privilege	Allows the account user to:
ACCOUNT	Execute commands that create or remove accounts, show account information, or modify account parameters and privileges.
ALLOCATION	Execute commands that create, modify, or delete storage resources.
ALLPRIVILEGE	Execute any StorHouse command. Having this privilege is equivalent to having all access and command privileges. (This is also an access privilege.)
ATF	Execute commands that specify ATF values for files.
CONSOLE	Execute commands that enable or disable the display of StorHouse operator messages to the user's own terminal.
COPY	Execute commands that copy files within StorHouse.
DELETE	Execute commands that delete files or change file attributes that control file deletion.
FILE	Execute commands that modify file attributes or show file information.
GET	Execute the GET command and to open a VRAM file for reading and, when used in conjunction with PUT privilege, for updating.
GROUP	Execute commands that create or remove groups, show group information, or modify group passwords.
LOCK	Execute commands that lock and unlock files, or show which files are locked.
MESSAGE	Execute the MESSAGE command.
OPERATOR	Execute commands that control system, user, and device activity. Also allows the user to reply to operator messages.
PASSWORD	Enter commands that change file or access group passwords.
PUT	Execute the PUT command and to open a VRAM file for writing and, when used in conjunction with the GET privilege, for updating.
RECORD	Execute commands for VRAM files.
SCHEDULE	Execute commands that schedule other commands to be executed as background transactions, remove schedule commands from the schedule, or show scheduled commands.
SERVICE	Execute the SERVICE command.
SETGROUP	Enter commands that use an access group other than the account's default access group. The account user still must obtain access to the specified group; this privilege does not bypass group passwords.
SHOW	Execute any of the SHOW commands.
SQLCOMMAND	Issue the StorHouse Command Language EXECUTE STH_LOAD command.
SQLEXECUTE	Submit SQL statements in StorHouse/RM.

Table B-2: Command Privileges (continued)

Privilege	Allows the account user to:
SYSTEM	Execute commands that modify or display system parameters, or initiate system backup or file migration activities.
VTF	Execute commands that specify values for backup control and VTF file attributes.

Index

Symbols

$$BUFFER file set (performance buffer) 4-13

A

access group
- name 3-2
- passwords 3-20

access levels for accounts
- account owner 6-2
- any account 6-2
- show account 6-2

access methods for files
- KEYED 3-17
- RECORD 3-17
- SEQUENTIAL 3-17

access modes for files
- APPEND 3-18
- READ 3-18
- UPDATE 3-18
- WRITE 3-18

access privilege, definition 6-2, B-1

access privileges
- ALLPRIVILEGE B-1
- ANYACCOUNT B-1
- ANYFILE B-1
- ANYGROUP B-1
- SHOACCOUNT B-1
- SHOFILE B-1
- SHOGROUP B-1
- SQLADMIN B-1

Access Time Factor (ATF) attribute 5-14

accessing the duplex copy of a file extent
- for load balancing 5-6
- in place of a disabled primary volume 5-5
- in place of an offline primary volume 5-6

accessor 2-3

account
- access levels 6-2
- default access group and rights 6-3
- default environment 6-3
- default file set 6-3
- default volume set 6-3
- definition 6-1
- disabled 6-1
- enabled 6-1
- identification code (aid) 6-1
- owner access 6-2
- passwords 6-2
- privileges 6-2
- system 1-3

ACCOUNT command privilege B-2

ACSLS 2-18

administration log 1-7

aid (account identification code) 6-1

allocating
- blank volumes 4-20
- empty volumes 4-21
- file sets 4-19

files 4-19
volume sets 4-19

ALLOCATION command privilege B-2

ALLPRIVILEGE
access privilege B-1
command privilege B-2

any account access 6-2

ANYACCOUNT access privilege B-1

ANYFILE access privilege B-1

ANYGROUP access privilege B-1

APPEND file access mode 3-18

application program interface level 1-5

ARCHIVE command 5-4

archive file directory 3-13

archiving user files 5-4

assigning a replication class 5-11

ATF
(Access Time Factor) attribute 5-14
command privilege B-2
system parameter 5-14

attributes
AUTO_STAGE 5-20
BACKUP 5-2
file set 4-10
HOLD 5-21
volume set 4-6

AUTO_STAGE attribute 5-20

AUTO_STAGE attribute (file set) 4-12

B

BACKUP
command 5-4
file attribute 5-2
system parameter 5-2

backup
duplex for user files 5-4
file directory 3-13
system files 7-1
user files 5-3

bar code label for tape volumes 2-11

bar code reader 2-3, 2-16

basic StorHouse interface level 1-5

blank volume
allocating 4-20
definition 4-20

BUFFERED file extent status 4-14

C

C (change) extent 3-9, 3-11

calculating file retention periods 5-7

Call Home error reporting 7-7

CALL_HOME system parameter 7-7

Callable Interface 1-5

catalog
volume 5-21
volume set 5-22

CATALOG DEVICE command 7-7

CATALOG VSET command 5-23

CATALOGED volume side state 5-18

CATALOGING volume side state 5-18

CHANGE record access function 3-19

change table 4-18

CHECKPOINT
command 7-2
file function 3-17

checkpoint table 4-17

checkpoint/recovery for system files 7-2, 7-5

CHKP_ACCOUNT system parameter 7-2

CHKP_FSET system parameter 7-2

CHKP_GROUP system parameter 7-2

CHKP_LIMIT system parameter 7-2

CHKP_ON system parameter 7-2

CHKP_TAKEN system parameter 7-2

CHKP_UPD_NOW system parameter 7-2

CHKP_VSET system parameter 7-2

CLOSE file function 3-17

command privilege, definition 6-2, B-1

command privileges
- ACCOUNT B-2
- ALLOCATION B-2
- ALLPRIVILEGE B-2
- ATF B-2
- CONSOLE B-2
- COPY B-2
- DELETE B-2
- FILE B-2
- GET B-2
- GROUP B-2
- LOCK B-2
- MESSAGE B-2
- OPERATOR B-2
- PASSWORD B-2
- PUT B-2
- RECORD B-2
- SCHEDULE B-2
- SERVICE B-2
- SETGROUP B-2
- SHOW B-2
- SQLCOMMAND B-2
- SQLEXECUTE B-2
- SYSTEM B-3
- VTF B-3

commands
- ARCHIVE 5-4
- BACKUP 5-4
- CATALOG DEVICE 7-7
- CATALOG VSET 5-23
- CHECKPOINT 7-2
- CONSOLE 1-5
- CREATE BACKUP 5-3
- CREATE FILE 4-16
- CREATE FSET 5-18, 5-20
- CREATE PRIMARY 5-12
- CREATE VSET 4-8
- ENABLE 4-15
- ERASE VOLUME 5-19, 5-23
- ERASE VSET 5-23
- EXPORT 5-22
- IMPORT 5-22
- LOCK 3-21
- MESSAGE /LOG 1-7
- MIGRATE 5-14, 5-15
- MIGRATE /BY_VSET 5-16
- MOVE VOLUME /MEMO 4-5
- MOVE VSET /MEMO 4-9
- NEWLOG 1-7
- PUT 4-15, 5-8
- RECOVER VOLUME 5-26
- RELOCATE 5-18
- REMOVE FILE 5-13
- REPLICATE 5-12
- RETIRE VOLUME 5-24
- SET FILE 5-3
- SET FSET 5-20
- SET USER 1-5
- SET VOLUME 4-4, 4-8
- SET VOLUME /CLEANED 4-5
- SET VOLUME /DISABLE 5-6
- SET VOLUME /ENABLE 5-6
- SET VOLUME /MEMO 4-5
- SET VOLUME /UNWRITELOCKED 4-5
- SET VSET 4-8
- SET VSET /MEMO 4-9
- SHOW FSET 5-20
- SHOW USER 1-5
- SHOW VOLUME 4-5
- SHOW VOLUME /FULL 4-8
- SHOW VSET 4-9
- SHOW VSET /FULL 4-8
- UNCATALOG VOLUME 5-19, 5-22
- UNCATALOG VSET 5-22

CONSOLE
- command 1-5
- command privilege B-2

contiguous space for file sets
    extensions 4-11, 4-22
    initial allocation 4-11, 4-22

COPY command privilege B-2

CREATE BACKUP command 5-3

CREATE FILE command 4-16

CREATE FSET command 5-18, 5-20

CREATE FSET/FORCE_RETENTION 4-13

CREATE PRIMARY command 5-12

CREATE VSET command 4-8

CREATE-OPEN file function 3-17

CYCLE attribute 5-18

cycle time
    volume sets 4-8
    volumes 4-4

D

D (data) extent 3-8, 3-10, 3-11

Data Facility Data Set Services (DFDSS) 3-5

data record attributes for transportable files 3-15

data record format 3-7

data table 4-17

DEACTIVATED volume side state 5-18

deactivation time
    volume sets 4-8
    volume sides 4-3

deallocating
    file sets 4-24
    files 4-24
    volume sets 4-23

dedicated device 2-16

default environment for accounts 6-3

deferred dynamic system parameters 1-6

definitions
access group name 3-2
access mode 3-18
access privilege B-1
access time factor 5-14
accessor 2-3, 2-13
account 6-1
account identification code (aid) 6-1
administration log 1-7
bar code reader 2-3
blank volume 4-20
command privilege B-1
contiguous space allocation 4-11
deactivation time 4-3
dedicated device 2-16
deferred dynamic system parameter 1-6
device identification code 2-1
device scan recovery 7-7
disabled volume 4-5, 5-26
drive 2-3
dynamic system parameter 1-6
empty volume 4-20
exchange station 2-3
expiration time 4-4
explicit lock 3-21
extent sequence number 3-3
external key 3-6
extraction files 7-2, 7-5
file 3-1
file access group 3-1
file extent 3-3
file identifier 3-1
file name 3-2
file number 3-1
file replication class name attribute 5-11
file revision 3-2
file set 4-9
file set attribute 4-10
file set name 4-10
file set replication class name attribute 4-13
file staging 5-20
file version 3-2
frame 3-4
free pool 4-20
general space 4-12
header (record) 3-4

HOLD attribute 4-7
implicit lock 3-22
internal key 3-6
key 3-6
LIMIT attribute (file set) 4-11
LIMIT attribute (volume set) 4-6
logical volume 4-1
MAGDISK 4-9
media attribute 4-7
media type 2-4
migration factor 5-14
mount count 5-24
mount limit 5-24
noncontiguous space allocation 4-11
performance buffer 2-12
physical volume 2-3, 4-1
record 3-4
record segment 3-4
recording type 2-4
relative file revision 3-3
relative file version 3-2
replicated file 5-10
replication class 4-13, 5-11
retained file 5-7
retention attribute (file set) 4-12
retention attribute (file) 5-7
retention period 5-7
robotic arm 2-3
session 1-3
shadow copy 7-1
side indicator 2-11
site identifier 4-1
slot 2-3
stage 5-20
static system parameter 1-6
storage allocation attribute 4-11
subunit 2-2
subunit number 2-3
system identifier 3-1, 4-1
system parameter 1-6
transport arm 2-3
truncated file 3-3
unit number 2-2
update space 4-12
upward migration 5-20
usage factor 5-15
user file replication 5-10
user log 1-7
volume identification code 2-4
volume label 2-10
volume label record 4-1
volume set 4-6
volume set attribute 4-6
volume set name 4-6
vulnerability time factor (VTF) 5-3
write-back 5-4

DEL_FILE_PERM system parameter 5-13

DELETE
command privilege B-2
record access function 3-19

delete password
access group 3-20
file 3-21

deleted file directory 3-12

device
accessor 2-3
bar code reader 2-3
drive 2-3
exchange station 2-3
fixed 2-2
identification code (did) 2-1
levels 2-2
library 2-2
network 2-2
removable 2-2
shelf 2-2
slot 2-3
subunit number 2-3
subunit type 2-3
unit number 2-2

device scan recovery 7-7

DF (definitions) extent 3-9, 3-10, 3-11

DFDSS (Data Facility Data Set Services) 3-5

did (device identification code) 2-1

DIRECT value for VTF attribute 5-3

directory
- archive 3-13
- backup 3-13
- deleted 3-12
- extent information 3-16
- file information 3-14
- functions 3-16
- group information 3-13
- primary 3-13
- version information 3-14

directory recovery
- from extraction files 7-5
- from physical volumes 7-6
- through device scan recovery 7-7

disabled volume 4-5, 5-26

DISABLED volume side state 5-18

DISABLED_FLAG 5-11

Disk File Transfer Interface 1-4

drive 2-3

duplex feature for user files 5-4

DUPLEX_BALANCE system parameter 5-6

DUPLEX_DIRECTORY system parameter 5-5

DUPLEX_ENABLE system parameter 5-5, 5-6

dynamic system parameters 1-6

E

empty volume
- allocating 4-21
- definition 4-20

ENABLE command 4-15

end-user interface level 1-4

erasable optical media 2-4

ERASE VOLUME command 5-19, 5-23

ERASE VSET command 5-23

erasing
- volume 5-23
- volume set 5-23

ERASING volume side state 5-18

exchange station 2-3, 2-17

expiration time
- volume set 4-8
- volume sides 4-4

explicit lock 3-21

EXPORT command 5-22

exporting a volume 5-22

extended recovery for system files 7-4

extending a file set 4-22

extent (file)
- definition 3-3
- information for file versions 3-16
- sequence number 3-3
- set 4-15
- statuses 3-16
- types 3-8

extent statuses
- BUFFERED 4-14
- LAST 4-14
- NEW 4-14
- NONE 4-14
- UPDATE 4-14
- WRITEBACK_DISABLED 4-14

extent types
- C (change) 3-9, 3-11
- D (data) 3-10, 3-11
- DF (definitions) 3-9, 3-10, 3-11
- K (key data base) 3-9, 3-10
- M (map) 3-9, 3-11
- U (update) 3-9, 3-11

external key 3-6

extraction files 7-2

extraction files, definition 7-5

F

fid (file identifier) 3-1

file
- access functions 3-17
- access group 3-1
- access methods 3-17
- access modes 3-18
- archive 5-4
- attributes 5-3
- backup 5-3
- definition 3-1
- descriptor flags 3-15
- directories 3-12
- directory information 3-14
- duplex feature for user files 5-4
- extent 3-3
- extent set 4-15
- extent statuses 4-14
- extent types 3-8
- format 3-7
- frame 3-4
- identifier (fid) 3-1
- lock 3-21
- migration between volume sets 5-16
- migration from performance buffer 5-14
- name 3-2
- number (fno) 3-1
- organizations 3-5
- passwords 3-21
- placement in a file set 4-23
- record 3-4
- recovery 5-12
- removal 5-13
- replicated file 5-10
- replication 5-10
- residence 4-14
- retained file 5-7
- retention 5-7
- revision 3-2
- size 4-15
- staging 5-20
- status 3-15
- storage 4-14
- storage deallocation 4-24
- transfer functions 3-19
- transportable 3-5
- version 3-2

file access functions
- CHECKPOINT 3-17
- CLOSE 3-17
- CREATE-OPEN 3-17
- OPEN-SEQ 3-17
- OPEN-VRAM 3-17

file access methods
- KEYED 3-17
- RECORD 3-17
- SEQUENTIAL 3-17

file access modes
- APPEND 3-18
- READ 3-18
- UPDATE 3-18
- WRITE 3-18

file attributes
- ATF 5-14
- BACKUP 5-2
- LIMIT 5-13
- replication class name 5-11
- retention 5-7
- VTF 5-3

FILE command privilege B-2

file directories
- archive 3-13
- backup 3-13
- deleted 3-12
- primary 3-13

file extent statuses
- BUFFERED 4-14
- LAST 4-14
- NEW 4-14
- NONE 4-14
- UPDATE 4-14
- WRITEBACK_DISABLED 4-14

file extent types
- C (change) 3-9
- D (data) 3-8

DF (definitions) 3-9
K (key data base) 3-9
M (index map) 3-9
U (update) 3-9

file organizations
KEYED 3-6
KEYSEQUENTIAL 3-7
RECORD 3-6
SEQUENTIAL 3-5, 3-10
STORHOUSE 3-7, 3-11
VRAM 3-10

file set
$$BUFFER 4-13
allocating space 4-22
attributes 4-10
AUTO_STAGE attribute 4-12
automatic staging 4-12
contiguous space
extensions 4-22
initial allocation 4-22
deallocating space 4-24
definition 4-9
extending 4-22
file extent placement 4-23
forcing retention 4-13
general space 4-12
level F 4-13
LIMIT attribute 4-11
name 4-10
noncontiguous space
extensions 4-22
initial allocation 4-22
performance buffer 4-13
placement in volume set 4-21
replication class name attribute 4-13
retention attribute 4-12
size 4-11
specifying retention 4-12
storage allocation attribute 4-11
update attribute 4-12
update space 4-12

File System, StorHouse 1-6

file transfer functions
GET 3-19
PUT 3-19
XFER 3-19

files with keys 4-17

fno (file number) 3-1

formats, media
magnetic disk volume 4-2
magnetic tape volume 4-2
optical disk volume 4-2

frame
definition 3-4
information for files 3-15

Frame Pointer Table (FPT) 3-8

free pool 4-7, 4-20

FREE_POOL_didmmr system parameter 4-20

G

general space for file sets 4-12

GET
command privilege B-2
file transfer function 3-19

group
definition 3-1
directory information 3-13

GROUP command privilege B-2

H

header (record) 3-4

hexadecimal values A-2

HOLD attribute
volume sets 4-7
volumes 4-3, 5-21

I

identifiers
site 1-6, 4-1

system 1-6, 4-1

implicit lock
definition 3-22
read-lock 3-22
write-lock 3-22

IMPORT command 5-22

importing a volume 5-22

interfaces
application program 1-5
basic StorHouse 1-5
Callable 1-5
Disk File Transfer 1-4
end-user 1-4
network support 1-5

internal key 3-6

K

K (key data base) extent 3-9, 3-10

key 3-6

key index table 4-17

key name table 4-17

key segment location table 4-17

key types
external 3-6
internal 3-6

KEYED
file access method 3-17
file organization 3-6

KEYSEQUENTIAL file organization 3-7

keysequential files 4-17

L

LAST file extent status 4-14

level F
file sets 4-13
volume set 4-9

levels of storage
F (fixed) 2-2
L (library) 2-2
N (network) 2-2
S (shelf) 2-2

levels, device 2-2

library device
attribute 4-7
malfunction 2-17
optical disk 2-12
subunits 2-15

LibraryStation 2-18

LIMIT attribute
file 5-13
file set 4-11
volume set 4-6

LIMIT system parameter 5-13

LINKNAME 5-11

load balancing 5-4, 5-6

LOCK
command 3-21
command privilege B-2

locks
explicit 3-21
implicit 3-22

logical volume 4-1

logs
administration 1-7
user 1-7

LSMCO Callable Interface function 5-8

LSMOS Callable Interface function 5-8

M

M (map) extent 3-9, 3-11

MA media type and recording types 2-5

MAGDISK volume set 4-9

magnetic disk
- drives 2-11
- volume formats 4-2

magnetic tape
- drives 2-16
- library device 2-15
- volume formats 4-2

MB media type and recording types 2-5

MC media type and recording types 2-5

MD media type and recording types 2-5

ME media type and recording types 2-5

media and recording types
- MA 2-5
- MB 2-5
- MC 2-5
- MD 2-5
- ME 2-5
- MF 2-5
- MG 2-6
- MIA 2-6
- MIB 2-6
- MJA 2-6
- MJB 2-6
- NA 2-10
- NC 2-10
- OA 2-6
- OC 2-6
- OE 2-7
- PA 2-7
- TB 2-8
- TC 2-8
- TD 2-8
- TE 2-9
- TF 2-9
- TG 2-9
- TH 2-10

media attribute 4-7

media type
- definition 2-4
- sector-reusable 2-4
- volume-reusable 2-4

memos
- for volume sets 4-9
- for volumes 4-5

MESSAGE /LOG command 1-7

MESSAGE command privilege B-2

MF media type and recording types 2-5

MG media type and recording types 2-6

MI media type and recording type 2-6

MIG_FAC_PERIOD system parameter 5-15

MIG_FAC_UNIT system parameter 5-15

MIG_MAX system parameter 5-15

MIG_MIN system parameter 5-15

MIG_REPOP_LOAD system parameter 5-20

MIG_REPOP_MAX system parameter 5-20

MIG_STAGE_LOAD system parameter 5-20

MIGRATE /BY_VSET command 5-16

MIGRATE command 5-14, 5-15

MIGRATE function 5-14

migrating files
- between performance buffer and volume sets 5-14
- between volume sets 5-16

migrating volumes 5-21

migration attributes
- /MINIMUM 5-17
- /ORDER_BY 5-17

migration factor 5-14

MJ media type and recording type 2-6

mount count 5-24

mount limit 5-24

MOVE VOLUME /MEMO command 4-5

MOVE VSET /MEMO command 4-9

N

NA media type and recording types 2-10

NC media type and recording types 2-10

network support interface level 1-5

NETWORK_DEVICE 5-11

NEW file extent status 4-14

NEWLOG command 1-7

NEXT value for VTF attribute 5-3

noncontiguous space for file sets
- extensions 4-12, 4-22
- initial allocation 4-11, 4-22

NONE file extent status 4-14

non-erasable optical media 2-4

normal system recovery 7-3

NOW value for VTF attribute 5-3

O

OA media type and recording types 2-6

OC media type and recording types 2-6

OE media type and recording types 2-7

offline volumes 5-21

OPEN-SEQUENTIAL file function 3-17

OPEN-VRAM file function 3-17

OPERATOR command privilege B-2

optical disk
- drives 2-13
- exchange station 2-14
- library device 2-12
- library device subunits 2-13
- volume formats 4-2

P

PA media type and recording type 2-7

PASSWORD command privilege B-2

passwords
- access group 3-20
- account 6-2
- file 3-21

PERF_BUF_FSET system parameter 4-13

PERF_BUF_VSET system parameter 4-9

performance buffer
- $$BUFFER file set 4-13
- definition 2-12

physical volume 2-3, 4-1

primary file
- directory 3-13
- versions 5-13

priority processing for tape 2-16

privileges
- access 6-2, B-1
- command 6-2, B-2

PUT
- command 4-15
- command privilege B-2
- file transfer function 3-19

R

READ
- file access mode 3-18
- record access function 3-18

read password
- access group 3-20
- file 3-21

READ-KEYED record access function 3-18

read-lock 3-22

READ-NEXT-KEY record access function 3-19

READ-RECORD record access function 3-18

READ-SEQUENTIAL record access function 3-18

RECORD
- command privilege B-2
- file access method 3-17
- file organization 3-6

record
- definition 3-4
- modification table 4-18
- segment 3-4

record access functions
- CHANGE 3-19
- DELETE 3-19
- READ 3-18
- READ-KEYED 3-18
- READ-NEXT-KEY 3-19
- READ-RECORD 3-18
- READ-SEQUENTIAL 3-18
- WRITE 3-19
- WRITE-KEY 3-19

recording type 2-4

RECOVER VOLUME command 5-26

recovering a volume 5-26

recovery
- checkpoint 7-5
- device scan recovery 7-7
- directory recovery from extraction files 7-5
- directory recovery from physical volumes 7-6
- extended 7-4

recovery types
- normal 7-3
- user file 5-12

relative
- file revision 3-3
- file version 3-2
- record number 3-6

RELOCATE command 5-18

REMOVE FILE command 5-13

removing user files 5-13

REPLICATE command 5-12

replicated file 5-10

replicating user files 5-10

replication class 5-11
- assigning 5-11
- default 5-11
- DISABLED_FLAG 5-11
- LINKNAME 5-11
- name attribute 5-11
- NETWORK_DEVICE 5-11
- RPL_CLASS_NAME 5-11
- SYSNAME 5-11
- VSET 5-11

replication class name attribute
- file 5-11
- file set 4-13

residence
- file version 4-14
- volume 4-3
- volume set 4-7

retained file 5-7

retaining user files 5-7

retention
- /RETENTION=DEFAULT 5-8
- /RETENTION=FOREVER 5-8
- /RETENTION=number of days 5-8
- /RETENTION=ZERO 5-8
- determining the value when DEFAULT is specified 5-9
- effect on select commands 5-9

retention attribute
- file 5-7
- file set 4-12

retention period 5-7
- calculating 5-7
- setting 5-8

RETENTION_MODE system parameter 5-9

RETIRE VOLUME command 5-24

retiring volumes 5-24

reusable media
    sector 2-4
    volume 2-4

revision (file) 3-2

revision number 3-2

RPL_CLASS_NAME 5-11

S

SCHEDULE command privilege B-2

sector-reusable media 2-4

SEQUENTIAL
    file access method 3-17
    file organization 3-5, 3-10
    file size 4-15

SERVICE command privilege B-2

session 1-3

SET FILE command 5-3

SET FSET command 5-20

SET FSET/FORCE_RETENTION 4-13

SET GROUP command privilege B-2

SET USER command 1-5

SET VOLUME /CLEANED command 4-5

SET VOLUME /DISABLE command 5-6

SET VOLUME /ENABLE command 5-6

SET VOLUME /MEMO command 4-5

SET VOLUME /UNWRITELOCKED command 4-5

SET VOLUME command 4-4, 4-8

SET VSET /MEMO command 4-9

SET VSET command 4-8

setting retention periods 5-8

shadowed system files 7-1

SHOACCOUNT access privilege B-1

SHOFILE access privilege B-1

SHOGROUP access privilege B-1

show account access 6-2

SHOW command privilege B-2

SHOW FSET command 5-20

SHOW USER command 1-5

SHOW VOLUME /FULL command 4-8

SHOW VOLUME command 4-5

SHOW VSET /FULL command 4-8

SHOW VSET command 4-9

sid (system identifier) 1-6, 3-1

side indicator 2-11

site identifier 1-6, 4-1

SITE_ID system parameter 1-6, 4-1

size
    change table 4-18
    checkpoint table 4-17
    data table 4-17
    estimating for extents 4-16
    file 4-15
    file set 4-11
    files with keys 4-17
    key index table 4-17
    key name table 4-17
    key segment location table 4-17
    keysequential files 4-17
    record modification table 4-18
    segment location table 4-17
    SEQUENTIAL file 4-15
    updated files 4-18
    volume set 4-6
    VRAM data extent 4-16
    VRAM DF extent 4-17
    VRAM file 4-16
    VRAM K extents 4-18
    VRAM update extents 4-19

slot 2-3

software disabled file version 3-20

SQL_MAX_EXT_DATA system parameter 3-12

SQL_MAX_EXT_HASH system parameter 3-12

SQL_MAX_EXT_VALUE system parameter 3-12

SQLADMIN access privilege B-1

SQLCOMMAND command privilege B-2

SQLEXECUTE command privilege B-2

staging files to the performance buffer 5-20

states for volume sides
- CATALOGED 5-18
- ERASING 5-18
- WRITELOCKED 5-18

static system parameters 1-6

statuses for file extents
- BUFFERED 4-14
- LAST 4-14
- NEW 4-14
- NONE 4-14
- UPDATE 4-14
- WRITEBACK_DISABLED 4-14

storage allocation
- attribute 4-11
- blank volumes 4-20
- empty volumes 4-21
- file sets 4-19
- files 4-19
- volume sets 4-19

storage allocation attribute
- contiguous 4-11
- noncontiguous 4-11

storage deallocation
- empty volumes 4-23
- file sets 4-24
- files 4-24
- volume sets 4-23

storage levels
- F (fixed) 2-2
- L (library) 2-2
- N (network) 2-2
- S (shelf) 2-2

storage management functions 5-1

StorHouse
- account system 1-3
- components 1-2
- description xiii
- File System 1-6
- functions 1-2
- interface levels 1-4
- site identifier 1-6
- storage levels 2-2
- system identifier 4-1

STORHOUSE file organization 3-7, 3-11

StorHouse/Admin xiv

StorHouse/Control Center xiv

StorHouse/Performance Monitor 1-7

StorHouse/RM xiii, 3-7

StorHouse/SM xiii

subunit
- definition 2-2
- number 2-3
- optical disk library device 2-13
- tape library device 2-15

subunit types
- A (accessor) 2-3
- D (drive) 2-3
- E (exchange station) 2-3
- S (slot) 2-3

SYSNAME 5-11

SYSTEM command privilege B-3

system files
- backups 7-1
- shadowed 7-1

system identifier (sid) 1-6, 3-1, 4-1

system parameters, general
- deferred dynamic 1-6
- definition 1-6
- dynamic 1-6
- static 1-6

system parameters, specific
- ATF 5-14
- BACKUP 5-2
- CALL_HOME 7-7
- CHKP_ACCOUNT 7-2
- CHKP_FSET 7-2
- CHKP_GROUP 7-2
- CHKP_LIMIT 7-2
- CHKP_ON 7-2
- CHKP_TAKEN 7-2
- CHKP_UPD_NOW 7-2
- CHKP_VSET 7-2
- DEL_FILE_PERM 5-13
- DUPLEX_BALANCE 5-6
- DUPLEX_DIRECTORY 5-5
- DUPLEX_ENABLE 5-5, 5-6
- FREE_POOL_didmmr 4-20
- LIMIT 5-13
- MIG_FAC_PERIOD 5-15
- MIG_FAC_UNIT 5-15
- MIG_MAX 5-15
- MIG_MIN 5-15
- MIG_REPOP_LOAD 5-20
- MIG_REPOP_MAX 5-20
- MIG_STAGE_LOAD 5-20
- PERF_BUF_FSET 4-13
- PERF_BUF_VSET 4-9
- RETENTION_MODE 5-9
- SITE_ID 1-6, 4-1
- SQL_MAX_EXT_DATA 3-12
- SQL_MAX_EXT_HASH 3-12
- SQL_MAX_EXT_VALUE 3-12
- SYSTEM_ID 1-6, 4-1
- TMD_DISMNT_DELAY 2-18
- TMD_HOLD_HIGH 2-18
- TMD_HOLD_LOW 2-18
- VRAM_UPDATE 3-11, 4-18, 4-19
- VSET_HOLD 4-8
- VTF 5-3

system recovery types
- checkpoint 7-5
- device scan recovery 7-7
- extended 7-4
- normal 7-3

SYSTEM_ID system parameter 1-6, 4-1

T

tables
- change 4-18
- checkpoint 4-17
- data 4-17
- key index 4-17
- key name 4-17
- record modification 4-18

tape, magnetic 2-16

TB media type and recording types 2-8

TC media type and recording types 2-8

TD media type and recording types 2-8

TE media type and recording types 2-9

TF media type and recording types 2-9

TG media type and recording types 2-9

TH media type and recording types 2-10

TMD_DISMNT_DELAY system parameter 2-18

TMD_HOLD_HIGH system parameter 2-18

TMD_HOLD_LOW system parameter 2-18

transport arm 2-3

transportable data records
- ASCII 3-5
- binary 3-5

truncated file 3-3

U

U (update) extent 3-9, 3-11

uid (user identification code) 1-3

UNCATALOG VOLUME command 5-19, 5-22

UNCATALOG VSET command 5-22

UNCATALOGED volume side state 5-18

uncataloging
- volume 5-22
- volume set 5-22

UNCATALOGING volume side state 5-18

unit number 2-2

update attribute for file sets 4-12

UPDATE file access mode 3-18

UPDATE file extent status 4-14

update space for file sets 4-12

updated files 4-18

upward migration 5-20

usage factor 5-15

user files
- archiving 5-4
- backing up 5-3
- migrating between volume sets 5-16
- migrating from performance buffer 5-14
- recovering 5-12
- removing 5-13
- replicating 5-10
- retaining 5-7
- staging 5-20

user identification code (uid) 1-3

user log 1-7

V

version directory information 3-14

version, file 3-2

vid (volume identification code) 2-4

volume
- access 2-11
- blank 4-20
- cycle time 4-4
- empty 4-20
- erasing 5-23
- exporting 5-22
- HOLD attribute 4-3
- importing 5-22
- label 2-10
- label record 4-1
- logical 4-1
- magnetic disk format 4-2
- magnetic tape format 4-2
- media type 2-4
- memos 4-5
- migration 5-21
- optical disk format 4-2
- physical 4-1
- recording type 2-4
- recovery 5-26
- residence 4-3
- retirement 5-24
- uncataloging 5-22

volume classes
- cataloged 5-21
- offline 5-21
- uncataloged 5-21

volume identification code (vid) 2-4

volume set
- attributes 4-6
- cycle time 4-8
- deactivation time 4-8
- definition 4-6
- erasing 5-23
- expiration time 4-8
- exporting 5-22
- file set placement 4-21
- free pool 4-20
- HOLD attribute 4-7
- level F 4-9
- library device attribute 4-7
- LIMIT attribute 4-6
- MAGDISK 4-9
- memos 4-9
- name 4-6
- residence 4-7
- size 4-6
- storage deallocation 4-23
- uncataloging 5-22

volume side
- deactivation time 4-3

expiration time 4-4
writelocked 4-4

volume table of contents (VTOC) 4-2

volume-reusable media 2-4

VRAM
data extent size 4-16
DF extent size 4-17
file organization 3-10
file size 4-16
K extent size 4-18
update extent size 4-19

VRAM_UPDATE system parameter 3-11, 4-18, 4-19

VSET 5-11

VSET_HOLD system parameter 4-8

VTF
attribute 5-3
command privilege B-3
system parameter 5-3

VTOC (volume table of contents) 4-2

W

WORM (write-once, read-many) optical media 2-4

WRITE
file access mode 3-18
record access function 3-19

write password
access group 3-20
file 3-21

write-back for file extents 4-23

WRITEBACK_DISABLED file extent status 4-14

WRITE-KEY record access function 3-19

write-lock 3-22

WRITELOCKED volume side state 5-18

writelocked volume sides 4-4

X

XFER file transfer function 3-19