



## Utilities Reference Manual

Publication Number  
900141 Rev. C  
April 25, 2002

---

The FileTek logo consists of a solid teal square. To its left, the word "FileTek" is written in a white, bold, sans-serif font. The entire logo is positioned on the right side of a horizontal dotted line that spans the width of the page.

FileTek



All rights reserved. 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.

Copyright © 1999-2000 by FileTek, Inc., Rockville, MD  
Publication Number: 900141 Rev. C.

**NOTE: U.S. GOVERNMENT USERS**

**Restricted Rights Legend**

Use, duplication or disclosure by the Government is subject to the restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or the Commercial Computer Software - Restricted Rights clause at 48 CFR 52.227-19, as applicable. Unpublished-rights reserved under the copyright laws of the United States. 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. Other patents pending.

# Contents

<b>Welcome .....</b>	<b>xv</b>
Purpose of this document .....	xvi
Intended audience .....	xvi
Scope of this document .....	xvi
Contents .....	xvii
Related documentation .....	xxii
StorHouse user document set .....	xxii
StorHouse/RM user document set .....	xxiv
Control Center user document set .....	xxvi
Customer support document set (work-in-process) .....	xxvii
Conventions .....	xxviii
 <b>Chapter 1: Introduction .....</b>	 <b>1-1</b>
What are internal StorHouse utilities? .....	1-1
What utilities are available? .....	1-2
How do I use the StorHouse utilities? .....	1-6
Understanding format conventions .....	1-6
Executing StorHouse utilities .....	1-7
From the StorHouse operating system (UNIX) prompt .....	1-7
From the StorHouse command prompt (?) .....	1-10
Redirecting the output .....	1-11

Getting online help .....	1-11
From the StorHouse operating system (UNIX) prompt .....	1-12
From the StorHouse command prompt (?) .....	1-12

## **Chapter 2: add\_accounts utility .....2-1**

How add_accounts works .....	2-1
Properties of StorHouse accounts .....	2-2
Format .....	2-3

## **Chapter 3: asd utility .....3-1**

Overview of the asd process .....	3-1
Including a setup script in a fixes file .....	3-2
Identifying replacement software to be installed .....	3-3
Adding a JUSTDOIT file .....	3-3
Format .....	3-4
Guidelines for creating a fixes file .....	3-5
Error handling .....	3-6
Examples .....	3-6
Output .....	3-8
Interpreting log data .....	3-8
Sample log file .....	3-10

## **Chapter 4: AUTOMV utility .....4-1**

Overview of the AUTOMV process .....	4-1
Including a shell script in a fixes file .....	4-2
Format .....	4-2

Guidelines for creating a fixes file .....	4-3
<b>Chapter 5: balance utility .....</b>	<b>5-1</b>
Balancing the load of volume mounts .....	5-1
Rules used by the load balancing process .....	5-2
Format .....	5-2
Examples .....	5-4
Sample report .....	5-5
Interpreting the report data .....	5-7
<b>Chapter 6: bmdi utility .....</b>	<b>6-1</b>
Format .....	6-1
Sample output .....	6-2
<b>Chapter 7: build_sxm utility .....</b>	<b>7-1</b>
build_sxm tasks .....	7-1
Before you use build_sxm .....	7-3
Format .....	7-3
<b>Chapter 8: cfg_modem utility .....</b>	<b>8-1</b>
Input to the cfg_modem utility .....	8-2
Implementing modem security .....	8-2
Enabling or disabling remote access .....	8-3
Format .....	8-4
Examples .....	8-5

<b>Chapter 9: cfg_net utility .....</b>	<b>9-1</b>
Input to the cfg_net utility .....	9-2
Format .....	9-3
Examples .....	9-4
 <b>Chapter 10: cleanipcs utility .....</b>	 <b>10-1</b>
How to use cleanipcs .....	10-1
Format .....	10-2
 <b>Chapter 11: cpshow utility .....</b>	 <b>11-1</b>
Format .....	11-1
Sample output .....	11-2
 <b>Chapter 12: cpstop utility .....</b>	 <b>12-1</b>
Format .....	12-1
Examples .....	12-2
 <b>Chapter 13: ctest utility .....</b>	 <b>13-1</b>
How ctest works .....	13-1
Primary steps performed by ctest .....	13-2
Files created by ctest .....	13-3
Where ctest writes data .....	13-4
Before you use ctest .....	13-4
Format .....	13-6
Possible error conditions .....	13-8

Volume isn't in selected library .....	13-8
Unable to create the test file .....	13-9
Write and read results aren't identical .....	13-9
Sample output .....	13-10
<b>Chapter 14: ctestsu utility .....</b>	<b>14-1</b>
How ctestsu works .....	14-1
Files created by ctestsu .....	14-1
Where ctestsu writes data .....	14-2
Format .....	14-3
Sample output .....	14-3
<b>Chapter 15: cxclean utility .....</b>	<b>15-1</b>
Format .....	15-1
<b>Chapter 16: dbdown utility .....</b>	<b>16-1</b>
Format .....	16-1
Examples .....	16-2
Sample output .....	16-2
<b>Chapter 17: dbup utility .....</b>	<b>17-1</b>
Forcing a database online .....	17-1
Format .....	17-2
Examples .....	17-2
Sample output .....	17-3

<b>Chapter 18: dc_maint utility .....</b>	<b>18-1</b>
Format .....	18-1
Examples .....	18-5
Sample output .....	18-6
Interpreting the output .....	18-7
 <b>Chapter 19: doit utility .....</b>	 <b>19-1</b>
Rules for command parsing .....	19-2
Format .....	19-2
Sample doit procedure for VRAM files .....	19-13
Sample doit procedure for non-VRAM files .....	19-14
 <b>Chapter 20: doit.basic utility .....</b>	 <b>20-1</b>
Format .....	20-2
Sample output .....	20-3
 <b>Chapter 21: dumpulog utility .....</b>	 <b>21-1</b>
Format .....	21-2
Examples .....	21-4
Sample output .....	21-5
Possible error conditions and resolutions .....	21-7



<b>Chapter 22: follow utility .....</b>	<b>22-1</b>
How follow works .....	22-1
Format .....	22-2
Examples .....	22-3
Sample output .....	22-3
 <b>Chapter 23: gen_dev utility .....</b>	 <b>23-1</b>
Inputs to gen_dev .....	23-2
config_dev .....	23-2
spx_maint.inp .....	23-2
Modifying the template files .....	23-3
Adding entries to spx_maint.inp .....	23-3
Installing new systems – build_sxm and gen_dev .....	23-4
build_sxm tasks .....	23-5
gen_dev tasks .....	23-5
Updating an existing system with gen_dev .....	23-9
Preparing to execute gen_dev .....	23-10
gen_dev tasks .....	23-10
Format .....	23-11
Examples .....	23-12
 <b>Chapter 24: genqmm utility .....</b>	 <b>24-1</b>
Files generated by genqmm .....	24-2
Format .....	24-2

<b>Chapter 25: gzip utility .....</b>	<b>25-1</b>
Format .....	25-2
Sample output .....	25-2
 <b>Chapter 26: maint utility .....</b>	 <b>26-1</b>
Format .....	26-2
Sample output .....	26-5
 <b>Chapter 27: make_bkups utility .....</b>	 <b>27-1</b>
Before you use make_bkups .....	27-2
Format .....	27-2
Examples .....	27-3
 <b>Chapter 28: meta_rstr utility .....</b>	 <b>28-1</b>
How meta_rstr works .....	28-1
Before you use meta_rstr .....	28-3
Format .....	28-4
Correcting severe error conditions .....	28-4
Format of error messages .....	28-4
Problem determination checklist .....	28-5
List of error messages .....	28-8
 <b>Chapter 29: movevol utility .....</b>	 <b>29-1</b>
How movevol works .....	29-1
Balancing the load of volumes to be moved .....	29-1

Reserving one or more library drives .....	29-2
Monitoring the movement of volumes .....	29-2
Format .....	29-3
Examples .....	29-5
 <b>Chapter 30: port utility .....</b>	<b>30-1</b>
Format .....	30-2
Examples .....	30-3
 <b>Chapter 31: qchsend utility .....</b>	<b>31-1</b>
Format .....	31-2
Examples .....	31-2
Sample Call Home report .....	31-3
 <b>Chapter 32: rd utility .....</b>	<b>32-1</b>
Format .....	32-2
Keystroke commands .....	32-2
Main menu commands .....	32-4
RL and RQ Queues display commands .....	32-7
Line editing commands .....	32-9
Refreshing the display .....	32-9
Sample output .....	32-10

<b>Chapter 33: rebuild_disk utility .....</b>	<b>33-1</b>
Before you use rebuild_disk .....	33-1
How to execute rebuild_disk .....	33-2
Format .....	33-2
 <b>Chapter 34: recds utility .....</b>	 <b>34-1</b>
Format .....	34-2
Examples .....	34-4
Sample output .....	34-5
Possible error conditions and resolutions .....	34-7
 <b>Chapter 35: set_timezone utility .....</b>	 <b>35-1</b>
Selecting a time zone .....	35-1
Format .....	35-3
Examples .....	35-3
Sample output .....	35-4
 <b>Chapter 36: shelf utility .....</b>	 <b>36-1</b>
Format .....	36-1
Sample report .....	36-3
 <b>Chapter 37: startsm utility .....</b>	 <b>37-1</b>
Format .....	37-3

<b>Chapter 38: sthtest utility .....</b>	<b>38-1</b>
How the sth_val_prep procedure works .....	38-1
Input to sth_val_prep .....	38-2
Steps performed by sth_val_prep .....	38-2
Before you use sth_val_prep .....	38-3
Format for sth_val_prep .....	38-4
Files created by sth_val_prep .....	38-6
Sample output from sth_val_prep .....	38-7
How the sth_val_del procedure works .....	38-8
Steps performed by sth_val_del .....	38-8
Format for sth_val_del .....	38-9
File created by sth_val_del .....	38-9
Sample output from sth_val_del .....	38-9
 <b>Chapter 39: stopsm utility .....</b>	 <b>39-1</b>
Format .....	39-1
 <b>Chapter 40: sxm utility .....</b>	 <b>40-1</b>
How to use sxm .....	40-1
Executing sxm indirectly .....	40-1
Executing sxm directly .....	40-2
Format .....	40-2
 <b>Chapter 41: syscreate utility .....</b>	 <b>41-1</b>
Format .....	41-2

<b>Chapter 42: timeup utility .....</b>	<b>42-1</b>
Format .....	42-1
Sample output .....	42-2
 <b>Appendix A: List of terms .....</b>	 <b>A-1</b>
 <b>Appendix B: StorHouse directory structure .....</b>	 <b>B-1</b>
Environment variables .....	B-1
StorHouse/SM directory structure .....	B-5
StorHouse/RM directory structure .....	B-10
 <b>Index .....</b>	 <b>Index-1</b>

# Welcome

*StorHouse®* is FileTek's enterprise-wide solution for managing the capture, storage, movement, and access of gigabytes to petabytes of relational and non-relational atomic-level data. StorHouse technology combines industry-leading, scalable storage devices and Open System processors with FileTek's specialized hierarchical storage management (HSM) and relational database management system (RDBMS) software components. Together, these components make StorHouse the ideal hub server, or central storage repository, for enterprise-wide transaction-level data.

*StorHouse/SM*, FileTek's standard HSM component, controls a hierarchy of storage devices comprised of cache, redundant array of independent disk (RAID), erasable and write-once-read-many (WORM) optical disk jukeboxes, and automated tape libraries. StorHouse/SM is also responsible for automating critical system management tasks, like data migration, backup, and recovery. StorHouse/SM comes standard with all StorHouse systems.

*StorHouse/RM*, FileTek's RDBMS component, works in conjunction with StorHouse/SM to specifically administer the storage, access, and movement of relational data. StorHouse/RM provides row-level SQL access to atomic data on any layer—including tape—in the StorHouse storage hierarchy. SQL access is available from different platforms through a variety of industry-standard protocols. StorHouse/RM is the ideal way to feed target data marts the specific atomic data they need for analysis and decision making.

## Purpose of this document

The *Utilities Reference Manual* serves as a reference for internal StorHouse utilities; that is, utilities reserved for use by FileTek® customer support and other authorized personnel rather than StorHouse end users. This manual contains command formats and examples and provides guidelines for using StorHouse utilities on a StorHouse 5.x (Solaris 2-based) system and, optionally, a StorHouse/RM 2.x system. It is designed to help you provide StorHouse customers with responsive, efficient, and economical service.

## Intended audience

The *Utilities Reference Manual* is intended for FileTek customer support representatives and other support personnel who are responsible for installing, maintaining, and troubleshooting StorHouse systems. This manual assumes that you are familiar with the UNIX® operating system and that you already understand StorHouse concepts. In addition, you should be familiar with the support terminology described in Appendix A of this manual as well as the StorHouse directory structure described in Appendix B.

## Scope of this document

The *Utilities Reference Manual* is a work-in-progress that will be developed in four phases. The current version of the document, which represents the culmination of efforts for phases 1 and 2, describes the 41 StorHouse utilities deemed most important by customer support.



Some of the utilities described in this version of the manual refer to utilities that aren't scheduled to be added to the *Utilities Reference Manual* until a subsequent phase. Utilities that are referenced but not yet described in this manual include:

- bmon\_ctcs
- bmon\_nvm
- diskfree
- jgen
- qm\_maint
- rxt\_show
- spx\_maint

These utilities will be added to the *Utilities Reference Manual* in phases 3 and 4. However, a brief description of each utility appears in Appendix A, “List of terms,” in this version of the manual to give you a general idea of each utility’s purpose.

## Contents

This document contains the following chapters and appendixes (utilities are organized in alphabetical order):

- Chapter 1, “Introduction,” lists and briefly describes the internal StorHouse utilities. In addition, it describes the format conventions you must follow when using StorHouse utilities, the various ways you can execute the utilities, and how you can obtain online help. You should read this chapter in its entirety.
- Chapter 2, “add\_accounts utility,” describes the purpose and format of the add\_accounts utility. In addition, it provides an overview of the standard StorHouse accounts created by the utility, including a list of the privileges granted to each account.

- Chapter 3, “asd utility,” describes the purpose of the asd utility and provides an overview of the asd process, including the tasks you must perform when using this utility. It also describes the format of the asd command, lists the guidelines you need to follow when using this utility, and presents a sample log file.
- Chapter 4, “AUTOMV utility,” describes the purpose and format of the AUTOMV utility and provides an overview of the AUTOMV process. It also lists the guidelines you need to follow when using this utility.
- Chapter 5, “balance utility,” describes the purpose and format of the balance utility. In addition, the chapter describes the rules used by the load balancing process, provides examples of using the utility, and presents a sample report.
- Chapter 6, “bmdi utility,” describes the purpose and format of the bmdi utility and presents sample output.
- Chapter 7, “build\_sxm utility,” describes the purpose and format of the build\_sxm utility, as well as the tasks you must perform before you execute build\_sxm. In addition, it lists and describes each major task that build\_sxm performs.
- Chapter 8, “cfg\_modem utility,” describes the purpose and format of the cfg\_modem utility, as well as the input to the modem configuration process. It also describes how to implement modem security and how to enable or disable remote access to a StorHouse system via the telephone lines. Finally, the chapter provides examples of using the utility.
- Chapter 9, “cfg\_net utility,” describes the purpose and format of the cfg\_net utility, as well as the input to the network configuration process. It also provides examples of using the utility.
- Chapter 10, “cleanipcs utility,” describes the purpose and format of the cleanipcs utility. It also describes how to use cleanipcs and recover from errors.

- Chapter 11, “cpshow utility,” describes the purpose and format of the cpshow utility and presents sample output.
- Chapter 12, “cpstop utility,” describes the purpose and format of the cpstop utility. It also provides examples of using the utility.
- Chapter 13, “ctest utility,” describes the purpose and format of the ctest utility. In addition, it describes how ctest works and what you should do before you use ctest. Finally, the chapter describes possible error conditions and presents sample output.
- Chapter 14, “ctestsu utility,” describes the purpose and format of the ctestsu utility. In addition, it describes how ctestsu works and presents sample output.
- Chapter 15, “cxclean utility,” describes the purpose and format of the cxclean utility.
- Chapter 16, “dbdown utility,” describes the purpose and format of the dbdown utility and presents sample output.
- Chapter 17, “dbup utility,” describes the purpose and format of the dbup utility. In addition, it describes how to force a database online, provides examples of using the utility, and presents sample output.
- Chapter 18, “dc\_maint utility,” describes the purpose and format of the dc\_maint utility. It also provides examples of using the utility and presents sample output.
- Chapter 19, “doit utility,” describes the purpose and format of the doit utility. In addition, it lists the rules you must adhere to when using this utility and provides sample doit procedures.
- Chapter 20, “doit.basic utility,” describes the purpose and format of the doit.basic utility and presents sample output.

- Chapter 21, “dumpulog utility,” describes the purpose and format of the dumpulog utility. In addition, it presents sample output and describes possible error conditions and resolutions.
- Chapter 22, “follow utility,” describes the purpose and format of the follow utility. It also provides examples of using the utility.
- Chapter 23, “gen\_dev utility,” describes the purpose and format of the gen\_dev utility. It lists and describes the inputs to the gen\_dev process and describes both the high-level tasks and the related sub-tasks that gen\_dev performs for both a new and an existing StorHouse system. In addition, the chapter provides examples of using the gen\_dev command.
- Chapter 24, “genqmm utility,” describes the purpose and format of the genqmm utility. In addition, it describes the StorHouse system files generated by genqmm.
- Chapter 25, “gzip utility,” describes the purpose and format of the gzip utility and presents sample output.
- Chapter 26, “maint utility,” describes the purpose and format of the maint utility and presents sample output.
- Chapter 27, “make\_bkups utility,” describes the purpose and format of the make\_bkups utility. In addition, it describes steps you should perform before you use make\_bkups and how make\_bkups works.
- Chapter 28, “meta\_rstr utility,” describes the purpose and format of the meta\_rstr utility. In addition, it describes how meta\_rstr works and what you should do before you use meta\_rstr. Finally, the chapter lists and describes fatal error messages that might result during execution, and provides possible resolutions for error conditions.
- Chapter 29, “movevol utility,” describes the purpose and format of the movevol utility. In addition, it describes how movevol works and provides examples of using the utility.

- Chapter 30, “port utility,” describes the purpose and format of the port utility. It also provides examples of using the utility.
- Chapter 31, “qchsend utility,” describes the purpose and format of the qchsend utility. It also provides examples of using the utility and presents a sample Call Home report.
- Chapter 32, “rd utility,” describes the purpose and format of the rd utility. It displays a sample screen shot of the rd main menu and describes each keystroke command you can use with the utility. The chapter also presents sample output.
- Chapter 33, “rebuild\_disk utility,” describes the purpose and format of the rebuild\_disk utility. In addition, it describes steps you should perform before you use rebuild\_disk and how to execute the utility.
- Chapter 34, “recds utility,” describes the purpose and format of the recds utility. In addition, it provides examples of using the utility, presents sample output, and describes possible error conditions and resolutions.
- Chapter 35, “set\_timezone utility,” describes the purpose and format of the set\_timezone utility. In addition, it describes how to use the utility and how to select a time zone. Finally, the chapter provides examples of using the utility and presents sample output.
- Chapter 36, “shelf utility,” describes the purpose and format of the shelf utility and presents a sample report.
- Chapter 37, “startsm utility,” describes the purpose and format of the startsm utility.
- Chapter 38, “sthtest utility,” describes the purpose of the sthtest utility and the format of the two UNIX scripts, or *procedures*, that comprise the utility. In addition, it describes how the procedures work, including the inputs and outputs. Finally, the chapter provides sample output from each procedure.

- Chapter 39, “stopsm utility,” describes the purpose and format of the stopsm utility.
- Chapter 40, “sxm utility,” describes the purpose and format of the sxm utility. In addition, it describes how to use sxm and provides examples of using the utility.
- Chapter 41, “syscreate utility,” describes the purpose and format of the syscreate utility. In addition, it lists the rules for naming a StorHouse/RM database and presents sample output.
- Chapter 42, “timeup utility,” describes the purpose and format of the timeup utility and presents sample output.
- Appendix A, “List of terms,” defines terms used by the StorHouse support organization as well as other terms used in this book.
- Appendix B, “StorHouse directory structure,” defines the directories, files, and environment variables that make up a StorHouse system.

## Related documentation

It may be helpful to be familiar with the documentation listed in the following sections.

### StorHouse user document set

- The *StorHouse Glossary*, publication number 900027, defines the terminology used in FileTek StorHouse publications.
- The *StorHouse Concepts and Facilities Manual*, publication number 900026, defines the basic concepts, structures, and functions of StorHouse.

- The *Command Language Reference Manual*, publication number 900005, contains descriptions of StorHouse commands and is intended for all users. You should be familiar with the information presented in this manual before reading the *System Administrator's Guide*.
- The *System Administrator's Guide*, publication number 900007, describes system recovery, account administration, and storage management procedures and concepts for StorHouse.
- The *System Operator's Guide*, publication number 900008, contains basic operating instructions for StorHouse hardware and software.
- The *StorHouse System Administrator's Quick Reference*, publication number C00006, presents the syntax of StorHouse Command Language commands that system administrators frequently use.
- The *StorHouse System Operator's Quick Reference*, publication number C00003, presents the procedures for logging in and out of the StorHouse operating system, manually starting the StorHouse software, and signing on and off StorHouse.
- The StorHouse *Messages and Codes Manual*, publication number 900032, lists all StorHouse system and host software messages.
- The *User Log Format*, publication number 900028, defines the record format for the user log file. Programmers can access this file for report generation purposes.
- 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 *Host Installation and Operations Guide*, publication number 900011 for IBM MVS hosts, 900002 for DEC™ VAX™/VMS™ hosts, 900051 for UNIX hosts, and 900052 for DOS hosts, explains how to install the StorHouse host software.

## **StorHouse/RM user document set**

- The *StorHouse/RM Glossary*, publication number 900112, defines the terminology used in the StorHouse/RM User Document Set.
- The *StorHouse/RM Concepts* manual, publication number 900132, describes the key concepts of StorHouse/RM. It explains the structures that store relational data on StorHouse and the facilities that access and manage that data.
- The *StorHouse Database Administration Guide*, publication number 900108, describes StorHouse database concepts and explains how to create user tables and indexes, manage accounts and privileges, set up user tablespaces, and perform other StorHouse system and database administration tasks.
- The *StorHouse SQL Reference Manual*, publication number 900111, is the comprehensive reference for StorHouse SQL. It defines StorHouse SQL statements, functions, predicates, and data types. In addition, it contains format descriptions, examples, and a list of all StorHouse status codes.
- The *StorHouse SQL Quick Reference*, publication number 900122, is a standalone mini-manual that contains formats and examples of all StorHouse SQL statements, functions, and predicates.
- The *StorHouse ESQl Manual*, publication number 900121, explains how to use StorHouse SQL in application programs.
- The *FileTek MVS Data Loader Utility Manual*, publication number 900109, describes how to load data into StorHouse user tables from an MVS environment.



- The *FileTek FTP Data Loader Manual*, publication number 900115, explains how to load data into StorHouse user tables from a UNIX, VAX, or other FTP-enabled host environment.
- The *LOAD Quick Reference*, publication number C00005, contains the format and an example of the LOAD (or LOAD INTO TABLE) statement used to load data into StorHouse user tables.
- The *FileTek FTP Data Loader Commands and Keywords Quick Reference*, publication number C00008, contains the File Transfer Protocol (FTP) commands for loading data into StorHouse user tables. It also describes the customized keywords for the FTP put command.
- The *FileTek FTP Data Unloader Manual*, publication number 900137, explains how to unload data from StorHouse databases using FTP.
- The *UNLOAD Quick Reference*, publication number C00009, contains the format and examples of the UNLOAD statement used to unload data from StorHouse user tables.
- The *FileTek FTP Data Unloader Commands and Keywords Quick Reference*, publication number C00010, contains the FTP commands for unloading StorHouse user table data. It also describes the customized keywords for the FTP put and get commands.
- The *StorHouse/RM Metadata Conversion Manual*, publication number 900142, explains how to convert metadata from previous StorHouse/RM 2.x releases to the current release.

## Control Center user document set

- The *Control Center Glossary*, publication number 900140, defines terms and acronyms used throughout the Control Center user document set.
- *Getting Started with Control Center*, publication number 900138, explains how to install and configure the basic Control Center software and each of the Control Center client modules.
- *Getting Started with StorHouse/Admin*, publication number 900135, explains how to perform StorHouse system and database administration tasks.
- *Getting Started with StorHouse/Performance Monitor*, publication number 900134, explains how to set up StorHouse/Performance Monitor, and how to create and work with reports.
- The *StorHouse/Admin System Administrator's Quick Reference*, publication number 900147, provides quick reference procedures for performing StorHouse system administration tasks with StorHouse/Admin.
- The *StorHouse/Admin System Operator's Quick Reference*, publication number 900149, provides quick reference procedures for performing StorHouse system operator tasks with StorHouse/Admin.
- The *StorHouse/Admin Database Administrator's Quick Reference*, publication number 900150, provides quick reference procedures for performing StorHouse database administration tasks with StorHouse/Admin.

## Customer support document set (work-in-process)

The following manuals are under development by the FileTek Customer Support organization and have not yet been published:

- The *StorHouse Software Support Concepts* manual, publication number 900143, provides an overview of the StorHouse software, with an emphasis on concepts related to system installation and support.
- The *StorHouse Supported Devices and Related Software* manual, publication number 900145, lists and describes all supported hardware devices, their related software and firmware, and their StorHouse configuration information.
- The *StorHouse Installation and Upgrade Guide*, publication number 900144, provides step-by-step instructions for installing and upgrading StorHouse and StorHouse/RM software.
- The *StorHouse System Monitoring and Problem Determination Guide*, publication number 900146, describes troubleshooting procedures you can use to determine whether an error in a StorHouse system is the result of a hardware malfunction or a software problem.

## Conventions

This manual uses the following notational conventions:

Convention	Meaning
Helvetica font	StorHouse utility command formats and examples
<i>Italics</i>	New terms, emphasized text, and publication titles
<code>Courier font</code>	Code

See "Understanding format conventions" on page 1-6 for specific StorHouse utility conventions.

# Introduction

This chapter provides an overview of FileTek's internal StorHouse utilities. It answers the following questions:

- What are internal StorHouse utilities?
- What utilities are available?
- How do I use the StorHouse utilities?
- How can I get help for StorHouse utilities?

## What are internal StorHouse utilities?

FileTek's StorHouse utilities are internally-developed UNIX scripts or programs that enable FileTek customer support representatives and other support personnel to perform a variety of tasks. Among those tasks are installing and maintaining StorHouse systems, and diagnosing and correcting performance-related issues and other problems.

Authorized personnel generally run the StorHouse utilities from the StorHouse operating system (UNIX) command prompt; however, a limited set of utilities can also be executed using the StorHouse Command Language RUN command from the StorHouse Interactive Interface command prompt (?). The various ways of executing the utilities are described later in this chapter.

## What utilities are available?

The following table lists the internal StorHouse utilities that are currently available.

**Table 1-1: Internal StorHouse utilities**

Use...	To...	During...
add_accounts	Create a standard set of StorHouse accounts for a newly installed StorHouse system.	Installation
asd	Install software enhancements and fixes on existing FileTek StorHouse systems.	Installation/maintenance
AUTOMV	Install updates or software fixes for the FileTek StorHouse system and execute special files at each customer site on a daily basis.	Installation/maintenance
balance	Monitor volume mount activity for each optical library in a StorHouse system and balance the load of volume mounts, if necessary.	Operation
bmdi	Determine if all level F disks are operational and produce either a report for each disk or an error message if a disk is inaccessible or is not initialized correctly. Use also to initialize level F disks after installation.	Installation/maintenance
build_sxm	Configure a new StorHouse system.	Installation
cfg_modem	Configure a modem for a StorHouse system.	Installation
cfg_net	Configure network-related aspects of a StorHouse system.	Installation

**Table 1-1: Internal StorHouse utilities (continued)**

Use...	To...	During...
cleanipcs	Clean up shared system resources (shared memory, semaphores, and message queues) for StorHouse/RM.	Maintenance
cpshow	Query the status of one or all active StorHouse processes.	Operation/maintenance
cpstop	Stop one or more active StorHouse processes.	Maintenance
ctest	Ensure that storage devices in a StorHouse system can write and read data within acceptable thresholds.	Testing
ctestsu	Create read-only test files that can be read by the ctest utility.	Testing
cxclean	Clean up Common Support (C) facility shared system resources.	Maintenance
dbdown	Take a StorHouse database offline manually and force it into a “down” state.	Maintenance
dbup	Reset a downed StorHouse database to an “up” (online) state.	Maintenance
dc_maint	Check the status of the channels; that is, StorHouse’s connectivity to the host (mainframe).	Operation/installation/ maintenance
doit	Test StorHouse API functions or Command Language commands.	Testing
doit.basic	Perform a basic Virtual Record Access Manager (VRAM™) test after performing maintenance procedures on a StorHouse system.	Testing

**Table 1-1: Internal StorHouse utilities (continued)**

Use...	To...	During...
dumplog	Produce a formatted dump for any number of, or all, user log record types for one or more days.	Operation/maintenance
follow	Monitor the progress of various files on a customer's StorHouse system.	Testing/maintenance
gen_dev	Generate or update device configuration and system files for a new or existing StorHouse system.	Installation/maintenance
genqmm	Generate a set of StorHouse system files that describe default values for all possible devices and media/recording types in a StorHouse system.	Installation/maintenance
gzip	Compress the size of a UNIX file containing software fixes.	Installation/maintenance
maint	Prepare a StorHouse system for maintenance and perform selected maintenance activities.	Installation/operation/ maintenance
make_bkups	Back up the system disk (boot device) on a StorHouse server.	Installation/maintenance
meta_rstr	Recover metadata for one or more StorHouse databases from backup files.	Maintenance
movevol	Test an optical library device to ensure that the library and its drives are working properly.	Testing
port	Configure ports for a StorHouse system.	Installation
qchsend	Issue a Call Home report as well as test the Call Home error reporting software for StorHouse.	Testing



**Table 1-1: Internal StorHouse utilities (continued)**

Use...	To...	During...
rd	Debug and test the StorHouse software by analyzing a real-time display of system activities.	Maintenance
rebuild_disk	Rebuild partitions that you previously backed up on a replacement system disk.	Maintenance
recds	Produce a short form dump for a subset of user log record types for one or more days.	Operation/maintenance
set_timezone	Designate the time zone in which a StorHouse system will be operating.	Installation
shelf	Monitor I/O station activity for each optical library in a StorHouse system for a specified day or date range.	Operation
startsm	Start the StorHouse software.	Operation
sthtest	Verify that you successfully installed a StorHouse/RM system.	Testing
stopsm	Shut down the StorHouse software.	Operation
sxm	Invoke a StorHouse utility for which specific setup tasks must be performed prior to execution.	Maintenance
syscreate	Create a StorHouse database.	Installation/operation
timeup	Determine whether the StorHouse software is running.	Operation

## How do I use the StorHouse utilities?

You can execute internal StorHouse utilities in a number of ways. However, before you can use StorHouse utilities, you must understand the format conventions for utility commands.

### Understanding format conventions

UNIX differentiates between capital and small letters. Therefore, it is important that you type utility commands exactly as they appear in this book. Most commands must appear in lowercase; however, if a command appears in uppercase in this book, you must type the command using capital letters.

The following table can help you interpret the symbols and other conventions used in StorHouse utility formats.

Convention	Description
<i>Italics</i>	Indicates user-specified values or a user-specified identifier that has specific values associated with it.
" "	Delimits arguments whose values contain special characters (that is, characters that have special meaning to UNIX such as parentheses). These values must be enclosed in quotation marks (either double or single).
( ) , : "	Indicates required characters (parentheses, commas, colons, and quotation marks) that must be entered as part of the syntax.
{ }	Indicates that the item shown between the braces is required. When a list of items is enclosed in braces and separated by a vertical bar, you must choose one item in the list.
[ ]	Indicates that the item shown between the square brackets is optional, such as an optional parameter for a command.
	Separates alternatives. You can specify any one of the alternatives shown.

For example, the following command format uses the conventions defined in the preceding table:

recds "{today | yesterday | first=*yyyyjjj*},[last=*yyyyjjj*],[select=(*n:n:n*)"

In this example:

- today, yesterday, and first are enclosed in braces ({ }) and separated by a vertical bar (|) because one of them is required to complete the command. The braces indicate a requirement, while the vertical bar indicates a choice of items within the braces.
- last=*yyyyjjj* and select=(*n:n:n*) are enclosed in square brackets ([ ]) because these parameters are not required to complete this command. They are optional.
- *yyyyjjj* and *n:n:n* are shown in italics because they are user-specified values.
- The arguments and their values are enclosed in quotation marks because one of the values contains special characters (parentheses).

## Executing StorHouse utilities

The primary means of executing internal StorHouse utilities is from the StorHouse operating system (UNIX) command prompt. However, it is possible to execute some of the StorHouse utilities using the Command Language RUN command from the StorHouse command prompt (?). The following sections describe these methods of executing StorHouse utilities and illustrate the general format of the commands. Refer to the chapter that describes the utility you want to use for specific command syntax.

### From the StorHouse operating system (UNIX) prompt

In order to execute StorHouse utilities from the StorHouse operating system prompt, you must first log in to the StorHouse server using the UNIX operator

account. You can then enter utility commands using either of the following formats:

#### ■ Command-line format

When you use this format, you enter the entire command on a single line, then press **Enter** (↵):

*program\_name* [argument1], . . . [,argument*n*]

where *program\_name* is the name of the utility you want to execute.

For example:

recds "first=1999001,last=1999087,select=(10:13:20)"

#### ■ Interactive format

When you use this format, you type only the command name for the utility (for example, recds). Then you press **Enter** (↵) and respond to one or more online prompts, depending on the type of interactive format used by the utility. There are three basic kinds of interactive formats for internal StorHouse utilities:

Format	Description
Menu	Displays a menu of available commands and presents online instructions. See "rd utility" on page 11-1 for an example of this format.

Format	Description
Instruction	Presents the command arguments as a series of instructions and lists both the valid and default values you can specify for each argument. Either specify a value for an argument and press <code>Enter↵</code> , or press <code>Enter↵</code> without specifying a value to accept the default value.
Single prompt	<p>Presents a single prompt, which instructs you to enter commands. Because this type of format doesn't display line-by-line prompts, you must be familiar with the required input for the command. When you use this format, you type each argument and its corresponding value on a separate line and press <code>Enter↵</code> after each line. The last argument you enter must be go, which instructs the utility to process your input. The format is as follows:</p> <pre><i>program_name</i> [argument1] . . [argumentn] go</pre> <p>For example:</p> <pre>recds first=1999001 last=1999087 select=(10:13:20) go</pre>

Compare the example of the command-line format shown on page 1-5 with the example of the single prompt format shown in the table on page 1-6. Note the following differences:

- The arguments in the command-line format must be enclosed in quotation marks; however, quotation marks are not allowed when you use any of the interactive formats.
- The last argument in the single prompt interactive format must be go; this argument is not required when you use the command-line format.

To simplify the presentation of utility commands in this book, only the command-line format is described in the remaining chapters. However, if you have the option of executing a given StorHouse utility using one of the interactive formats described above, that information is noted in the chapter that describes the utility.

### **From the StorHouse command prompt (?)**

You can execute a subset of the StorHouse utilities from the StorHouse command prompt (?) by using the StorHouse Command Language RUN command. The RUN command is intended to enable StorHouse customers to perform a limited number of procedures. Because this command imposes certain limitations (for example, you can't direct the output of a RUN command to a file), it generally is not the preferred method of executing StorHouse utilities by customer support representatives.

To execute a utility using the RUN command, you first must log in to the StorHouse server operating system using the operator account. Then sign on to StorHouse (start a StorHouse session) using the system account.

The format for executing a utility using the RUN command is similar to the UNIX command-line format described on page 1-7 except that the run command precedes the utility name and the quotation marks enclosing the arguments are optional:

```
run program_name [argument1],. . . [,argumentn]
```

For example:

```
run recds first=1999001,last=1999087,select=(10:13:20)
```

If a given StorHouse utility can be executed using the RUN command, that information is noted in the chapter that describes the utility.

## Redirecting the output

When you execute a StorHouse utility using the command-line format from the StorHouse operating system (UNIX) prompt, the result (or output) of the command goes to *standard output*, which is usually your terminal. You can redirect standard output and error messages to a file using the `>&` symbols. The output is the same whether it goes to your screen or a file.

For example, the command:

```
recds "first=1999001,last=1999087,select=(10:13:20)" >& myrecds
```

dumps the selected records for the specified date range, as well as any error messages, into a file called `myrecds`. If `myrecds` already exists, the contents are overwritten.

Note that the file you specify when you redirect output is created in your current directory unless you include a pathname with the file name. For example, if your current directory is `/filetek/operator` and you specify:

```
recds "first=1999001,last=1999087,select=(10:13:20)" >& recds.log
```

the output goes to `/filetek/operator/recds.log`. But if you specify:


```
recds "first=1999001,last=1999087,select=(10:13:20)" >& /filetek/recds.log
```

the output goes to `/filetek/recds.log` regardless of your current directory.

## Getting online help

Several of the StorHouse utilities provide an online help facility. The procedure for displaying help for a utility depends on whether you request help from the StorHouse operating system (UNIX) prompt or the StorHouse command prompt (?).

## From the StorHouse operating system (UNIX) prompt

Generally, you invoke help for a utility that provides an online help facility from within the utility program. First, invoke the utility without specifying any arguments. Then type the following command and press **Enter** :

```
help
```

If the help facility subsequently presents a list of specific topics for which you can request help, retype the help command followed by the topic name and press **Enter** . For example, enter:

```
help topic
```

where *topic* is the name of the specific topic for which you want help.

Some StorHouse utilities provide a different method for accessing their related help facility. For example, to access the rd help facility, you select the H (Help) command from its main menu.

## From the StorHouse command prompt (?)

To display a list of utilities for which help is available, issue the following command from the StorHouse command prompt (?):

```
run help
```

To get help for a specific utility, issue the following command:

```
run help program_name
```

where *program\_name* is the name of the utility for which you want help.



## **add\_accounts utility**

The `add_accounts` utility creates a standard set of StorHouse accounts for a newly installed StorHouse system. Use this utility immediately after you configure a new StorHouse system using the `build_sxm` utility.

### **How add\_accounts works**

The `add_accounts` utility executes a series of StorHouse Command Language `CREATE ACCOUNT` commands to create the following StorHouse accounts:

- OPERATOR
- USER
- SYSTEM
- SYSADM
- SERVICE

In addition, `add_accounts` executes two StorHouse Command Language `CREATE GROUP` commands to create the `SERVICE` and `STH` file access groups. Each of these groups serves as the default file access group for one or more of the StorHouse accounts previously created.

A file access group is one of several properties associated with a standard StorHouse account. The following section provides a complete list of all account properties as well as the default value(s) assigned to each by the `add_accounts` utility.

## Properties of StorHouse accounts

The add\_accounts utility assigns the following properties to the standard set of StorHouse accounts at create time:

- Default password
- Set of access and command privileges
- Default environment (for the SYSTEM, SYSADM, and SERVICE accounts only), which defines default values for the account's:
  - File access group
  - Access rights to the default file access group
  - Volume set
  - File set

The following table provides an overview of the standard StorHouse accounts and their properties.

Account ID	Default password	Privileges	Default group, VSET, and FSET	Intended for
OPERATOR	OPERATOR	CONSOLE, OPERATOR, SCHEDULE, SHOW	–	System operator
USER	USER	ACCOUNT, DELETE, GET, LOCK, PUT, SHOFIELD, SHOW	–	System administrator to provide a convenient template for setting up new user accounts
SYSTEM	SYSTEM	ALLPRIVILEGE	GROUP=SERVICE VSET=SYSTEM FSET=SERVICE	System administrator

Account ID	Default password	Privileges	Default group, VSET, and FSET	Intended for
SYSADM	SYSADM	ALLPRIVILEGE	GROUP=STH VSET=SYSTEM FSET=SERVICE	StorHouse database administrator
SERVICE	SERVICE	CONSOLE, DELETE, FILE, GET, OPERATOR, PUT, RECORD, SCHEDULE, SERVICE, SETGROUP, SHOW, VTF	GROUP=SERVICE VSET=SYSTEM FSET=SERVICE	FileTek customer support representatives <sup>a</sup>

a. Use the SERVICE account to test StorHouse when you install new hardware or software.

## Format

add\_accounts

**2**

**add\_accounts utility**

---

Format

## asd utility

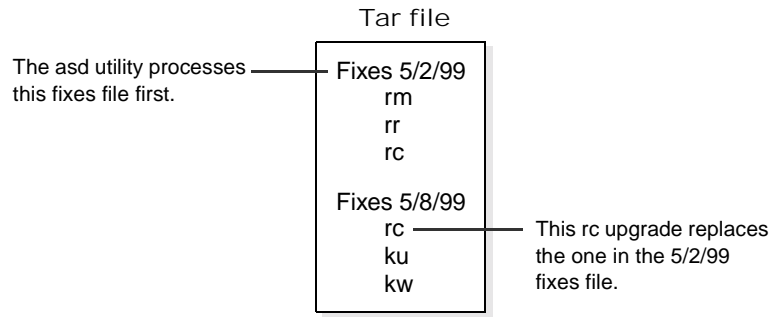
The Automated Software Delivery (asd) utility is a shell script that installs FileTek StorHouse software enhancements and fixes on existing StorHouse systems. The StorHouse startsm procedure executes the asd utility automatically. However, you can also log in to a customer's StorHouse server operating system and execute the asd utility manually.

### Overview of the asd process

The asd utility processes compressed tar files that contain replacement software files, or *fixes*. It creates several directories, which include tar extraction directories and directories for the files to be replaced and the replacement files. Before asd replaces old files, it creates a directory in /filetek/asd.old called old.*nnn*, where *nnn* is the current date in Julian format (001-365, or 366 for a leap year). Then it creates asd subdirectories such as osta, oper, lib, bin, and include (the actual directories created depend on the contents of the software delivery). Finally, asd moves each module that is being replaced into the appropriate /filetek/asd.old/old.*nnn*/*directory*, where *directory* is the name of the actual directory. (Note that asd never overwrites the files in these directories, regardless of how many times you replace a module during a given asd execution.)

Before installing new fixes files, the asd utility removes old directories and files from a previous fix if they are more than 30 days old. Then it processes each fixes file independently in chronological order. For example, in the diagram on page 2-2, the tar file contains two fixes files. The first file was created on 5/2/99 and contains upgrades for the rm, rr, and rc modules. The second file was created on 5/8/99 and contains upgrades for the rc, ku, and kw modules. The asd utility

program processes the file dated 5/2/99 first and installs the upgrades for rm, rr, and rc. Then it processes the file dated 5/8/99 and installs the upgrades for ku and kw. It also replaces the previously installed upgrade for rc with the later version.



## Including a setup script in a fixes file

Optionally, you can create a shell script named `setup` and include it in a fixes file. You can include tasks like adding a system parameter and verifying the contents of previously created recovery files in a setup script.

You may need to verify the contents of previously created recovery files before you install software upgrades. For some StorHouse files, you cannot install fixes when recovery records from a previous system shutdown still exist. The FileTek software development group identifies such files in the notes that accompany each software delivery. When you are installing upgrades for these types of StorHouse files, you can create a setup script that executes the `rxt_show` utility to check for recovery records on the customer's system. The `rxt_show` utility displays information about every record in the Resource Management Facility transfer table (Rxt). You can make the setup script subsequently abort the software upgrade depending on the results of `rxt_show`.

When you include a setup file in a fixes file, the asd utility runs it prior to installing any software replacement files in the fixes file. If setup returns a non-zero exit status, the asd utility aborts the installation of the fixes file. If an asd execution contains multiple fixes files, any fixes files that were successfully installed prior to the one that failed are unaffected. However, asd aborts the installation of all subsequent fixes files because each fix depends on the successful installation of the previous one.

Refer to page 2-9 for a sample log file that was produced when the asd utility processed three fixes files and two setup scripts. Each setup script executed `rxt_show`.

## Identifying replacement software to be installed

A UNIX cron job runs a StorHouse procedure called daily at 12:15 a.m. on every customer's StorHouse system. This procedure executes an asd check to identify replacement software to be installed and generates a report called `sw_log`, which lists (among other things) the fixes files that are pending installation. The CS server at FileTek headquarters polls customer sites, retrieves the `sw_log` report, and stores it on the FileTek CS server in `~ps/sites/site_name`, where *site\_name* is the name of the customer site.

## Adding a JUSTDOIT file

Software fixes remain in the `/usr/spool/uucppublic/filetek/asd` directory on a customer's StorHouse system until you add a JUSTDOIT file to the customer's asd directory. The JUSTDOIT file signifies that software fixes are ready to be installed and triggers the asd installation process the next time you issue an asd install command or a startsm occurs.

You create a JUSTDOIT file with the following command:

```
touch $UUCP_SPOOL/asd/JUSTDOIT
```

Alternatively, you can copy an existing JUSTDOIT file to the customer's asd directory using the uucp utility.

The size of the JUSTDOIT file is not important. It can be zero or more bytes. However, you must specify JUSTDOIT in upper case.

## Format

asd {check | install}

Argument	Description
check	Checks to see if any software delivery files are waiting to be installed. You can use this argument whether or not the StorHouse software is running.
install	<p>Checks to see if a JUSTDOIT file exists. If so, installs the software delivery files. Also does the following:</p> <ul style="list-style-type: none"><li>■ Creates an asd.log file (see page 2-7 for information about the log files)</li><li>■ Sends an email report containing the contents of the asd.log file to FileTek headquarters</li><li>■ Appends information to the asd_master.log</li><li>■ Creates backup (shadow) directories</li><li>■ Removes the JUSTDOIT file</li></ul> <p><b>Important:</b> Use the install argument only when the StorHouse software is not running.</p>



## Guidelines for creating a fixes file

Observe the following guidelines when creating a fixes file:

- Always ship the fixes file in the compressed tar file format.
- Use the following naming convention for the fixes file:

*filename.asd.Z* or *filename.asd.gz*

where *filename* is the name you assign to the file. The file name suffix (Z or gz) is determined by the UNIX program you use to compress the tar file (either compress or gzip, respectively).

- When you are preparing to create the asd fixes file, maintain the same directory structure as that used in the software delivery provided by FileTek software development. That is, place all StorHouse files that are to be installed in the osta or oper directories on the customer's StorHouse system in either osta or oper, respectively. Likewise, place all StorHouse/RM files that are to be installed in the bin, lib, or include directories in the corresponding asd subdirectory. Only include directories that contain new or replacement files in the tar file.
- If you include a setup script in a fixes file, ensure that it is executable and that it is located in the highest level of the directory that you use to create the fixes file rather than a subdirectory.
- When you download the fixes files (\*.asd.Z or \*.asd.gz) to the StorHouse system to be upgraded, always store them in the \$UUCP\_SPOOL/asd directory. The asd utility automatically installs the files in the correct directories on the customer's StorHouse system (osta, oper, and so on) after it extracts them from the tar file.

## Error handling

If an error occurs during the installation of a fixes file (for example, the system is out of disk space), the asd utility does the following:

- Uses the older versions of the files residing in `/filetek/asd.old` to revert back to the original state of the system
- Reports the error in the `asd.log` file and emails the contents of the file to FileTek headquarters
- Aborts the installation process

If you want to back out part or all of an upgrade for any reason, you can do so by using the UNIX move (`mv`) command to move each old file from `/filetek/asd.old/old.nnn/` directory to its original directory and write over the newly installed file(s) that you want to replace. To back out a newly installed file that did not previously exist on the customer's StorHouse system, use the UNIX remove (`rm`) command.

## Examples

This section presents two examples of how to run the asd utility. The first example executes an `asd check`; the second executes an `asd install`.

### Example 1

The following command:

```
asd check
```

determines whether any software delivery files are waiting to be installed. If so, the utility displays the following message:

The following ASD files are waiting to be installed:

*filename.asd.Z* or *filename.asd.gz*

⋮

where *filename* is the name you assigned to the tar file.

If no files are waiting to be installed, the utility displays the following message:

No ASD files present.

## Example 2

The following command:

```
asd install
```

checks the `/usr/spool/uucppublic/filetek/asd` directory on the customer's StorHouse system to see if a JUSTDOIT file is present. If so, it installs the software delivery files. In addition, it does the following:

- Creates an `asd.log` file
- Emails the contents of the `asd.log` file to FileTek headquarters
- Appends information to the `asd_master.log`
- Creates backup (shadow) directories
- Removes the JUSTDOIT file

## Output

The asd utility creates a log file called asd.log, which is replaced each time you use asd to install a new software delivery. The utility also appends information about the latest installation to the asd\_master.log. The following table provides more information about these files.

Log file name	Path	Data provided
asd_master.log	/filetek/operator	Date and delivery numbers for each set of deliveries installed. The most recent deliveries are listed at the end of the log.
asd.log	\$UUCP_SPOOL/asd	Site name, date and time, and list of individual files that were installed for the most recent asd execution, as well as the overall status of the asd execution.

The asd utility captures the contents of the asd.log file and emails it to FileTek headquarters. In addition, the utility automatically removes the tar delivery files after it successfully installs the software.

## Interpreting log data

The “Sample log file” section on page 2-10 illustrates a sample asd.log file. Before you examine the sample log file, however, review the following diagram and

accompanying description to learn the meaning of the various columns of data in the log file.

①	②	③	④	⑤	⑥	⑦
-rwxrwxr-x	1	operator	sm	417792	Feb 18 14:26	osta/bmds
-rwxrwxr-x	1	operator	sm	532480	Feb 2 11:03	osta/kc
-Rwxrwxr-x	1	operator	sm	589824	Feb 2 11:05	osta/ks
⋮						

### Column Identifies

1	<p>The set (read, write, and execute) of permissions assigned to the associated file (see column 7) for three different UNIX user categories: owner (characters 1 through 3), members of the group to which the file belongs (characters 4 through 6), and all other users (characters 7 through 9).</p> <p>When a hyphen (-) appears in place of an r, w, or x in this column, it signifies that the owner did not grant the associated permission to the corresponding user category. For example, <code>r-x</code> in the last three positions of this column indicates that users other than the owner or members of the file's group can read (r) and execute (x) each of the files shown in the diagram but can't write (-) to them.</p>
2	The link count, that is, how many links (names) this file has.
3	The owner of the file.
4	The group to which the file belongs.
5	The size of the file in bytes.
6	The date and time the file was last modified.
7	The target path (directory and file name) on the customer's StorHouse server.

## Sample log file

The following sample asd.log file is the output of the successful installation of a software delivery that includes three different fixes files, two of which contain setup scripts (313.setup.asd and 313.setup.oper.asd). Each setup script performs an rxt\_show procedure. The output line “Rxt is empty” in the log file indicates that the rxt\_show procedure found no recovery records. Therefore, in this example, asd continued with the installation process and successfully installed the replacement files in the fixes file.

Note that, for each fixes file that contains a setup script, the asd utility processes the setup script first, and installs the software replacement files only after successful execution of the setup script.

The spacing between the columns in the sample log file has been condensed to fit the page width of this document.

Title: Automated Software Delivery (ASD) Report

Site: anybank

Report prepared: Sat Mar 6 01:05:26 EST 1999

=====

This is the first  
fixes file.

```
Processing ASD file 311-319.asd.
-rwxrwxr-x 1 operator sm 417792 Feb 18 14:26 osta/bmds
-rwxrwxr-x 1 operator sm 532480 Feb 2 11:03 osta/kc
-rwxrwxr-x 1 operator sm 589824 Feb 2 11:05 osta/ks
-rwxrwxr-x 1 operator sm 688128 Feb 2 11:06 osta/ku
-rwxrwxr-x 1 operator sm 581632 Feb 2 11:07 osta/kw
-rwxrwxr-x 1 operator sm 2613248 Feb 15 13:12 osta/libsxm.so.1
-rwxrwxr-x 1 operator sm 1015808 Feb 18 16:36 osta/rc
-rwxrwxr-x 1 operator sm 229376 Feb 9 15:56 osta/rhb
-rwxrwxr-x 1 operator sm 253952 Feb 9 15:51 osta/rhc
-rwxrwxr-x 1 operator sm 270336 Feb 9 15:56 osta/rhm
-rwxrwxr-x 1 operator sm 278528 Feb 18 16:32 osta/rm
-rwxrwxr-x 1 operator sm 614400 Feb 9 21:46 osta/rq
-rwx----- 1 operator sm 327680 Feb 8 15:32 osta/schkp
Installation of 311-319.asd complete.
```

The second fixes  
file contains a  
setup script.

```
Processing ASD file 313.setup.asd.
Running setup script.
Rxt is empty.
Zero exit status from setup script, installation continuing.
-rwxrwxr-x 1 operator sm 204800 Feb 2 20:35 osta/maint
-rwxrwxr-x 1 operator sm 172032 Feb 5 11:08 osta/r_frmck
-rwxrwxr-x 1 operator sm 229376 Feb 2 20:36 osta/re_maint
-rwxrwxr-x 1 operator sm 172032 Feb 2 20:36 osta/re_show
-rwxrwxr-x 1 operator sm 368640 Feb 2 20:34 osta/rr
-rwxrwxr-x 1 operator sm 253952 Feb 9 16:45 osta/rr_show
-rwxrwxr-x 1 operator sm 409600 Feb 2 13:15 osta/rtm
-rwxrwxr-x 1 operator sm 319488 Feb 2 13:15 osta/rts
-rwxrwxr-x 1 operator sm 196608 Feb 2 20:36 osta/rxt_maint
-rwxrwxr-x 1 operator sm 188416 Feb 2 20:36 osta/rxt_show
-rwxrwxr-x 1 operator sm 188416 Feb 3 21:50 osta/sevx_maint
-rw-rw-r-- 1 operator sm 136241 Jan 28 16:46 osta/toolhelp.hlb
-rwxrwxr-x 1 operator sm 221184 Feb 3 20:13 osta/tpt_maint
-rwxrwxr-x 1 operator sm 204800 Feb 3 20:13 osta/tpt_show
Installation of 313.setup.asd complete.
```

The third fixes  
file also contains  
a setup script.

```
Processing ASD file 313.setup.oper.asd.
Running setup script.
Rxt is empty.
Zero exit status from setup script, installation continuing.
-r-xr-xr-x 1 operator sm 561 Feb 4 21:44 oper/r_frmck
-r-xr-xr-x 1 operator sm 561 Feb 2 17:39 oper/rr_show
-r-xr-xr-x 1 operator sm 3602 Feb 7 16:26 oper/save_files
-r-xr-xr-x 1 operator sm 562 Feb 3 20:14 oper/tpt_show
Installation of 313.setup.oper.asd complete.
```

```
Successfully installed the fixes.
Sat Mar 6 01:06:30 EST 1999
```

**3**

**asd utility**

---

Output



## AUTOMV utility

The Automatic Move (AUTOMV) utility is a shell script that installs updates or software fixes for a FileTek StorHouse system. A UNIX cron job runs a StorHouse procedure called daily at 12:15 a.m. on every customer's StorHouse system. This procedure executes the `sw_log.local` script, which in turn invokes AUTOMV. You can also execute the AUTOMV utility manually from the StorHouse operating system prompt if, for example, you want to install the update prior to the time the cron job is scheduled.

Unlike the `asd` utility, AUTOMV can run when the StorHouse software is up because the types of updates it installs (for example, updates to internal utilities like CTEST or updates to the StorHouse operating system software) do not affect the StorHouse program files. AUTOMV also differs from `asd` in that you can't use it to install updates or fixes to StorHouse/RM directories.

## Overview of the AUTOMV process

AUTOMV processes compressed tar files that contain replacement software files, or fixes. It updates the following files on a StorHouse 5.x system (the \* prefix in the file name stands for any characters):

File name	Path
*.AUTOOPER	/filetek/vxry/o/oper; /filetek2/vxry/o/oper
*.AUTOPOLARIS	/filetek/etc/polaris; /filetek2/etc/polaris
*.AUTOOBIN	/filetek/operator/bin; /filetek2/operator/bin

File name	Path
*.AUTOORUN	/filetek/operator/run; /filetek2/operator/run
*.AUTOLOGIN	/filetek/operator; /filetek2/operator
*.AUTOFETC	/filetek/etc; /filetek2/etc
*.AUTOFBIN	/filetek/bin; /filetek2/bin
*.AUTOOPBIN	/filetek/vxry/o/operator/bin; /filetek2/vxry/o/operator/bin

For each file in the list above, the AUTOMV utility renames the file and copies it to the appropriate directory specified in the Path column of the table. For example, AUTOMV would rename `sw_log.AUTOOBIN` to `sw_log` and then copy it to `/filetek/operator/bin` and `/filetek2/operator/bin`.

## Including a shell script in a fixes file

Optionally, you can create one or more shell scripts with a file name suffix of `.AUTOXQT` and include them in a fixes file. You can use `AUTOXQT` to perform any desired procedure for a specific customer or to install files that the AUTOMV utility does not accommodate. If an `AUTOXQT` script exists, the AUTOMV utility runs it prior to installing any files in the fixes file.

## Format

AUTOMV

Note that you must specify this command in upper case.

## Guidelines for creating a fixes file

Observe the following guidelines when creating a fixes file:

- Always ship the fixes file in the compressed tar file format.
- Use the following naming convention for the fixes file:

AUTOMV.TAR.*filename*.Z or AUTOMV.TAR.*filename*.gz

where *filename* is the name you assign to the file. The file name suffix (Z or gz) is determined by the UNIX program you use to compress the tar file (either compress or gzip, respectively).

- When you download the fixes files (AUTOMV.TAR.\*.Z or AUTOMV.TAR.\*.gz) to the StorHouse system to be upgraded, always store them in the \$UUCP\_SPOOL/AUTOMV directory. The AUTOMV utility automatically installs the files in the correct directories on the customer's StorHouse system (oper, polaris, and so on) after it extracts them from the tar file.

# 4

## **AUTOMV utility**

---

Guidelines for creating a fixes file

## **balance utility**

The balance utility monitors volume mount activity for each library in a StorHouse system for a specified day or date range. The utility opens each user log file associated with the time period you specify and reads all data records with a record code number of 20 (volume mount). It then produces a report that identifies the number of times each volume was mounted in each optical library for the specified time period. Volumes are listed in descending order based on how frequently they were mounted.

### **Balancing the load of volume mounts**

The balance report produced by the balance utility can assist you in performing load balancing. If the report indicates that a particular library is performing the bulk of the workload (that is, mounting the greatest number of volumes), you can use the balance utility to adjust the load of volume mounts.

When you use the balance utility in this manner, the utility evaluates the workload for the libraries you specify and identifies the library with the most mounts. Then it identifies the volume in that library that has been mounted most frequently and moves it to the library with the least mounts. By moving the most frequently mounted volume to the least active library, the balance utility distributes the workload more equitably between the libraries in a StorHouse system.

## Rules used by the load balancing process

The load balancing process observes the following rules:

- Only libraries you specify when you execute the balance utility are considered during the load balancing process.
- The utility bypasses a move if the number of mounts for the most active library is less than 1,000 or if moving the volume would increase the load imbalance between the two libraries.

## Format

balance {today | yesterday | week | month | first=yyyyjjj},[last=yyyyjjj],  
[string=nn[n...]]

You can use either the *yyyyjjj* or *yyyymmdd* format when specifying a date.

**Note:** You can also execute the balance utility in interactive mode, and by using the RUN command from the StorHouse command prompt (?).

Argument	Description
today	Specifies that the user log with the current date is the first log file to process.
yesterday	Specifies that the user log with the prior calendar day's date is the first log file to process.
week	Specifies that the first user log file to process is dated one week (7 calendar days) prior to the current date and the last user log file to process is yesterday's date.

Argument	Description
month	Specifies that the first user log file to process is dated one month (30 calendar days) prior to the current date and the last user log file to process is yesterday's date.
first=yyyymmdd or first=yyyymmdd	Calendar year and date of the first user log file to process if other than today, yesterday, week, or month. If you use the Julian date <sup>a</sup> format (yyyymmdd), specify a number from 001 to 365 or 366 following the year.
last=yyyymmdd or last=yyyymmdd	Calendar year and date of the last user log file to process when used in conjunction with the "first" argument. If you omit this argument, the default is last=first.  If you use the Julian date format, specify a number from 001 to 365 or 366 following the year.
string=nn[n...]	ID of each optical library device that you want the balance utility to consider during the load balancing process. For example, to specify that libraries L01, L02, and L03 be considered during the load balancing process, specify 123 for this argument.  If you use this argument, you must specify at least two libraries. The utility evaluates the workload for the libraries you specify and moves the most frequently mounted volume from the busiest library to the least active library.  If you omit this argument, the balance utility produces a report but does not balance the workload.

- a. The Julian date is a number assigned in sequence to each day of the year, starting with Julian date 001 for January 1 and ending with Julian date 365 (or 366 for a leap year) for December 31.

## Examples

This section provides two examples of using the balance utility.

### Example 1

The following command produces a balance report for the previous 30-day period:

```
balance month
```

### Example 2

The following command produces a balance report for the prior calendar day's date and balances the volume mounts using libraries L00, L01, L02, and L03:

```
balance string=0123,yesterday
```

### Example 3

The following command produces a balance report for the previous seven-day period and balances the volume mounts using libraries L00, L02, and L03:

```
balance string=023,week
```



## Sample report

The following command:

```
balance yesterday
```

produces the following sample balance report for the previous day's volume mounts for all optical libraries in a StorHouse system. The report is divided into two sections: summary and detail. Volumes are listed in descending order based on how frequently they were mounted during the specified time period.

```
BALANCE(4.2)
```

```
DATE-RANGE is: 1999305 thru 1999305
```

This is the  
summary  
section of  
the report.

```
→ Total mounts for L00=42
   Working set for L00=12
   Total mounts for L01=431
   Working set for L01=55
```

This is the  
volume detail  
section of the  
report.

```
→ OEBHIT00191 device=L01 vset=L01_1 mounts=10
   OEBHIT00132 device=L01 vset=L01_3 mounts=10
   OEBHIT00100 device=L01 vset=L01_6 mounts=10
   OEB2DA19A30 device=L01 vset=L01_2 mounts=9
   OEBHIT00131 device=L01 vset=L01_5 mounts=9
   OEB2E115069 device=L01 vset=L01_2 mounts=9
   OEBHIT00003 device=L01 vset=L01_5 mounts=9
   OEBHIT00124 device=L01 vset=L01_7 mounts=9
   OEBHIT00134 device=L01 vset=L01_8 mounts=9
   OEBHIT00114 device=L01 vset=L01_3 mounts=9
   OEBHIT00117 device=L01 vset=L01_7 mounts=9
   OEBHIT00127 device=L01 vset=L01_8 mounts=9
   OEBHIT00148 device=L01 vset=L01_9 mounts=9
   OEBHIT00141 device=L01 vset=L01_9 mounts=8
   OEBHIT00096 device=L01 vset=L01_0 mounts=8
   OEBHIT00122 device=L01 vset=L01_2 mounts=8
   OEBHIT00133 device=L01 vset=L01_3 mounts=8
   → OEBHIT00110 device=L01 vset=L01_1 mounts=8
```

```

OEBHIT00143 device=L01 vset=L01_4 mounts=8
OEBHIT00111 device=L01 vset=L01_9 mounts=8
OEBHIT00144 device=L01 vset=L01_2 mounts=8
OEBHIT00125 device=L01 vset=L01_4 mounts=8
OEBHIT00138 device=L01 vset=L01_1 mounts=8
OEBHIT00101 device=L01 vset=L01_0 mounts=8
OEBHIT00145 device=L01 vset=L01_6 mounts=8
OEBHIT00135 device=L01 vset=L01_5 mounts=8
OEBHIT00129 device=L01 vset=L01_8 mounts=8
OEBHIT00102 device=L01 vset=L01_9 mounts=8
OEBHIT00126 device=L01 vset=L01_4 mounts=8
OEBHIT00147 device=L01 vset=L01_2 mounts=8
OEBHIT00137 device=L01 vset=L01_1 mounts=8
OEBHIT00146 device=L01 vset=L01_6 mounts=8
OEBHIT00136 device=L01 vset=L01_5 mounts=8
OEBHIT00103 device=L01 vset=L01_0 mounts=8
OEBHIT00104 device=L01 vset=L01_9 mounts=8
OEBHIT00115 device=L01 vset=L01_3 mounts=8
OEBHIT00116 device=L01 vset=L01_7 mounts=8
OEBHIT00118 device=L01 vset=L01_4 mounts=8
OEB351911C8 device=L01 vset=L01_2 mounts=8
OEBHIT00130 device=L01 vset=L01_1 mounts=8
OEBHIT00128 device=L01 vset=L01_5 mounts=8
OEBHIT00106 device=L01 vset=L01_0 mounts=8
OEBHIT00140 device=L01 vset=L01_3 mounts=8
OEBHIT00107 device=L01 vset=L01_7 mounts=8
OEBHIT00121 device=L01 vset=L01_1 mounts=8
OEBHIT00109 device=L01 vset=L01_8 mounts=7
OEB31725FEA device=L01 vset=L01_7 mounts=7
OEBHIT00139 device=L01 vset=L01_6 mounts=7
OEBHIT00119 device=L01 vset=L01_8 mounts=7
OEBHIT00120 device=L01 vset=L01_4 mounts=7
OEBHIT00108 device=L01 vset=L01_0 mounts=7
OEBHIT00142 device=L01 vset=L01_6 mounts=6
TDA000111 device=L00 vset=L00_4 mounts=5
TDA000116 device=L00 vset=L00_9 mounts=5
TDA001273 device=L00 vset=L00_2 mounts=5
TDA000112 device=L00 vset=L00_3 mounts=5
TDA000114 device=L00 vset=L00_6 mounts=4

```

TDA000115	device=L00 vset=L00_8	mounts=4
TDA000113	device=L00 vset=L00_7	mounts=4
OEB2DD8A6F6	device=L01 vset=USERLOG	mounts=3
TDA000110	device=L00 vset=L00_5	mounts=3
TDA000119	device=L00 vset=L00_5	mounts=3
OEB2FAF7996	device=L01 vset=EXTRACT	mounts=2
TDA000118	device=L00 vset=L00_10	mounts=2
OEBHIT00092	device=L01 vset=MZAGUREK	mounts=2
TDA001271	device=L00 vset=L00_0	mounts=1
TDA001272	device=L00 vset=L00_1	mounts=1

## Interpreting the report data

The summary section of the balance report identifies the total number of mounts and the working set for each library device. A *working set* is the total number of different volumes that were mounted for a specific library device. For example, a working set of 12 and total mounts of 42 for library L00 indicate that 12 different volumes were mounted in L00 a total of 42 times. You can determine the number of times each of the 12 volumes was mounted by reviewing the quantity that appears in the right-most (mounts) column of the volume detail section of the report.

The following table identifies the data in the detail section of the report.

Column	Description
1	Volume ID of each volume that was mounted during the specified time period.
2	Library device on which a volume was mounted.
3	Name of the volume set to which a volume belongs.
4	Number of times the volume was mounted.

**5**

**balance utility**

---

Sample report

## bmdi utility

The Base Facility Magnetic Disk Initialization (bmdi) utility enables you to perform the following tasks after installing or replacing a level F device in a StorHouse system:

- Initialize all or selected level F devices
- Scan all level F devices to determine if all disks are operating correctly

When you perform a level F scan, bmdi accesses every configured level F device and produces either a report for each disk or an error message if a disk is inaccessible or initialized incorrectly.

## Format

bmdi [all | *dev\_name* | scan]

Argument	Description
all	Initializes all level F devices.

Argument	Description
<code>dev_name</code>	Initializes the specified level F device.  <b>Warning:</b> If you added new level F partitions to an existing StorHouse system, always use this argument when you issue the bmdi command so you initialize only the new devices.
<code>scan</code>	Scans all level F devices for which the label can be read to ensure that the devices are operating correctly. You can perform this operation at any time with no ramifications to the customer's operations.

## Sample output

The following command:

```
bmdi scan
```

results in the sample output below:

BMDI - Magnetic Disk Initialization Utility.

F00 is labeled and has been used.

Library = 00, unit = 00

Device file = "/dev/rdisk/rh0c0d0c".

Previously labeled at 26-JAN-1998:04:49:08.

F01 is labeled and has been used.

Library = 01, unit = 00

Device file = "/dev/rdisk/rh0c1d0c".

Previously labeled at 26-JAN-1998:04:49:19

The Device file line contains the internal operating system path to the partition for the disk.

If you issued the bmdi scan command for a newly initialized level F device, the first line of the output would look similar to the following:

F01 is labeled and has not been used.

This line merely indicates that data has not yet been stored on the device.

## **build\_sxm utility**

The `build_sxm` utility configures a new StorHouse system. Use this utility following the installation of the UNIX operating system and the StorHouse software on a new StorHouse server.

In addition to deleting and creating various files, the `build_sxm` utility executes a number of internal StorHouse utilities, such as `gen_dev`. This chapter provides a high-level description of the role that each of these utilities plays in the StorHouse installation process. For additional information on these utilities, refer to the chapters that describe them.

The following section lists and briefly describes each of the major tasks that `build_sxm` performs.

### **build\_sxm tasks**

The `build_sxm` utility performs the following high-level tasks:

- Prompts you to confirm deletion of existing system files and the catalog of user files
- Checks for the existence of the `config_dev` and `spx_maint.inp` configuration files in the `$BUILD` directory and, using the `gen_dev` check function, compares the device information in both files for consistency

- Creates the \$BUILD/config.sh file, which gen\_dev uses as a log file to record commands executed by various script files in subsequent phases of the gen\_dev process
- Deletes the contents of all SMD directories, which contain existing system files
- Initializes the permanent storage area (NVM or disk file) on the new StorHouse system
- Generates system files for the Administration (A) and Directory manager (D) facilities
- Calls the gen\_dev utility to build the Resource management (R), Storage allocation manager (J), and Basic device support (B) facility system files (refer to Chapter 23 for more information on the gen\_dev utility)
- Invokes the genqmm utility to build Quality Maintenance Manager (qmm) system files (refer to Chapter 24 for more information on the genqmm utility)
- Invokes the spx\_maint utility using the spx\_maint.inp file (which contains StorHouse/SM-specific system parameters and definitions) as input
- If the \$STH\_RELEASE environment variable is defined on the current StorHouse system, invokes the spx\_maint utility using the spx\_maint.sth.inp file (which contains StorHouse/RM-specific system parameters and definitions) as input
- Copies the correct version of the smc file (which is used by StorHouse to control system startup) into \$SMD\_PRIMARY and \$SMD\_SECONDARY, based on whether StorHouse/RM is installed on the system
- Generates various other files and system files



- Invokes the bmdi utility with the all argument to initialize all level F devices (refer to Chapter 6 for more information on the bmdi utility)
- Exits

## Before you use build\_sxm

As indicated in “build\_sxm tasks” on page 7-1, build\_sxm invokes the gen\_dev utility to build a set of StorHouse system files. The gen\_dev utility has two generic input files, config\_dev and spx\_maint.inp, which you must modify prior to execution of gen\_dev to reflect the customer’s StorHouse configuration. Refer to Chapter 23, “gen\_dev utility,” for more information on modifying these files.

## Format

build\_sxm

**7**

**build\_sxm utility**

---

Format

## **cfg\_modem utility**

The `cfg_modem` utility configures a modem for a StorHouse system. The task of configuring a modem is necessary to enable FileTek customer support representatives and other authorized support personnel to dial in to a customer's site remotely. In addition, if a customer plans to implement the Call Home feature using a modem rather than the Internet, you must initialize the modem using this utility.

The configuration process consists of the following tasks:

- Tailoring the configuration template, `modem_config`, to a customer's site
- Preparing the modem for initialization
- Initializing the modem
- Implementing modem security
- Enabling or disabling remote access to a customer's StorHouse system

Initially, use this utility following the installation of a new StorHouse system. Subsequently, use this utility when you want to change the modem password, change the type of modem security in effect, or deny remote access to a customer's StorHouse system.

## Input to the cfg\_modem utility

The `cfg_modem` utility uses an input file, `modem_config`, during the modem configuration process. This file, which resides in the `/filetek/etc` directory on each customer's StorHouse server, is a configuration template that you edit and tailor to reflect the type of modem installed on a customer's StorHouse system. In addition to the modem type, this file describes details such as the full path name of the special device file to be used to access the modem and, if modem callback security is enabled (see "Implementing modem security"), the prefix and suffix used in conjunction with the callback telephone number.

When you execute the `cfg_modem` utility, it checks to verify the existence of the `modem_config` file. The utility then reads the data in the configuration file and, based on the modem type specified in the file, completes the specified task.

## Implementing modem security

The `cfg_modem` utility provides two types of security: password security and callback security. Modem security is intended to prevent unauthorized users from gaining remote access to a customer's StorHouse system. You must implement either password or callback security during the modem configuration process.

Password security requires that FileTek's dial-in procedure supply the password you designate during the modem configuration process when dialing in to a customer's site. To implement this form of security, specify a password (other than "callback") when you initialize the modem using the `cfg_modem` utility.

Callback security represents a higher level of security. After FileTek's dial-in procedure supplies a system-defined (internal) password and establishes a connection with a customer's StorHouse system, the customer's modem terminates the call. It then retrieves the telephone number associated with the password from its modem configuration records (which are created when you run the `cfg_modem` utility) and returns the call. When FileTek's modem

responds to the incoming call, that serves as confirmation that FileTek attempted to establish a connection with the customer's StorHouse system.

To implement callback security, do the following:

- Ensure that the `modem_config` configuration file identifies any prefix or suffix that the customer's modem must dial to complete the call to FileTek.
- Specify a value of "callback" for the password argument when you initialize the modem.

## Enabling or disabling remote access

The `cfg_modem` utility provides a feature that allows you to grant or deny remote access to a StorHouse system via the telephone lines. This feature (referred to as auto-answer), provides the highest degree of control to customers over access to their systems by allowing them to deny dial-in access unless they specifically authorize it (for example, when customer support needs to perform system maintenance). A customer can temporarily enable the auto-answer feature to allow customer support to dial in and perform the maintenance, then disable auto-answer upon completion of the maintenance.<sup>a</sup>

The `cfg_modem` utility enables the auto-answer capability by default when you initialize a modem. Therefore, unless you subsequently deny remote access to a customer's StorHouse system using the disable option, a customer's modem automatically answers any and all incoming calls (however, the connection is not complete until the calling modem supplies the correct password or successfully concludes the callback security test). When you disable the auto-answer feature, the customer's modem ignores all incoming calls, regardless of the source, and customer support is unable to dial in to the customer's StorHouse system

---

a. At this time, the only way StorHouse customers can disable or enable the auto-answer feature is by using the `cfg_modem` utility. A script does not yet exist that would allow customers to perform these tasks using a `RUN` command.

remotely to perform maintenance. Note that the customer's ability to initiate outgoing calls is not affected by this option.

## Format

Note that, although you can run the `cfg_modem` utility in batch mode, you must execute it interactively if you're initializing a modem of type T8820.

`cfg_modem {prepare | initialize {password} | password {password} | disable | enable}`

Argument	Description
<code>prep[are]</code>	Displays online instructions to enable you to prepare the modem for initialization. The instructions vary depending on the model of the modem specified in the modem configuration file ( <code>/filetek/etc/modem_config</code> ).
<code>init[ialize] <i>password</i></code>	<p>Initializes the modem for use and identifies the type of modem security that you are implementing. If you are using password security, this argument sets the initial dial-in password. Use this argument only after you have prepared the modem for initialization. You must specify a value for <i>password</i> when you use this argument.</p> <p>When you initialize a modem, the <code>cfg_modem</code> utility enables the auto-answer capability by default.</p>
<code>pass[word] <i>password</i></code>	Changes either the type of modem security enabled for an initialized modem or the current dial-in password (if you are using password security). You must specify a value for <i>password</i> when you use this argument.
<code>disa[ble]</code>	Disables the auto-answer capability, thereby denying remote access to a StorHouse system via the telephone lines.

Argument	Description
enab[le]	Enables the auto-answer capability, thereby granting remote access to a StorHouse system via the telephone lines.
password	<p>Value that defines the type of modem security you are implementing. Valid values are:</p> <ul style="list-style-type: none"><li>■ A non-trivial password (other than “callback”) of up to 8 characters (which can include mixed case letters) – Enables password security and sets the dial-in password that FileTek must supply when it initiates a call to a customer’s StorHouse system.</li><li>■ “callback” – Enables modem callback security. This value specifies that FileTek’s dial-in procedure must supply a system-defined (internal) password when it initiates a call to a customer’s StorHouse system.</li></ul>

## Examples

This section presents four examples of how to use the `cfg_modem` utility.

### Example 1

The following command initializes a previously prepared modem, enables password security, and enables the auto-answer capability by default:

```
cfg_modem initialize answerme
```

### Example 2

The following command initializes a previously prepared modem, enables modem callback security rather than password security, and enables the auto-answer capability by default:

```
cfg_modem initialize callback
```

### **Example 3**

The following command changes the current modem password to secret:

```
cfg_modem password secret
```

### **Example 4**

The following command disables the auto-answer capability for a customer's modem, thereby denying remote access to the StorHouse system via the telephone lines:

```
cfg_modem disable
```



## cfg\_net utility

The `cfg_net` utility configures network-related aspects of a StorHouse system. The utility performs the following tasks:

- Removes all old output files from previous executions of `cfg_net`
- Prepares the system to use a particular routing protocol
- Configures StorHouse for a non-networked system, if applicable
- Generates the appropriate output files based on the type of network

Use this utility during the installation process for a new StorHouse system.

You must log in to the StorHouse server operating system as superuser (using the root account) to execute this utility. In addition, you should be familiar with the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol prior to executing this utility.

After you execute the `cfg_net` utility, reboot the StorHouse server to activate the changes.

## Input to the cfg\_net utility

The `cfg_net` utility uses an optional input file, `net_config`, during the network configuration process. This file, which resides in the `/filetek/etc` directory on each customer's StorHouse server, is a configuration template that you edit and tailor to reflect the actual network environment in which the StorHouse system will operate. This file describes details such as the node name (hostname) of the StorHouse system (for example, Alpha2), the type and IP address of each network interface used in the customer's network, and the routing protocol that is used. When you execute the `cfg_net` utility, it checks to determine the existence of the `net_config` file. If the file exists, the utility reads the data in the file and configures the network based on the information specified in the file.

FileTek recommends that you configure the network information for StorHouse using a network configuration file (to facilitate subsequent changes to the configuration). However, you can run the `cfg_net` utility using one of the arguments provided with the utility when StorHouse will be operating in a non-networked or simple networked environment. (Generally speaking, a simple networked environment is one in which the configuration consists only of a node name and a single Ethernet interface.)

## Format

cfg\_net [*name* [*IPaddr* | *bay#*]]

Argument	Description
[no argument]	Configures the network information for StorHouse using the data you specified in the network configuration file, /filetek/etc/net_config.
<i>name</i>	Node name (hostname) of the StorHouse system. When used alone, this argument specifies that StorHouse is part of a non-networked system. You can use this argument in lieu of a network configuration file.
<i>IPaddr</i>	<p>IP address of the StorHouse server. When StorHouse is being added to a customer's existing network, the customer usually supplies this value.</p> <p>You can configure StorHouse using this argument (in conjunction with the <i>name</i> argument) in lieu of a network configuration file when the StorHouse system you are configuring is part of a simple networked system.</p>
<i>bay#</i>	<p>Number that represents the FileTek Manufacturing bay in which the StorHouse system is located. Valid values are:</p> <ul style="list-style-type: none"><li>■ 1 - 9</li><li>■ 1a - 9a</li></ul> <p>You can configure StorHouse using this argument (in conjunction with the <i>name</i> argument) in lieu of a network configuration file when the StorHouse system you are configuring is part of a simple networked system and is located in one of FileTek's Manufacturing bays.</p>

## Examples

This section presents three examples of how to use the `cfg_net` utility.

### Example 1

The following command configures the network information for StorHouse using the data you specify in the network configuration file, `/filetek/etc/net_config`:

```
cfg_net
```

### Example 2

The following command configures StorHouse as part of a simple networked system:

```
cfg_net abcbank1 192.0.1.27
```

### Example 3

The following command configures StorHouse as part of a system in FileTek Manufacturing:

```
cfg_net FileTek1 1a
```

## cleanipcs utility

The Clean Interprocess Communication Services (cleanipcs) utility cleans up the following shared system resources on a StorHouse/RM system:

- Shared memory
- Semaphores
- Message queues

The utility destroys the shared locking environment that StorHouse/RM creates and releases the shared resources back to the UNIX operating system.

Ordinarily, you execute cleanipcs manually from the StorHouse operating system (UNIX) prompt when you are upgrading a StorHouse/RM system from one release of the software to another. However, the startsm utility also executes cleanipcs automatically each time you run it on a StorHouse/RM system.

**Important:** You must always shut down the StorHouse software using the stopsm utility prior to executing the cleanipcs utility.

## How to use cleanipcs

Each new release of StorHouse/RM provides an updated version of the cleanipcs utility that is specifically designed to clean up after that release level of the software. The executable for the utility is installed in /filetek/sth/v2r $x$ /bin, where  $x$  corresponds to the release level of the software.

Because the utility is release-dependent, never attempt to clean shared structures from one version of StorHouse/RM using a different version of cleanipcs. Ensure that the version of cleanipcs you are using corresponds to the release level of the StorHouse/RM software on the system. For example, if you are upgrading a system from v2r1 to v2r2, do the following:


1. Execute stopsm on the StorHouse/RM v2r1 system.
2. Execute the v2r1 version of cleanipcs.
3. Upgrade the system to StorHouse/RM v2r2.

The next time you run startsm on the upgraded system, it will execute the v2r2 version of cleanipcs automatically.

After you issue the cleanipcs command, the UNIX command prompt returns, provided cleanipcs completes successfully. If cleanipcs finishes with errors (for example, it returns an error message or a return code of “-1”) or it doesn’t finish at all (the UNIX command prompt never reappears), you must reboot the StorHouse system.

## Format

sxm cleanipcs

**Note:** The sxm utility executes the version of cleanipcs that corresponds to the release level of StorHouse/RM defined for SSTH\_RELEASE. If you want to execute a different version of cleanipcs, change to /filetek/sth/v2r $x$ /bin, where  $x$  corresponds to the version of cleanipcs that you want to execute (for example, v2r1). Then type the following command and press **Enter** :

./cleanipcs

Make sure you type `./` preceding the command to instruct UNIX to execute the cleanipcs utility from the current directory.





## cpshow utility

The cpshow utility presents the current status of a StorHouse system. In addition to listing each StorHouse process that is currently running, the cpshow utility displays data such as the percentage of CPU and memory consumed by each process as well as the date or time each process was started. The output from cpshow also identifies the name that StorHouse assigned to each active process. The process name is required input when you want to stop a particular process using cpstop.

You can also query the status of a specific StorHouse process if you know the process name.

### Format

cpshow [*process\_name* ...]

Argument	Description
[no argument]	Displays the status of all active StorHouse processes.
<i>process_name</i>	Name of the StorHouse process(es) for which you want to display the status.

## Sample output

The cpshow utility executes the UNIX process status (ps) command (with the u, n, g, x, and w options) and displays the results (with minor modifications) as output. Therefore, if you need assistance in interpreting the data shown in the sample output, enter the following command to display the UNIX man help for the ps command:

```
man -s 1b ps
```

The following sections illustrate the sample output that results when you issue the cpshow command both with and without a value for the *process\_name* argument.

### Sample 1

The following command:

```
cpshow bmon_dev
```

results in the sample output below:

```
System Status as of Wed Jan 19 11:37:30 2000
```

PROCESS	PID	%CPU	%MEM	SZ	RSS	S	START	TIME	PROGRAM
bmon_dev	22240	0.0	0.3	4520	1456	S	Jan 16	0:00	/filetek/v5r2/o/osta/bmon_dev

### Sample 2

The following sample output results when you issue the cpshow command with no argument (the output is not shown in its entirety to conserve space):

```
System Status as of Wed Jan 19 11:26:30 2000
```

PROCESS	PID	%CPU	%MEM	SZ	RSS	S	START	TIME	PROGRAM
acol	22275	0.0	0.3	4560	1544	S	Jan 16	0:31	/filetek/v5r2/o/osta/acol
al	22306	0.0	0.4	4584	1568	S	Jan 16	0:40	/filetek/v5r2/o/osta/al

alsp	22307	0.0	0.4	3904	1576	S	Jan 16	0:04	/filetek/v5r2/o/osta/alsp
am	22312	0.0	0.3	3504	1488	S	Jan 16	0:01	/filetek/v5r2/o/osta/am
bm000000	22301	0.0	0.4	5552	1920	S	Jan 16	0:01	/filetek/v5r2/o/osta/bmnds
dm	22313	0.0	0.4	4640	1696	S	Jan 16	1:10	/filetek/v5r2/o/osta/dm
eam	22495	0.0	0.5	6040	2320	S	Jan 16	0:01	/filetek/sth/v2r2/bin/eam
ec	22494	0.0	0.3	3552	1448	S	Jan 16	0:00	/filetek/sth/v2r2/bin/ec
gc	22488	0.0	0.3	3504	1368	S	Jan 16	0:00	/filetek/v5r2/o/osta/gc
gcd3	22490	0.0	0.3	3504	1488	S	Jan 16	0:00	/filetek/v5r2/o/osta/gcd
gcd3001	18000	0.0	0.3	2672	1344	S	Jan 18	0:00	/filetek/v5r2/o/osta/gcds
gcd3002	18174	0.0	0.4	2672	1616	S	11:01:07	0:00	/filetek/v5r2/o/osta/gcds
gct1	10880	0.1	0.3	3776	1168	S	11:32:32	0:28	/filetek/v5r2/o/osta/gct
gct1	11889	0.0	0.3	3776	1200	S	08:53:30	0:00	/filetek/v5r2/o/osta/gct
gct1	14954	0.0	0.3	3776	1200	S	09:55:30	0:00	/filetek/v5r2/o/osta/gct
gct1	18328	0.0	0.3	3776	1200	S	11:04:45	0:00	/filetek/v5r2/o/osta/gct
gct1	22489	0.0	0.4	3776	1696	S	Jan 16	0:02	/filetek/v5r2/o/osta/gct
gct1	22609	0.0	0.3	3776	1168	S	Jan 16	0:40	/filetek/v5r2/o/osta/gct
gct1	22802	0.0	0.3	3776	1168	S	Jan 16	0:52	/filetek/v5r2/o/osta/gct
kc000	22480	0.0	0.3	3184	1336	S	Jan 16	0:00	/filetek/v5r2/o/osta/kc
ki	22478	0.0	0.3	2512	1432	S	Jan 16	0:00	/filetek/v5r2/o/osta/ki
kn	22484	0.0	0.3	2472	1368	S	Jan 16	0:00	/filetek/v5r2/o/osta/kn
ks000	22483	0.0	0.4	3360	2008	S	Jan 16	0:07	/filetek/v5r2/o/osta/ks
ku000	7888	0.0	0.3	5440	1448	S	Jan 17	0:00	/filetek/v5r2/o/osta/ku
kw000	22481	0.0	0.4	4312	1944	S	Jan 16	0:24	/filetek/v5r2/o/osta/kw
lc	22499	0.0	0.6	6792	2928	S	Jan 16	0:00	/filetek/sth/v2r2/bin/lc
lfl	11848	0.0	0.5	3104	2360	S	08:52:57	0:00	/filetek/sth/v2r2/bin/lfl
qchc	22238	0.0	0.3	2496	1416	S	Jan 16	0:00	/filetek/v5r2/o/osta/qchc
qmm	22309	0.0	0.4	3688	1712	S	Jan 16	0:02	/filetek/v5r2/o/osta/qmm
rc	22318	0.0	0.5	5912	2512	S	Jan 16	3:05	/filetek/v5r2/o/osta/rc
rf	22475	0.0	0.4	3528	1640	S	Jan 16	0:00	/filetek/v5r2/o/osta/rf
rhb	22314	0.0	0.3	3528	1488	S	Jan 16	0:00	/filetek/v5r2/o/osta/rhb
rhc	22315	0.0	0.3	3512	1432	S	Jan 16	0:00	/filetek/v5r2/o/osta/rhc
rhm	22316	0.0	0.3	3528	1440	S	Jan 16	0:00	/filetek/v5r2/o/osta/rhm
rl300	22459	0.0	0.5	5640	2272	S	Jan 16	0:00	/filetek/v5r2/o/osta/rl
rl301	22460	0.0	0.6	7448	2600	S	Jan 16	0:02	/filetek/v5r2/o/osta/rl
rl302	22461	0.0	0.4	4560	1936	S	Jan 16	0:00	/filetek/v5r2/o/osta/rl
rm010000	22399	0.0	0.4	4480	1744	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
rm030000	22389	0.0	0.4	5696	1984	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
rm030001	22390	0.0	0.4	5672	1968	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
rm030010	22392	0.0	0.5	5792	2472	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
rm030011	22393	0.0	0.4	5792	1960	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
rm030012	22394	0.0	0.4	4768	1952	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
rm030013	22395	0.0	0.4	5792	1960	S	Jan 16	0:01	/filetek/v5r2/o/osta/rm
rm030020	22398	0.0	0.4	4480	1744	S	Jan 16	0:00	/filetek/v5r2/o/osta/rm
ro300	22388	0.0	0.4	5600	1968	S	Jan 16	0:00	/filetek/v5r2/o/osta/ro
ro301	22391	0.0	0.4	5544	1952	S	Jan 16	0:00	/filetek/v5r2/o/osta/ro
ro302	22396	0.0	0.4	5528	1744	S	Jan 16	0:00	/filetek/v5r2/o/osta/ro
rq	22462	0.0	0.6	7672	2760	S	Jan 16	0:19	/filetek/v5r2/o/osta/rq
rr	22476	0.0	0.4	5560	1896	S	Jan 16	0:00	/filetek/v5r2/o/osta/rr
rt0m000	22319	0.0	0.7	6568	3240	S	Jan 16	0:18	/filetek/v5r2/o/osta/rtm
rt0m001	22320	0.0	0.5	6568	2104	S	Jan 16	0:00	/filetek/v5r2/o/osta/rtm

```

rt0m002 22321 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m003 22322 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m004 22323 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m005 22324 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m006 22325 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m007 22326 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m008 22327 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m009 22328 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m010 22329 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m011 22330 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m012 22331 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m013 22332 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m014 22333 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m015 22334 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m016 22335 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m017 22336 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m018 22337 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m019 22338 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m020 22339 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m021 22340 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m022 22341 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m023 22342 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m024 22343 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m025 22344 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m026 22345 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m027 22346 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m028 22347 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m029 22348 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m030 22349 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m031 22350 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m032 22351 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m033 22352 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m034 22353 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m035 22354 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m036 22355 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m037 22356 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m038 22357 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m039 22359 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m040 22360 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m041 22361 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m042 22362 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m043 22363 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
rt0m044 22366 0.0 0.4 5120 1712 S Jan 16 0:00 /filetek/v5r2/o/osta/rtm
soi 22283 0.0 0.3 2568 1496 S Jan 16 0:00 /filetek/v5r2/o/osta/soi
soit0 18329 0.0 0.4 2448 1560 S 11:04:45 0:00 /filetek/v5r2/o/osta/soit
soit2 14955 0.0 0.4 2448 1560 S 09:55:31 0:00 /filetek/v5r2/o/osta/soit
tc 22485 0.0 0.4 2520 1640 S Jan 16 0:00 /filetek/v5r2/o/osta/tc
...
...
...

```

## cpstop utility

The cpstop utility stops one or more StorHouse processes. You can use this utility to kill all active processes if, for example, the StorHouse system didn't shut down cleanly.

Generally, when multiple copies of a StorHouse process are running concurrently, each copy has a unique process name. To identify the name of the specific instance of the process that you want to stop, use the cpshow utility. Refer to Chapter 11, “cpshow utility,” for information on using that utility.

### Format

You must specify at least one option.

cpstop {-avkdsc} [*process\_name* ...]

Option/Argument	Description
-a	<p>Performs the specified action (kill, dump, suspend, or continue) on all StorHouse processes. If you stop a process and multiple copies of that process are running, cpstop terminates all of them.</p> <p><b>Warning:</b> Because this option doesn't perform a “clean” shutdown, you should use it only as a last resort, for example, to stop all processes when a StorHouse system isn't responding to normal shutdown procedures.</p>
-v	Identifies the name of each process that cpstop operates on.

Option/Argument	Description
-k [ <i>process_name</i> ]	Kills the specified StorHouse process(es) as quickly and abruptly as possible.
-d [ <i>process_name</i> ]	Produces a core dump of the memory being used by the specified StorHouse process(es) so someone can analyze the cause of failure.
-s [ <i>process_name</i> ]	Suspends the specified StorHouse process(es).
-c [ <i>process_name</i> ]	Continues one or more previously suspended process(es).
<i>process_name</i>	Name of the StorHouse process(es) that you want to kill, dump, suspend, or continue.

## Examples

This section presents three examples of how to run the cpstop utility.

### Example 1

The following command stops all StorHouse processes and displays the name of each process stopped:

```
cpstop -av
```

### Example 2

The following command suspends a specific copy of the rt process and displays the message “Stopping process rt0m000”:

```
cpstop -sv rt0m000
```

### **Example 3**

The following command resumes (continues) the rt process that was suspended in example 2:

```
cpstop -cv rt0m000
```





## **ctest utility**

The Confidence Test (ctest) utility ensures that optical disk units (ODUs), magnetic tape devices, and magnetic disk units (MDUs) can write and read data within acceptable thresholds. You can test specific drives or all drives in one or more libraries.

Use ctest in the following situations:

- Before shipment of a StorHouse system
- Following the repair or replacement of a tape or optical drive
- Following the installation of a new tape or optical library
- Following the installation of a new software release
- During preventive maintenance
- For diagnostic purposes

A successful ctest execution indicates that the library is configured correctly for use by StorHouse.

## **How ctest works**

The ctest utility executes StorHouse Callable Interface (LSMxxx) functions and StorHouse Command Language commands. The utility runs in interactive mode. After you execute ctest from the StorHouse operating system (UNIX) prompt, it displays a series of instructions, or prompts, to which you respond. As the utility runs, it displays status messages to inform you of its progress.

In order to use the ctest utility, the StorHouse software must be running. When you execute ctest and specify the device to test, ctest “reserves” the device for its use. Because StorHouse users are unable to access the device until ctest execution completes, FileTek recommends that you do not run ctest during production use. If you must run ctest during normal business hours, obtain the approval of the StorHouse system administrator at the customer site before you execute the utility.

Online help is available for several of the ctest prompts (the “Format” section beginning on page 13-6 specifies which prompts have help). To display online help for a prompt, type ? following the prompt and press Enter ↵.

## Primary steps performed by ctest

The ctest utility performs the following steps:

Step	Description
1	Accepts and validates the value you specify for each argument.
2	Determines whether the StorHouse software is running.
3	If the StorHouse software is running, establishes a session with StorHouse and ensures that the StorHouse software is set up for testing (validates the account and gets its default group, VSET, and FSET). Otherwise, ctest aborts.
4	Reserves the device you select to test for maintenance (if it's not already in a reserved state). This step ensures that users cannot access the device while you are executing ctest.  If the library containing the drives to be tested is in a down state when you execute ctest, ctest sets all drives within the library to maintenance mode, then brings up the library.
5	Writes a known data pattern to one or more StorHouse SEQUENTIAL files on the device you specify. (Refer to the table on page 13-3 for a list of the files ctest can create.)
6	Reads the records in the SEQUENTIAL file and compares the results to what was written to ensure the data is identical; displays any errors.

Step	Description
7	Removes the device from a reserved state. If the library was down when you executed ctest, ctest returns it to a down state, then unreserves the drives it reserved in step 4.
8	Disconnects from StorHouse.
9	Exits.

## Files created by ctest

When you use the ctest utility in write mode, ctest generates a StorHouse SEQUENTIAL file. The file name is determined by the file size you specify when you execute ctest as well as whether you enable error detection coding (EDC).

The following table lists and describes the types of StorHouse files that ctest can create.

This file...	Is a...
X1KB_WRT	1KB file
X1KB_WRT_EDC	1KB file with EDC enabled
X500KB_WRT	500KB file
X500KB_WRT_EDC	500KB file with EDC enabled
X10MB_WRT	10MB file
X10MB_WRT_EDC	10MB file with EDC enabled
X2GB_WRT	2GB file
X2GB_WRT_EDC	2GB file with EDC enabled

## Where ctest writes data

When the ctest utility writes data to StorHouse, it uses the default file set and volume set for the StorHouse account as the destination for the data write operation. The account that ctest uses depends on the library you select to test. The following table lists the StorHouse accounts and the corresponding default volume set and file set that is normally defined for each. The hexadecimal value *n* in the table corresponds to the ID of the library you are testing when the library is other than L00.

**Where ctest writes data**

Library	Account	Default VSET	Default FSET
F00	SERVICE	MAGDISK	\$\$BUFFER
L00	SERVICE	SYSTEM	SERVICE
L0 <i>n</i>	SERVICE <i>n</i>	SYSTEM <i>n</i>	SERVICE

You can view the default values for an account by logging into StorHouse using the SYSTEM account and entering the following command at the StorHouse command prompt (?):

```
? SHOW ACCOUNT {account} /FULL
```

## Before you use ctest

StorHouse can disable and/or writelock volume sides that contain excessive media errors. If you neglect to enable or unwritelock such a volume before you perform ctest and no other volume surfaces are available for the volume set associated with the device you select to test, StorHouse allocates a new volume to that volume set from the free pool. As a result, you could potentially write to multiple volumes when using ctest to troubleshoot a problem ODU.

To prevent ctest from extending the default volume set for the device you want to test (and possibly using multiple volumes unnecessarily), use the StorHouse Command Language SET VSET command to reduce the value for the /LIMIT attribute prior to performing ctest (you can reset the value of /LIMIT after ctest completes). This action causes the ctest utility to issue an error message when the /LIMIT attribute prevents it from writing a file to the volume set (see “Unable to create the test file” on page 13-9). The message serves as a reminder that you should execute the /UNWRITELOCK and/or /ENABLE commands for the most recently used volume, if necessary, before proceeding with ctest.

Use the following StorHouse Command Language commands, as necessary, to view or change values for volume sets:

Use this command	To
? SHOW VOLUME * /VSET={vset_name} /FULL	Display the status and location of all volumes in a VSET
? SHOW VSET {vset_name} /FULL	Display the current size and limit for a VSET
? SET VSET {vset_name} /LIMIT={new_value}	Modify the value of limit for a VSET
? SET VOLUME {vid} /UNWRITELOCK	Unwritelock one or more writelocked volume sides
? SET VOLUME {vid} /ENABLE	Enable a disabled volume

Refer to the StorHouse *Command Language Reference Manual*, publication number 900005, for a detailed description of these and other commands.

## Format

ctest

The utility displays a series of prompts, which are described in the table below. Valid values for each prompt are enclosed in parentheses following the prompt. In addition, the default value for each prompt is enclosed in angle brackets (< >) following the prompt. To accept the default value for a given prompt, press  without specifying a value. To specify a value other than the default value, type the desired value and press .

**Prompts displayed by the ctest utility**

Argument	Description	Default
Number of test iterations	Number of times you want to read and/or write the test file on the specified drive. Each iteration creates a new version of the test file. Valid values are 1 through 10000.	1
Drive to test	<p>ID of the library or Level F drive that you want to test. Valid values are:</p> <ul style="list-style-type: none"> <li>■ LxxDyy – Test only drive yy in library xx</li> <li>■ Lxx* – Test all available drives in library xx</li> <li>■ Lxx_ – Test a random drive in library xx</li> <li>■ L* – Test all available drives in library L00</li> <li>■ L_ – Test a random drive in library device L00</li> <li>■ F_ – Test a random level F (fixed) device drive</li> </ul> <p>where:</p> <ul style="list-style-type: none"> <li>■ xx is the device unit number</li> <li>■ yy is the drive number</li> </ul> <p>Online help is available for this prompt.</p>	L_

**Prompts displayed by the ctest utility (continued)**

Argument	Description	Default
File size code	Size of the file to use for write and read operations. Valid values are: <ul style="list-style-type: none"><li>■ 1 – 1KB</li><li>■ 2 – 500KB</li><li>■ 3 – 10MB</li><li>■ 4 – 2GB</li></ul> <b>Note:</b> Specify a value of 4 (2GB file size) only when you use ctest to test a magnetic tape drive.	3
Use EDC	Specifies whether error detection coding is enabled. Valid values are: <ul style="list-style-type: none"><li>■ N – Error detection coding is disabled</li><li>■ Y – Error detection coding is enabled</li></ul>	N
Write and read	Specifies whether you want to both write and read the test file from the specified device. Valid values are: <ul style="list-style-type: none"><li>■ Y – Perform both a write and read operation</li><li>■ N – Perform a read operation only</li></ul> <b>Note:</b> You should always run ctest in write and read mode. However, if you specify a value of N for this argument, you must have previously written a test file using the ctestsu utility (see	Y

**Prompts displayed by the ctest utility (continued)**

Argument	Description	Default
	Chapter 14) because ctest doesn't create read-only files. Online help is available for this prompt.	
Use device(s) currently in maintenance mode?	Specifies whether you want to perform ctest on a device regardless of whether it is already in maintenance mode. (This prompt appears only when you test a specific drive or all drives for a device.) Valid values are: <ul style="list-style-type: none"> <li>■ Y – Drives that were already in maintenance mode at the time you executed ctest are considered available for testing, unless they are down or allocated (reserved for use by another program or user).</li> <li>■ N – Drives that were already in maintenance mode at the time you executed ctest are considered unavailable, as though they were in a down state.</li> </ul> Online help is available for this prompt.	Y

## Possible error conditions

The following error conditions may result when you use the ctest utility.

### Volume isn't in selected library

The ctest utility produces an error message similar to the following if the ctest file you attempt to read resides on a volume that is in the wrong library (*xxx* is the actual library identifier):

Volume is in *xxx*, which is the wrong device.



Use the StorHouse Command Language MOVE VOLUME command to move the volume to the correct library.

No error message is displayed if the volume is on the shelf; the ctest utility waits for the operator to load the volume before proceeding.

## Unable to create the test file

If ctest is unable to write a test file to the specified drive because a volume set extension is required but the value specified for the /LIMIT attribute for the volume set is too small, ctest produces an error message similar to the following:

```
$$$$$UNABLE TO EXTEND VSET$$$$$
```

Are volumes writelocked???

To correct this problem, do the following:

- Ensure that the most recently used volume is not disabled or writelocked (use the /UNWRITELOCK and/or /ENABLE commands).
- Increase the value for the /LIMIT attribute.

## Write and read results aren't identical

When ctest compares the results of the read operation with those of the write operation, it produces an error message similar to the following if the data patterns are not identical:

```
$$$compare error$$$ recnr=n
```

where *n* is the record number in the SEQUENTIAL file that produced the error.

This error message, which rarely occurs, indicates that there is a serious hardware problem that FileTek hardware support representatives must fix.

## Sample output

This section provides three samples of the output that results when you execute ctest with various values.

Note that, when you accept the default value for a prompt, no text appears in the output following the colon in the prompt.

### Sample 1

The sample output shown below is displayed when you execute the ctest utility with the following values:

- One test iteration
- Device F\_
- 10MB file size
- No EDC
- Write and read operations

```
STH Confidence Test v4.3
```

```
Type ? at a prompt for available help with that prompt
```

```
Press just the ENTER key to use the <default> value
```

```
Enter number of test iterations (1 - 10000) <1>:
```

```
Enter drive to test (LxxDyy, Lxx*, Lxx_, L*, L_, F_) <L_>: f_
```

```
Enter file size code (1=1kb, 2=500kb, 3=10mb, 4=2gb) <3>:
```

```
Use EDC ? <N>:
```

```
Write and read? (Y N) <Y>:
```

```
Checking if StorHouse is ready...
Executing the test...

...begin writing file X10MB_WRT
...begin reading file X10MB_WRT

STH Confidence Test Completed
```

## Sample 2

The sample output shown below is displayed when you execute the ctest utility with the following values:

- One test iteration
- Device L00D01 (in maintenance mode prior to the execution of ctest)
- 1KB file size
- Use EDC
- Write and read operations
- Do not use device(s) currently in maintenance mode

```
STH Confidence Test v4.3
Type ? at a prompt for available help with that prompt
Press just the ENTER key to use the <default> value

Enter number of test iterations (1 - 10000) <1>:

Enter drive to test (LxxDyy, Lxx*, Lxx_, L*, L_, F_) <L_>: 100d01

Enter file size code (1=1kb, 2=500kb, 3=10mb, 4=2gb) <3>: 1

Use EDC ? <N>: y

Write and read? (Y N) <Y>:

Use device(s) currently in maintenance mode? (Y N) <Y>: n
```

```
Checking if StorHouse is ready...
```

```
Specified device(s) not available
```

## Sample 3

The sample output shown below is displayed when you execute the ctest utility with the following values:

- One test iteration
- Device L00\* (in a down state prior to the execution of ctest; contains two drives)
- 1KB file size
- No EDC
- Write and read operations
- Do not use device(s) currently in maintenance mode

```
STH Confidence Test v4.3
```

```
Type ? at a prompt for available help with that prompt  
Press just the ENTER key to use the <default> value
```

```
Enter number of test iterations (1 - 10000) <1>:
```

```
Enter drive to test (LxxDyy, Lxx*, Lxx_, L*, L_, F_) <L_>: 100*
```

```
Enter file size code (1=1kb, 2=500kb, 3=10mb, 4=2gb) <3>: 1
```

```
Use EDC ? <N>:
```

```
Write and read? (Y N) <Y>:
```

```
Use device(s) currently in maintenance mode? (Y N) <Y>: n
```

```
Checking if StorHouse is ready...
```

```
Waiting for L00 to come up...
```

```
Executing the test...
```

```
...begin writing file X1KB_WRT
```

```
...begin reading file X1KB_WRT
...begin writing file X1KB_WRT
...begin reading file X1KB_WRT
```

```
STH Confidence Test Completed
```



## **ctestsu utility**

The Confidence Test Setup (ctestsu) utility creates read-only test files that can be read by the ctest utility (refer to Chapter 13 for details on that utility) following the installation of a new StorHouse system. Run this utility once for each StorHouse SERVICE account (SERVICE, SERVICE1, and so on) defined for the StorHouse system you want to test.

### **How ctestsu works**

The ctestsu utility runs in interactive mode. When you execute the utility, it displays a prompt requesting the StorHouse account ID associated with the library to which you want to write the test files. After you specify the account ID and press **Enter** (↵), ctestsu connects to the StorHouse/SM software. Then it writes a set of StorHouse SEQUENTIAL files of various sizes into the default volume set associated with the account ID you specified. For each file size, the utility writes two files, one with error detection coding (EDC) enabled and one without. Finally, ctestsu disconnects from the StorHouse software.

### **Files created by ctestsu**

The ctestsu utility generates a set of StorHouse SEQUENTIAL files. The name of each file is determined by the size of the file and whether ctestsu enabled error detection coding for that particular file.

The following table lists and describes the StorHouse files created by the ctestsu utility.

This file...	Is a...
X1KB	1KB file
X1KB_EDC	1KB file with EDC enabled
X500KB	500KB file
X500KB_EDC	500KB file with EDC enabled
X10MB	10MB file
X10MB_EDC	10MB file with EDC enabled

## Where ctestsu writes data

When the ctestsu utility writes data to StorHouse, it uses the default file set and volume set for the StorHouse account you specify as the destination for the data write operation. The following table lists the StorHouse accounts and the corresponding default volume set and file set that is normally defined for each. The hexadecimal value *n* in the table corresponds to the ID of the library you are testing when the library is other than L00.

Account	Library	Default VSET	Default FSET
SERVICE	L00	SYSTEM	SERVICE
SERVICE <sub>n</sub>	L0 <sub>n</sub>	SYSTEM <sub>n</sub>	SERVICE

You can view the default values for an account by logging into StorHouse using the SYSTEM account and entering the following command at the StorHouse command prompt (?):

```
? SHOW ACCOUNT {account} /FULL
```



## Format

ctestsu

Argument	Description
Account ID	ID of the StorHouse SERVICE account associated with the library you want to test (for example, SERVICE for library L00 and SERVICE1 for library L01). The ctestsu utility writes a set of files to the default volume set for the account (for example, SYSTEM for library L00 and SYSTEM1 for library L01).

## Sample output

The following sample output is displayed when you execute the ctestsu command using account ID SERVICE1.

```
Alpha2.8> ctestsu
Account id:service1
connecting to SM.

Welcome to the FileTek StorHouse System - Release 5.1  UID:11
  StorHouse/RM Release 2.0
  Last signed on:13-AUG-1999:16:04:54
  Enter HELP for help
Starting to write CTEST file X1KB
close file.
Starting to write CTEST file X1KB_EDC
close file.
Starting to write CTEST file X500KB
close file.
Starting to write CTEST file X500KB_EDC
close file.
Starting to write CTEST file X10MB
close file.
```

```
Starting to write CTEST file X10MB_EDC
close file.
disconnect.
```

```
Signed off 24-AUG-1999:14:48:49
```

## cxclean utility

The Clean (cxclean) utility cleans up Common Support (C) facility shared system resources. Specifically, cxclean removes the following resources, which are identified by files in the current \$SMD\_GENERAL directory:

- Shared memory
- Mailboxes
- Locks

In addition, cxclean removes the temporary files created to manage the system resources.

**Warning:** Don't execute the cxclean utility when the StorHouse software is running. Execute it only after all StorHouse processes have terminated.

## Format

You can execute cxclean interactively or by using a shell script (for example, the stopsm utility).

cxclean [-v]

Option	Description
-v	Identifies the system-specific constructs that are being removed.



## dbdown utility

The Database Down (dbdown) utility takes a StorHouse database offline manually and force it into a “down” (offline) state.

Use this utility to prevent a StorHouse engine from connecting to a database to which you need to make manual repairs, and to prevent users from accessing the database until the repairs are complete. StorHouse displays an error message, “error(-20238):”, when a program or a user attempts to access a database that’s offline (for example, using ESQL or ISQL).

### Format

```
dbdown {-d database_name | -h}
```

Option/Argument	Description
-d <i>database_name</i>	Identifies the StorHouse database that you want to take offline and put into a “down” state.
-h	Displays online help for the dbdown utility.

The utility prompts you to confirm that you want to shut down the database. Type Y or y to continue, or press any other key (except **Enter**) to exit the dbdown utility without taking the database offline.

## Examples

This section presents two examples of how to use the dbdown utility.

### Example 1

The following command takes database STEPH1 offline:

```
dbdown -d STEPH1
```

### Example 2

The following command displays online help for the dbdown utility:

```
dbdown -h
```

## Sample output

The following sample output results when you execute the dbdown utility for a database named STEPH1 and confirm that you want to take the database offline.

```
Database Shutdown  
Copyright FileTek 1999
```

```
** shutdown database: STEPH1  
Are you sure? (Y or y) to continue: y
```

The StorHouse operating system (UNIX) prompt returns after dbdown completes.

## dbup utility

The Database Up (dbup) utility brings a downed (offline) StorHouse database back “up” (online). A database may be in an offline state due to either of the following reasons:

- The access manager (EAM) placed the database in an offline state when it detected that an automatic metadata recovery process failed, which potentially could leave the database in an inconsistent state.
- You took the database offline manually using the dbdown utility to repair the metadata.

A StorHouse engine can't connect to a database that's in an offline state. Therefore, after you make the necessary repairs to the metadata, you must run the dbup utility to return the database to an online state.

## Forcing a database online

When the state of a database changes to offline as a result of either of the actions described above, StorHouse/RM adds the database name to a list of downed databases. In addition, it creates an empty file named *database\_name*.LCK (where *database\_name* is the name of the downed database) in the database directory. Together, these actions serve to indicate that a given database is in an offline state.


When you run the dbup utility, it checks the database indicators described above to ensure that they are consistent for the specified database. If so, it removes the

.LCK file from the database directory and resets the database to an online state. If, however, the database indicators become inconsistent (for example, the .LCK file is somehow deleted before you run dbup), the dbup utility is unable to reset the database to an online state unless you specify the force option when you run the utility. This option specifies that you want to bring up the database regardless of the fact that the indicators are inconsistent.

## Format

dbup {-d *database\_name* | -h} [-f]

Option/Argument	Description
-d <i>database_name</i>	Identifies the StorHouse database that you want to reset to an online state.
-h	Displays online help for the dbup utility.
-f	Forces the specified database online even when the database indicators are in an inconsistent state.

The utility prompts you to confirm that you want to bring up the database. Type Y or y to continue, or press any other key (except ) to exit the dbup utility without bringing up the database.

## Examples

This section presents two examples of how to use the dbup utility.

### Example 1

The following command puts database STEPH1 online:



```
dbup -d STEPH1
```

## Example 2

The following command forces database STEPH1 online regardless of the fact that the database indicators are in an inconsistent state:

```
dbup -d STEPH1 -f
```

## Sample output

The following sample output results when you execute the dbup utility for a database named STEPH1 and confirm that you want to put the database online.

```
Database StartUp  
Copyright FileTek 1999
```

```
** StartUp database: STEPH1  
Are you sure? (Y or y) to continue: y
```

The StorHouse operating system (UNIX) prompt returns after dbup completes.



## dc\_maint utility

The Direct Connect (dc\_maint) utility enables you to verify StorHouse connectivity to a mainframe host by checking the status of the channels that link the StorHouse controller to a customer's mainframe system. You can also use dc\_maint to perform various control functions such as downloading firmware to interface boards, bringing boards online or offline, and running diagnostic tests. The utility provides commands to display information about specific boards or all boards in the system.

### Format

```
dc_maint {config | dchan [cont] | diag [cont] [count] [test_num] | dump [cont] |  
dw [cont] | forcech [cont] | getchan [cont] | lb [cont] [count] | lsscp [cont]  
[psscp] [/lsscp] | off [cont] | offer | on [cont] | psscp [cont] [psscp] | rev [cont]}
```

Command/Argument	Description
config	<p>Displays configuration information about each ESCON board in the system, including:</p> <ul style="list-style-type: none"><li>■ A line for each controller, which specifies the controller number, current state, and the link-level retry count (provided it is non-zero) if the board is online.</li><li>■ A line for each configured PSSCP, which specifies the PSSCP number, state, base address and number of units, and local and remote link addresses. This line also provides LSSCP summary information, including the number of active LSSCPs associated with the current PSSCP.</li></ul>

Command/Argument	Description
dchan	Obtains traces from the ESCON board and displays them to standard output.  <b>Note:</b> If you have not yet downloaded the firmware and configuration information for the board, you must do so before issuing this command.
[conf]	Controller or board number. If you do not specify a controller number, the default value of 0 is used.
diag	Performs various diagnostic tests on the specified ESCON board.  <b>Note:</b> The board must be in a downloaded state but not online when you run diagnostic tests.
[conf]	Controller or board number. If you do not specify a controller number, the default value of 0 is used.
[count]	Number of times you want to run the diagnostic test.
[test_num]	Number that identifies the test you want to run. The dc_maint utility accepts hexadecimal values by default; therefore, you can omit 0x when you specify the test number. Valid values are: <ul style="list-style-type: none"> <li>■ 01 – All local non-interactive tests</li> <li>■ 02 – NMI test</li> <li>■ 04 – Slow count test</li> <li>■ 08 – 6900 timer test</li> <li>■ 10 – DMA to channel test</li> <li>■ 20 – DMA to UNIX test</li> <li>■ 40 – Interrupt to UNIX test</li> <li>■ 80 – Increment polling duration</li> </ul> <p>If you do not specify a particular diagnostic test, the utility runs all diagnostic tests.</p> <p>To run a group of specific diagnostic tests, specify the sum of the test numbers (in hex) as the value for this argument. For example, to run a group of tests consisting of test numbers 1, 8, and 40, you would specify 49.</p>

Command/Argument	Description
	<p><b>Note:</b> You should use the bmon_ctcs utility to monitor the serial port while diagnostics are running because, in the event of a diagnostic failure, detailed information regarding the failure might be displayed.</p>
dump [conf]	<p>Obtains UNIX device driver traces and displays them to standard output.</p> <p><b>Note:</b> You must download the firmware and configuration information for the board before issuing this command.</p>
dw [conf]	<p>Downloads microcode (firmware) to the specified board.</p> <p><b>Note:</b> The board must be in an offline state before you issue this command.</p>
forcech [conf]	<p>Forces a driver/board trace.</p>
getchan [conf]	<p>Obtains any channel traces that were automatically collected by the driver or collected using the forcech command, and writes them to the file ./getchan.N, where N is the controller number. New data is appended to this file to prevent previously collected trace data from being overwritten.</p>
lb	<p>Runs a local loopback test where signals are looped from a test center through a data set or loopback switch and back to the test center for measurement.</p> <p><b>Note:</b> The board must be in a downloaded state and the local loopback connector must be in place before you issue this command.</p>
[conf]	<p>Controller number. If you do not specify a controller number, the default value of 0 is used.</p>
[count]	<p>Number of times you want to run the loopback test.</p>
lsscp	<p>Displays information about an LSSCP. An LSSCP represents an active connection between the ctcs driver (which supports a Polaris model 8900 Sbus board) and the IBM host subsystem.</p> <p>When you issue this command with no arguments, the default value of 0 is used for each of the arguments (conf, psscp, and lsscp).</p>

Command/Argument	Description
[ <i>cont</i> ]	Controller or board number. If you do not specify a controller number, the default value of 0 is used.
[ <i>psscp</i> ]	ID of the physical connection (image) associated with the LSSCP for which you want information.
[ <i>lsscp</i> ]	ID of the logical connection for which you want information. This number is assigned by the ctcs driver.
off [ <i>cont</i> ]	<p>Takes a specific board offline from the channel. If you do not specify a controller number, the default value of 0 is used.</p> <p><b>Note:</b> The board must be in an online state before you issue this command. If you are running the dc_maint utility interactively, the utility prompts you to confirm your request prior to executing the command.</p>
offer	Displays information about outstanding offers from the StorHouse software to the host to establish a link.
on [ <i>cont</i> ]	<p>Initiates online processing for the specified board. This command completes immediately; however, the board does not come online until the channel is up. Individual PSSCPs (images) do not come online until paths are established to the IBM mainframe.</p>
psscp [ <i>cont</i> ] [ <i>psscp</i> ]	<p>Displays detailed information about a PSSCP (image); that is, the physical connection to the StorHouse subsystem.</p> <p>When you issue this command with no arguments, it displays a list of all units that are still available but have not yet been allocated to the LSSCP.</p>
rev [ <i>cont</i> ]	<p>Displays the current microcode (firmware) revisions.</p> <p><b>Note:</b> If you have not yet downloaded the firmware and configuration information for the board, you must do so before issuing this command.</p>

## Examples

This section presents two examples of how to run the dc\_maint utility.

### Example 1

The following command displays the status of the channels and enables you to identify the controller/interfaces to be stopped:

```
dc_maint config
```

### Example 2

The following command takes controller number 1 offline from the channel:

```
dc_maint off 1
```

Sample output

The following dc\_maint command:

dc\_maint config

results in the sample output below:

Note that the sum of active devices (1) and available devices (3c) for this PSSCP is less than the count (40). That's because one bidirectional device is reserved for the PSSCP. In addition, each LSSCP requires two unidirectional devices: one for reading from MVS and one for writing to MVS.

C/PSSCP	State	(Rtry)	Base/Cnt	Lcl_LA	Rmt_LA	LSSCPs	Active	Avail	Reads	Writes
C 0	ONLINE									
P 0	online		00/40	87 00	8F 00	0	1	3C	2547	2546
P 1	online		00/40	87 01	85 00	0	3	3A	1C442	1C445
P 2	going on		00/40							
C 1	ONLINE									
P 3	online		00/40	86 03	83 00	0	1	3C	1C598	1C597
P 4	online		00/40	86 04	84 00	0	B	32	17BF	17B6
P 5	online		00/40	86 05	85 00	0	1	3C	1063D	1063C



## Interpreting the output

The following table identifies the type of data under each column in the sample output. Note that the numbers in the output are in hexadecimal format.

Column	Description
C/PSSCP	Controller or board number and number of each PSSCP configured for a controller.
State	Current state of each controller or PSSCP. Possible values are: <ul style="list-style-type: none"> <li>■ online</li> <li>■ offline</li> <li>■ going online</li> <li>■ downloaded</li> <li>■ not downloaded yet</li> <li>■ halted</li> </ul>
(Rtry)	Number of errors and successive attempts to establish a connection. This column contains a value only when the board is in an online state.
Base/Cnt	Base address and number (count) of units configured for the PSSCP. Configuration information for a board is maintained in /filetek/etc/polaris/parm/ <i>N</i> .txt, where <i>N</i> is the board number.
Lcl_LA	Local link address on the ESCON director for the StorHouse system. The link address consists of two parts: a port number and an image number. The PSSCP number corresponds to the image number.
Rmt_LA	Remote link address on the ESCON director for the host system.
LSSCPs	Number that corresponds to each active LSSCP. For example, if LSSCP number 0 was active for a given PSSCP, this column would display 0.
Active	Number of units that are currently active (allocated to the LSSCP) for a PSSCP.
Avail	Number of units that are currently available for a PSSCP.

Column	Description
Reads	Number of read operations performed since the unit was allocated. This value represents the sum of all input operations performed on all active units for a PSSCP.
Writes	Number of write operations performed since the unit was allocated. This value represents the sum of all output operations performed on all active units for a PSSCP.

## doit utility

The `doit` utility provides an alternative to writing a C program to test StorHouse API functions or Command Language commands. The utility provides a set of text commands that execute StorHouse Callable Interface (LSMxxx) functions for the C language. Use this utility to develop your own test scripts, either to conduct a general test of a StorHouse system, or to set up specific tests after performing maintenance procedures.

You can execute the `doit` utility in one of two ways, either interactively or in batch mode:

- To execute the `doit` utility interactively, type `doit` at the StorHouse operating system (UNIX) command prompt. The utility displays the DOIT: prompt. Enter the desired commands and associated parameters, one line at a time, and press Enter ↵ after each line.
- To execute the `doit` utility in batch mode, create an executable file (using any UNIX text editor) that contains the commands and parameters associated with the functions you want to test (the commands and parameters are described in the “Format” section of this chapter). Sample `doit` procedures are provided at the end of this chapter.

The `doit` utility parses `doit` commands and translates the text commands into C language function calls. The utility exits with a return code of 0 if there are no errors, or a return code of 1 if the program encounters errors.

See the *Generic Callable Interface Programmer's Guide*, publication number 900046, for more information about the StorHouse Callable Interface functions

including a complete description of the statement parameters and possible return codes.

## Rules for command parsing

Observe the following rules when entering doit commands:

- Commands must begin in the first position of each line in the file and must be specified as shown in the table beginning on page 19-3.
- Parameters must be specified as shown in the table; however, they are not case sensitive:
  - No white space can appear between a parameter and its associated value.
  - One or more spaces must appear between parameters.
  - Numeric parameters use the decimal number system by default. To use the hexadecimal number system, convert the number to its hexadecimal form, then add the prefix *0x* (for example, the hexadecimal form of 31 is 0x1F).
- An exclamation point (!) or pound sign (#) in the first position of a line identifies a comment.

## Format

doit

The “–” symbol in the Default column of the following table indicates that values for these arguments default to a null or all-blanks string (spaces) and are supplied by the StorHouse account’s default values. If the account defaults are also blank, the values are copied from the model file (if applicable). If the account defaults are blank and no model file exists, you must specify values for these arguments; otherwise, an error results.

Command/ Argument	Description	Default
callcreate	Executes an LSMCO (Create Open) function.	
[group=]	Name of the file access group for the StorHouse file you are creating.	–
file=	Name of the StorHouse file you are creating. If the file name contains lowercase letters, you can enclose the name in quotation marks to prevent doit from converting all letters to uppercase. If a KEYED file is required, you must specify a model file with an 8-byte key called <i>keyname</i> . For example:  KEY KEYNAME 1:8	
size=	Number of bytes in the file you are creating, specified in 1000-byte units. The value for the size argument must not exceed 2000000 (2 million). Do not use commas in the specification.	
[fset=]	Name of the file set for the StorHouse file you are creating.	–
[vset=]	Name of the volume set for the StorHouse file you are creating.	–
[model=]	Name of the StorHouse model file if a KEYED file is required.	
[edc=]	Specifies whether error detection coding (EDC) is enabled. Valid values are:  ■ -1 – No ■ 0 – Use default value ■ 1 – Yes	-1

Command/ Argument	Description	Default
[vtf=]	Vulnerability Time Factor (VTF), which controls when StorHouse copies a file from the performance buffer to its resident file set. Valid values are: <ul style="list-style-type: none"> <li>■ 0 – Use default value</li> <li>■ 2 – Next</li> <li>■ 4 – Direct</li> </ul>	2
[limit=]	File version limit value.	1
[ascii=]	Data storage format. If positive, the data is stored as ASCII characters.	0
[checkpoint=]	Caller-supplied value that indicates the checkpoint number where file processing should be restarted. A value of 0 indicates normal (non-restart) operations.	0
[cache=]	Number of sequential records that VRAM caches for an LSMRS, LSMRR, or an LSMRK function for this file when it is opened with an access mode of READ or UPDATE and an access method including RECORD and/or KEYED.	0
change	Executes an LSMCH (Change Record) function. This command must be preceded by a doit readkey, readrec, rdnxtkey, or readseq command.  No data modification is performed. The data record last read is rewritten as is.	
checkpoint	Executes an LSMCP (Checkpoint) function. The checkpoint number is printed and can be subsequently used with the doit openvram or callcreate commands.	
close	Executes an LSMCLO (Close) function.	
[abort=]	Integer that indicates whether a normal close takes place or a close is issued to abort the data transfer operation. Valid values are: <ul style="list-style-type: none"> <li>■ A zero value indicates a normal close.</li> <li>■ A non-zero value indicates that StorHouse should abort the data transfer operation.</li> </ul>	0

Command/ Argument	Description	Default
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the close operation.	
command	Executes an LSMSCI (StorHouse Command Interface) function. If a CREATE FILE command is used that prompts for key data, the responses will be provided to build an 8-byte key.	
connect	Executes an LSMCON (Connect) function to establish a command link (StorHouse session).	
acct=	Character string containing the StorHouse account identification code (AID) for the session.	
[pass=]	Character string containing the password associated with the StorHouse account.	—
[smid=]	Character string that identifies the specific StorHouse system to be accessed. If null, the default or only StorHouse system is accessed.	—
[ssid=]	Character string with a maximum length of 4. This argument is not used and should be null.	—
dump	Produces a formatted hexadecimal report for the last record read.	
delete	Executes an LSMDEL (Delete Record) function. This command must be preceded by a doit readkey, readrec, rdnxtkey, or readseq command.	
disconnect	Executes an LSMDIS (Disconnect) function, which concludes a StorHouse session by terminating the connection that was established by LSMCON.	

Command/ Argument	Description	Default
error	Controls error processing and message generation.	
[stop=]	Sets or resets the stop on error flag. The default value of 1 (yes) for batch mode specifies that the doit utility should stop processing if an error occurs. The default value of 0 (no) for interactive mode specifies the converse.	1-batch/0- interactive
[xlat=]	Translates error codes into text messages. The default value of 0 (no) for batch mode specifies that the doit utility should not include a text description of the error code when an error occurs. The default value of 1 (yes) for interactive mode specifies the converse.	0-batch/1- interactive
exit	Exits the doit program.	
openseq	Executes an LSMOS (Open Sequential) function.	
[group=]	Name of the file access group for the StorHouse file you are opening.	—
file=	Name of the StorHouse file you are opening. If the file name contains lowercase letters, you can enclose the name in quotation marks to prevent doit from converting all letters to uppercase.	
mode=	Character string that identifies the file reference mode. Valid values are READ and WRITE. These values may be specified in upper or lower case.	
size=	Number of bytes in the file you are opening, specified in 1-byte units. The value for the size argument must not exceed 2000000000 (2 billion). Do not use commas in the specification.	
[fset=]	Name of the file set for the StorHouse file you are opening.	—
[vset=]	Name of the volume set for the StorHouse file you are opening.	—



Command/ Argument	Description	Default
[edc=]	Specifies whether error detection coding (EDC) is enabled. Valid values are: <ul style="list-style-type: none"> <li>■ -1 – No</li> <li>■ 0 – Use default value</li> <li>■ 1 – Yes</li> </ul>	-1
[vtf=]	Vulnerability Time Factor (VTF), which controls when StorHouse copies a file from the performance buffer to its resident file set. Valid values are: <ul style="list-style-type: none"> <li>■ 0 – Use default value</li> <li>■ 2 – Next</li> <li>■ 4 – Direct</li> </ul>	2
[version=]	Long integer containing the file's version number. This argument applies only to READ operations. Zero is the most current version.	0
[maxrec=]	Maximum length, in bytes, for any record in the file.	0
[ascii=]	Data storage format. If positive, the data is stored as ASCII characters.	0
openvram	Executes an LSMOV (Open VRAM) function. The CREATE FILE command must have been executed prior to initially writing data to a VRAM file.	
[group=]	Name of the file access group of the file you are opening.	–
file=	Name of the file you are opening. If the file name contains lowercase letters, you can enclose the name in quotation marks to prevent doit from converting all letters to uppercase.	
mode=	Character string that identifies the file reference mode. The acceptable values are READ, APPEND, and UPDATE. These values may be specified in upper or lower case. <p><b>Note:</b> For mode=APPEND, the fset and vset for the file are assigned during CREATE FILE processing.</p>	

Command/ Argument	Description	Default
method=	Character string that identifies the type of file processing. The acceptable values are SEQUENTIAL, KEYED, RECORD, ALL, or a combination of any two or three of SEQUENTIAL, KEYED, and RECORD, separated by commas. These values may be specified in upper or lower case. This argument is ignored when mode is set to APPEND.	
[revision=]	Integer set by the user to indicate the file version's revision number. Valid values are: <ul style="list-style-type: none"> <li>■ 0 – default (most current) revision</li> <li>■ Positive – indicates an absolute revision number</li> <li>■ Negative – indicates a relative revision number</li> </ul>	0
[version=]	Long integer containing the file's version number. This argument applies only to READ operations. Zero is the most current version.	0
[checkpoint=]	Caller-supplied value that indicates the checkpoint number where file processing should be restarted. A value of 0 indicates normal (non-restart) operations.	0
rdnxtkey	Executes an LSMRNK (Read Next Key) function. This command must be preceded by a doit readkey command or a successful rdnxtkey command.	
bufsiz=	Size, in bytes, of the buffer to be used for I/O operations. The value you specify must be greater than or equal to the actual record size to avoid record truncation.	
[compare=]	Specifies that the command should validate data for each record read if you specify a value other than 0 for this argument.	0
[repeat=]	Number that allows you to perform multiple I/O operations with a single command. Key and recnum parameters are incremented for each repeat.	0

Command/ Argument	Description	Default
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the rdnextkey operation. If you specify multiple rdnextkey operations (that is, you specify a value other than 1 for the rdnextkey repeat argument), doit compares the value you specify for the expect argument against the results of the last rdnextkey operation only.	
read	Executes an LSMR (Read) function (SEQUENTIAL files only).	
bufsiz=	Size, in bytes, of the buffer to be used for I/O operations. The value you specify must be greater than or equal to the actual record size to avoid record truncation.	
[compare=]	Specifies that the command should validate data for each record read if you specify a value other than 0 for this argument.	0
[repeat=]	Number that allows you to perform multiple I/O operations with a single command. The recnum parameter is incremented for each repeat.	0
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the read operation. If you specify multiple read operations (that is, you specify a value other than 1 for the read repeat argument), doit compares the value you specify for the expect argument against the results of the last read operation only.	
readkey	Executes an LSMRK (Read Keyed) function (VRAM files only).	
bufsiz=	Size, in bytes, of the buffer to be used for I/O operations. The value you specify must be greater than or equal to the actual record size to avoid record truncation.	
[compare=]	Specifies that the command should validate data for each record read if you specify a value other than 0 for this argument.	0

Command/ Argument	Description	Default
[repeat=]	Number that allows you to perform multiple I/O operations with a single command. Key and recnum parameters are incremented for each repeat.	0
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the readkey operation. If you specify multiple readkey operations (that is, you specify a value other than 1 for the readkey repeat argument), doit compares the value you specify for the expect argument against the results of the last readkey operation only.	
key=	Pointer to an area containing the value of the key used for the record search.	
[keyname=]	Character string containing the name of the key used to locate the record.	KEYNAME
[keylen=]	Long integer that provides the length of the value in the area indicated by the key parameter.	8
readrec	Executes an LSMRR (Read Record) function (VRAM files only).	
bufsiz=	Size, in bytes, of the buffer to be used for I/O operations. The value you specify must be greater than or equal to the actual record size to avoid record truncation.	
[compare=]	Specifies that the command should validate data for each record read if you specify a value other than 0 for this argument.	0
[repeat=]	Number that allows you to perform multiple I/O operations with a single command. Key and recnum parameters are incremented for each repeat.	0
recnum=	Record number of the record to be read.	

Command/ Argument	Description	Default
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the readrec operation. If you specify multiple readrec operations (that is, you specify a value other than 1 for the readrec repeat argument), doit compares the value you specify for the expect argument against the results of the last readrec operation only.	
readseq	Executes an LSMRS (Read Sequential) function (VRAM files only).	
bufsiz=	Size, in bytes, of the buffer to be used for I/O operations. The value you specify must be greater than or equal to the actual record size to avoid record truncation.	
[compare=]	Specifies that the command should validate data for each record read if you specify a value other than 0 for this argument.	0
[repeat=]	Number that allows you to perform multiple I/O operations with a single command. The recnum parameter is incremented for each repeat.	0
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the readseq operation. If you specify multiple readseq operations (that is, you specify a value other than 1 for the readseq repeat argument), doit compares the value you specify for the expect argument against the results of the last readseq operation only.	
shell	Passes the remainder of the command to the Bourne shell.	
timestamp	Outputs a timestamp and displays the elapsed time since the last timestamp. If a data transfer process is active, this command also displays the transfer rate, which is useful for performance measurements.	

Command/ Argument	Description	Default
write	Executes an LSMW (Write) function (VRAM and SEQUENTIAL files).	
bufsiz=	Size, in bytes, of the buffer to be used for I/O operations. This value represents the size used for each record written by the write command.	
[repeat=]	Number that allows you to perform multiple write operations with a single command. Key or recnum parameters are incremented for each repeat, as is data.	0
[compare=]	Specifies that the command should generate data for writing if you specify a value other than 0 for this argument.	0
[expect=]	Return code or "EOF" string that specifies the condition that should result at the end of the write operation. If you specify multiple write operations (that is, you specify a value other than 1 for the write repeat argument), doit compares the value you specify for the expect argument against the results of the last write operation only.	

---

## Sample doit procedure for VRAM files

The following sample doit procedure tests the most important StorHouse API functions for VRAM files. The sample assumes that a model file named `doitmodelfile` exists.

```
doit<<EOF >my.doit
timestamp
connect acct=service pass=service
timestamp
callcreate file=justdoit2 size=1000 cache=100 model=doitmodelfile
timestamp
write bufsiz=1000 repeat=100 compare=1
checkpoint
write bufsiz=500 repeat=200 compare=1
checkpoint
write bufsiz=200 repeat=500 compare=1
close
openvram file=justdoit2 mode=update method=keyed
readkey bufsiz=1000 compare=1 key=100
change
readkey bufsiz=1000 compare=1 key=200
delete
close
openvram file=justdoit2 mode=read method=all
readkey bufsiz=1000 compare=1 key=250
rdnxtkey bufsiz=1000 compare=1 repeat=10
readseq bufsiz=1000 compare=1 repeat=10
readrec bufsiz=1000 compare=1 recnum=500 repeat=10
readrec bufsiz=1000 compare=1 recnum=500
close
openvram file=justdoit2 mode=read method=sequential
readseq bufsiz=1000 compare=1 repeat=801 expect=EOF
close
disconnect
exit
EOF
```

## Sample doit procedure for non-VRAM files

The following sample doit procedure tests the most important StorHouse API functions for non-VRAM files.

```
doit<<EOF >my.doit
connect acct=system pass=system
openseq file=justdoit mode=write size=5000000 ascii=1 maxrec=1000
write bufsiz=1000 repeat=1000 compare=1
write bufsiz=500 repeat=2000 compare=1
write bufsiz=200 repeat=5000 compare=1
close
openseq file=justdoit mode=read
read bufsiz=5000 compare=1
close
disconnect
exit
EOF
```



## doit.basic utility

The doit.basic utility performs a basic verification test of the StorHouse VRAM component. Use this utility as an installation validation procedure to ensure that the StorHouse software is installed correctly or after you make any program modifications.

The doit.basic utility invokes the doit utility and provides a set of text commands as input to doit. The doit.basic utility is designed to test a specific set of operations to verify that they are working correctly; therefore, the doit.basic file uses only a subset of the doit commands. As described in Chapter 19, “doit utility,” these commands execute StorHouse Callable Interface (LSM $xxx$ ) functions for the C language.

The doit.basic utility performs the following operations:

Operation	DOIT command	LSM function
Establishes a session with StorHouse.	connect	LSMCON
Creates a new VRAM RECORD file on StorHouse using the default value of 2 (Next) for the Vulnerability Time Factor (VTF). Also establishes a data transfer link for writing information to the file.	callcreate	LSMCO
Transfers the next sequential data record from the caller's buffer to a StorHouse file.	write	LSMW
Terminates the data transfer operation that was initiated by LSMCO.	close	LSMCLO
Opens a VRAM file on StorHouse.	openvram	LSMOV

Operation	DOIT command	LSM function
Requests a specific record (reads data) from the VRAM file. The record is identified by its relative record number. This operation is repeated three times, for three different records.	readrec	LSMRR
Terminates the data transfer operation that was initiated by LSMOV.	close	LSMCLO
Opens a VRAM file on StorHouse.	openvram	LSMOV
Requests the next sequential record from the VRAM file.	readseq	LSMRS
Terminates the data transfer operation that was initiated by LSMOV.	close	LSMCLO
Terminates (disconnects) the StorHouse connection that was established by LSMCON.	disconnect	LSMDIS
Exits the doit program.	exit	

The doit.basic utility writes a known data pattern of approximately 10MB to the performance buffer. It then reads back the entire file, by record and sequentially, to ensure that the file was written correctly. The utility displays messages during processing and indicates expected results.

Refer to Chapter 19, “doit utility”, for a description of the commands and parameters executed by the doit.basic utility. For additional information about the StorHouse Callable Interface functions, see the *Generic Callable Interface Programmer's Guide*, publication number 900046.

## Format

doit.basic

## Sample output

The command:

```
doit.basic
```

produces the following sample output:

```
DOIT: connect acct=service pass=SERVICE
```

```
DOIT: callcreate size=10000 file=basic
```

```
DOIT: write bufsiz=10000 repeat=900 compare=1
```

```
DOIT: close
```

```
DOIT: openvram file=basic mode=read method=record
```

```
DOIT: readrec bufsiz=10000 recnum=300 compare=1
```

```
DOIT: readrec bufsiz=10000 recnum=500 compare=1
```

```
DOIT: readrec bufsiz=10000 recnum=700 compare=1
```

```
DOIT: close
```

```
DOIT: openvram file=basic mode=read method=sequential
```

```
DOIT: readseq bufsiz=10000 repeat=901 compare=1 expect=5650
```

```
DOIT: close
```

```
DOIT: disconnect
```

```
DOIT: exit
```



## dumpulog utility

The dumpulog utility produces a formatted dump of the user log for one or more days. You can use the output of this utility to analyze StorHouse activity.

You can dump user log records for the current date, the prior day's date, a selected date, or a range of dates. You can also specify which record types appear in the output file by using the SELECT parameter. (See the *User Log Format*, publication number 900028, for more information about user log data record types.) If you do not specify record types, information for all record types is dumped by default. If multiple user logs exist for the specified date, the dumpulog utility dumps the appropriate records from all the logs.

In order to dump records for a specific date or range of dates from the user log, the StorHouse NEWLOG /USER command must have been issued subsequent to the specified time period. The NEWLOG command closes the current temporary log file, opens a new one, and writes the closed file to StorHouse user storage. The dumpulog utility then accesses the copy of the log file that resides in user storage. (See the *Command Language Reference Manual*, publication number 900005, for more information on the NEWLOG command.)

To ensure that FileTek customer support representatives can run the dumpulog utility whenever necessary, support personnel typically schedule the NEWLOG command to run daily (shortly before midnight) on every installed StorHouse system.

## Format

```
dumpulog "{today | yesterday | week | month |
first=yyyyjjj},[last=yyyyjjj],[select=(n:n:n)]"
```

You can use either the *yyyyjjj* or *yyyymmdd* format when specifying a date.

**Note:** You can also execute the dumpulog utility in interactive mode, and by using the RUN command from the StorHouse command prompt (?).

Argument	Description
today	Specifies that the user log with the current date is the first log file to process.
yesterday	Specifies that the user log with the prior calendar day's date is the first log file to process.
week	Specifies that the first user log file to process is dated one week (7 calendar days) prior to the current date and the last user log file to process is yesterday's date.
month	Specifies that the first user log file to process is dated one month (30 calendar days) prior to the current date and the last user log file to process is yesterday's date.
first=yyyyjjj or first=yyyymmdd	Calendar year and date of the first user log file to process if other than today, yesterday, week, or month. If you use the Julian date <sup>a</sup> format (yyyyjjj), specify a number from 001 to 365 or 366 following the year.

Argument	Description
last=yyyyjjj or last=yyyymmdd	Calendar year and date of the last user log file to process when used in conjunction with the “first” argument. If you omit this argument, the default is last=first.  If you use the Julian date format, specify a number from 001 to 365 or 366 following the year.
select=(n:n:n)	Specifies the type of user log records to process. You can specify any number of record types. If you omit this argument, the default is all record types listed in the <i>User Log Format</i> , publication number 900028.

- a. The Julian date is a number assigned in sequence to each day of the year, starting with Julian date 001 for January 1 and ending with Julian date 365 (or 366 for a leap year) for December 31.

## Examples

This section presents three examples of how to dump record types from the user log.

### Example 1

The following command dumps all record types from the user log with the prior calendar day's date:

```
dumpulog "yesterday"
```

### Example 2

The following command dumps record types 10, 13, and 20 from all user logs dated January 1, 1999 through March 28, 1999:

```
dumpulog "first=1999001,last=1999087,select=(10:13:20)"
```

### Example 3

The following command dumps record type 9 from the user log that contains data for March 13, 1999:

```
dumpulog "first=1999072,select=(9)"
```



## Sample output

The following dumpulog command:

```
dumpulog "yesterday,select=(10:15:20)"
```

results in the sample output below (the output is not shown in its entirety to conserve space):

DATE-RANGE is: 1999192 thru 1999192

\*\*\*\*\* Data record name = command\_exec

```
system_id=<002102> account_id=<SYSTEM> user_id=<3>  
log_time=<10-JUL-1999:23:59:00> start_time=<10-JUL-1999:23:59:00>  
end_time=<10-JUL-1999:23:59:00> command=<NEWLOG> status=<5633>  
status_message=<%WORLD-S-XWOK, A request was successfully  
completed.>  
severity=<5633> command_id=<703>
```

\*\*\*\*\* Data record name = signoff

```
system_id=<002102> account_id=<SYSTEM> user_id=<3>  
log_time=<10-JUL-1999:23:59:00> start_time=<10-JUL-1999:23:59:00>  
end_time=<10-JUL-1999:23:59:00> num_commands=<2>  
num_sec_attempts=<0>  
max_severity=<5633>
```

\*\*\*\*\* Data record name = signon

```
system_id=<002102> account_id=<SYSTEM> user_id=<3>  
log_time=<10-JUL-1999:23:59:00> start_time=<10-JUL-1999:23:59:00>  
end_time=<10-JUL-1999:23:59:00> status=<5633>  
status_message=<%WORLD-S-XWOK, A request was successfully  
completed.>
```

\*\*\*\*\* Data record name = command\_exec

system\_id=<002102> account\_id=<SYSTEM> user\_id=<5>  
log\_time=<10-JUL-1999:23:59:01> start\_time=<10-JUL-1999:23:59:01>  
end\_time=<10-JUL-1999:23:59:01> command=<mon /all /int=0>  
status=<5633>  
status\_message=<%WORLD-S-XWOK, A request was successfully  
completed.>  
severity=<5633> command\_id=<1301>

\*\*\*\*\* Data record name = file\_close

system\_id=<002102> account\_id=<SYSTEM> user\_id=<3>  
log\_time=<10-JUL-1999:23:59:02> start\_time=<10-JUL-1999:23:59:00>  
end\_time=<10-JUL-1999:23:59:02> linked=<10-JUL-1999:23:59:00>  
close\_received=<10-JUL-1999:23:59:02> status=<5633>  
status\_message=<%WORLD-S-XWOK, A request was successfully  
completed.>  
net\_link=<> mode=<4> method=<1> file\_sysid=<2102> file\_fno=<1080963>  
filename=<U1999191235900> group=<SERVICE> version=<0> revision=<1>  
max\_revision=<1> organization=<0> volume\_set=<USERLOG>  
file\_set=<USERLOG> num\_records=<0> transferred=<2179783>  
size\_ext=<>  
new\_rev=<1> deletes=<> updates=<> key\_changes=<> bytes=<>  
cache\_hits=<>

\*\*\*\*\* Data record name = library\_info

system\_id=<002102> account\_id=<> user\_id=<0>  
log\_time=<11-JUL-1999:00:00:00> start\_time=<10-JUL-1999:23:45:00>  
end\_time=<11-JUL-1999:00:00:00> src\_dev\_name=<L02> dtype=<>  
soft\_errors=<0> hard\_errors=<0> accessor\_req=<0>

\*\*\*\*\* Data record name = drive\_info

system\_id=<002102> account\_id=<> user\_id=<0>  
log\_time=<11-JUL-1999:00:00:00> start\_time=<10-JUL-1999:23:45:00>  
end\_time=<11-JUL-1999:00:00:00> src\_dev\_name=<L00D00>  
media\_type=<TC>

## Possible error conditions and resolutions

If you execute the dumpulog command and the utility returns the following message with no data:

DATE-RANGE is: 1999101 thru 1999101

the probable cause is one of the following:

- NEWLOG was not executed subsequent to the specified date or range of dates.

Resolution: Sign on to StorHouse and issue the following command at the StorHouse command prompt (?):

NEWLOG /USER

then sign off from StorHouse and execute the dumpulog command again.

- No data records of the type specified exist for the designated time period.

Resolution: No action is required. Specify a different set of record types, if desired.



## follow utility

The follow utility lets you monitor, or *follow*, the progress of various files on a customer's StorHouse system. Use this utility when you are logged on to a customer's StorHouse system remotely (for example, to diagnose a problem) to view operator messages, UNIX system log messages, and other output that might otherwise be unavailable to remote users.

### How follow works

When you dial in to a customer's StorHouse server via the modem, many of the messages that are written to the customer's console, such as ODU device down/up, box-rebuild start/stop, and optical device down/up, are not echoed to your terminal. The follow utility lets you view console messages by capturing them and directing them to standard output (stdout), which is normally defined as your terminal. As lines are added to the files you are following, follow copies them to the terminal.

Optionally, you can run follow in the background so you can perform other tasks at your terminal while follow executes. As it runs, follow displays console messages at your terminal without tying it up.

The follow utility performs a function similar to that of the tail utility with the -f option. However, follow differs from the tail utility in that it exits if its parent process (the UNIX shell) exits. This feature is particularly useful when you are running follow in the background because, when you either log off the StorHouse server operating system or get logged off, follow exits instead of tying up the terminal file (the modem connection to StorHouse). Therefore, the

modem connection is free in the event the customer's StorHouse system needs to generate a Call Home message.

For convenience, the follow utility provides command options for two frequently followed files: /var/adm/messages and the \$SMD\_GENERAL/operator.log. You can also specify different file names using the *filename* arguments. At least one file must be specified.

## Format

You must specify at least one option.

follow {-m -o *filename* [*filename*] [...]}

Option/Argument	Description
-m	Specifies that you want to follow the /var/adm/messages file.
-o	Specifies that you want to follow the \$SMD_GENERAL/operator.log file.
<i>filename</i>	Name of a file other than those specified above that you want to follow.

## Examples

This section presents two examples of how to run the follow utility.

### Example 1

The following command displays all console messages for both the UNIX system log (/var/adm/messages file) and the operator log in the background at your terminal:

```
follow -mo &
```

### Example 2

The following command displays lines of text in the background at your terminal as they are added to a file named my.log:

```
follow my.log &
```

## Sample output

The following sample output results when you issue the follow command with the -o option (the first line of text in the sample exceeds the width of this page so it wraps to the following line):

```
01/21/00-17:09:17 0001 (A) XROLDP LOAD-BLANK L00E00, VOLUMES=1, MEDIA  
TYPE=TDA, REPLY 'C'-CONTINUE 'E'-ERROR
```

```
01/21/00-17:09:22 0001 ® E
```

```
01/21/00-17:09:34 (I) XROINFO, L00D01, DEVICE IS DOWN
```

```
01/21/00-17:09:42 (I) XROINFO, L00D01, DEVICE IS UP
```





## gen\_dev utility

The primary purpose of the `gen_dev` utility is to create a new device configuration file (`bconfig`) or update the existing one. This file contains all device information for a StorHouse system. Device information includes things like device, drive, and media types as well as the number of slots and available free pool volume sets in each library device.

The `gen_dev` utility also creates the library device volume (`bldv`) system file for each library, which maintains library device and volume information for a StorHouse system. The `bldv` file keeps track of information such as which volume resides in which slot of a library device.

The output of the `gen_dev` utility and the way you execute it depend on whether you are installing a new StorHouse system or modifying an existing one.

- When you install a new StorHouse system, you execute the `build_sxm` utility, which calls `gen_dev` for you (refer to Chapter 7 for more information on the `build_sxm` utility). You do not invoke `gen_dev` directly (see page 23-4). The end result is a set of new system files that includes `bconfig` and `bldv`.
- When you install a new library on an existing StorHouse system or make changes to an existing library configuration, you execute `gen_dev` directly (see page 23-9). In this case, `gen_dev` just produces new `bconfig` and `bldv` files.

## Inputs to gen\_dev

The `gen_dev` utility has two generic input files:

- `config_dev`
- `spx_maint.inp`

These files, which reside in the `$BUILD` directory on each customer's StorHouse server, are configuration templates that you edit and tailor to reflect the actual configuration of a customer's StorHouse system.

**config\_dev.** The standard `config_dev` file contains generic configuration information for StorHouse devices. It also contains several sets of commands that are subsequently passed to various UNIX shell scripts that `gen_dev` invokes.

**spx\_maint.inp.** The `spx_maint.inp` file contains generic StorHouse system parameter information. You must customize these parameters for individual StorHouse installations. In addition, you must synchronize the information in `spx_maint.inp` and `config_dev` because some system parameters are device-dependent. For detailed information about the system parameters discussed in this chapter, refer to Appendix A in the *Command Language Reference Manual*, publication number 900005.

## Modifying the template files

Before you use `gen_dev`, you must modify the standard `config_dev` and `spx_maint.inp` files to include site-specific device and system parameter information. You need to:

1. Use the UNIX editor to modify `$BUILD/config_dev` and `$BUILD/spx_maint.inp` to include the specific library devices and system parameters that the customer's system requires.
2. Add any additional system parameters that may also be required.

## Adding entries to `spx_maint.inp`

You must add/define system parameter entries in the `spx_maint.inp` file for each library device that you define in `config_dev`. The `gen_dev` utility compares device information in both files for consistency and exits with an error if the files are not consistent. It is critical that you keep the information in these files in sync.

To keep `config_dev` and `spx_maint.inp` in sync, you must add the following entries in `spx_maint.inp` for each library:

This entry	Adds
<code>ADD DEFAULT_MED_did</code>	The default media system parameter for the library device indicated by the specified device identification code (did).
<code>ADD FREE_POOL_didmmr</code>	A free pool volume set system parameter for the specified library device identification code and media/recording type (didmmr).
<code>ADD AUTO_REBUILD_did</code>	A library device rebuild system parameter for the specified library (did).

For example, if you add library L01 to the config\_dev file and that library will contain 4X erasable optical media (media type of OE and recording type of D), you must add the following system parameters for L01 to the spx\_maint.inp file: DEFAULT\_MED\_L01, FREE\_POOL\_L01OED, and AUTO\_REBUILD\_L01.

For some systems, you must define additional parameters in spx\_maint.inp. For instance, if you are installing a customer system that uses ACSLS or Library Station software to share automated tape libraries with StorHouse and non-StorHouse applications, you must add the BARCODE\_MIN\_n and BARCODE\_MAX\_n system parameters to the spx\_maint.inp file.

## Installing new systems – build\_sxm and gen\_dev

When you install a new StorHouse system, you do not explicitly execute the gen\_dev utility. Instead, you execute the build\_sxm utility, which automatically calls gen\_dev to build the bconfig file and other system files.

When called by build\_sxm, gen\_dev executes three gen\_? scripts, where ? can be b, j, or r. Each of these gen\_dev scripts contains the shell commands to generate or update systems files for one of the following StorHouse software facilities:

Facility	Definition
Basic device support (B)	Provides a way to access all user data storage and network devices used for data transfer such that the specific device characteristics for a device type are transparent to the rest of the StorHouse software. Also logs errors generated by these devices and maximizes throughput for data transfer requests.
Storage allocation manager (J)	Manages the allocation of storage for user data such that the specific characteristics of the storage media do not have to be known by other StorHouse facilities.
Resource management (R)	Controls data transfer, volume management, and some aspects of file storage.

## **build\_sxm tasks**

The `build_sxm` utility executes the following high-level tasks before calling `gen_dev`:

- Prompts you to confirm deletion of existing system files and the catalog of user files
- Creates the `$BUILD/config.sh` file, which `gen_dev` uses as a log file to record commands executed by various script files in subsequent phases of the `gen_dev` process
- Initializes the permanent storage area (NVM or disk file) on the new StorHouse system

Refer to Chapter 7 for more information on the `build_sxm` utility.

## **gen\_dev tasks**

The `gen_dev` utility performs the following high-level tasks:

1. Retrieves the relevant script commands from the `config_dev` file and compares device information in `config_dev` and `spx_maint.inp`
2. Deletes the current base configuration system file (`bconfig`) and generates a new one
3. Creates the library device volume system file (`bldv`)
4. Generates system files for the J facility
5. Generates system files for the R facility
6. Determines whether the `$STH_RELEASE` environment variable is defined on the current StorHouse system. If so, executes the `gen_sth` script.

The following diagram illustrates these tasks.

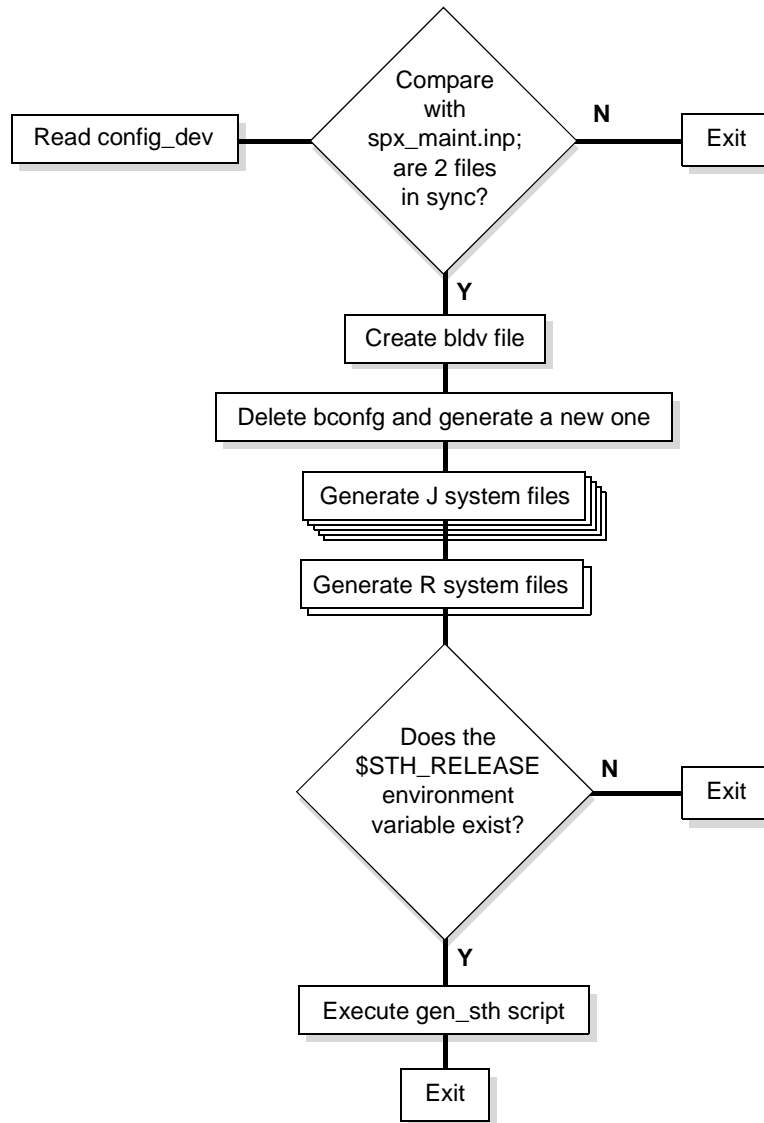


Table 23-1 presents a detailed listing of these gen\_dev tasks and their sub-tasks.

**Table 23-1: gen\_dev process for a new StorHouse system**

Phase	Performed by	Description
1	gen_dev utility	<p>Performs the gen_dev check function by comparing device information in the config_dev file and the spx_maint.inp file.</p> <ul style="list-style-type: none"> <li>■ If the device information is consistent, deletes the initial generic bconfig (base configuration) system file and proceeds to phase 2.</li> <li>■ If the device information is inconsistent, reports an error and terminates processing.</li> </ul>
2	gen_dev utility	Executes the gen_b script.
2a	gen_b script	Creates two temporary UNIX script files, bconfig_maint.sh and bldv_maint.sh, using the config_dev file as input.
2b	bconfig_maint script	<ul style="list-style-type: none"> <li>■ Invokes the bconfig_maint utility.</li> <li>■ Creates a new and shadow copy of the bconfig (device configuration) system file.</li> <li>■ Executes the bconfig_maint commands from the config_dev file.</li> </ul>
2c	bldv_maint script	<ul style="list-style-type: none"> <li>■ Invokes the bldv_maint utility.</li> <li>■ Creates the bldv system file.</li> <li>■ Executes the bldv_maint commands from the config_dev file.</li> </ul>
3	gen_dev utility	Executes the gen_j script.
3a	gen_j script	<ul style="list-style-type: none"> <li>■ Creates the jbol (backout list) system file.</li> <li>■ Creates a script file, jgen_n.sh.</li> </ul>

**Table 23-1: gen\_dev process for a new StorHouse system (continued)**

Phase	Performed by	Description
3b	jgen_n.sh	<ul style="list-style-type: none"> <li>■ Invokes the jgen utility.</li> <li>■ Creates the following system files: <ul style="list-style-type: none"> <li>■ jvset – volume set information file.</li> <li>■ jfset – file set information file.</li> <li>■ jsset – surface set information file.</li> <li>■ jsurf – surface partition information file.</li> </ul> </li> <li>■ Executes the jgen commands from the config_dev file.</li> </ul>
4	gen_dev utility	Executes the gen_r script.
4a	gen_r script	<p>Invokes the following utilities and supplies commands as input:</p> <ul style="list-style-type: none"> <li>■ rvol_maint – maintains the volume information (rvol) system file.</li> <li>■ ract_maint – maintains the active transactions (ract) system file.</li> <li>■ re_maint – maintains the file extent information (reinfo and reinfo2) and cross-reference (rexref) system files.</li> <li>■ rpb_maint – maintains the performance buffer contents information (rpb) system file.</li> <li>■ roper_maint – maintains the operator action (roper) system file.</li> </ul> <p>As a result of executing these utilities, the gen_r script creates the reinfo (file extent information file) and rvol (volume information file) system files.</p>
5	gen_dev utility	Determines whether the \$STH_RELEASE environment variable is defined (which indicates that StorHouse/RM is installed); if so, gen_dev executes the gen_sth utility.
5a	gen_sth utility	<ul style="list-style-type: none"> <li>■ Creates a script file, sth_config.sh.</li> <li>■ Retrieves the drda.input commands from the config_dev file and copies them into a new file, \$BUILD/drda.input.</li> </ul>



**Table 23-1: gen\_dev process for a new StorHouse system (continued)**

Phase	Performed by	Description
5b	sth_config script	Invokes the config utility with the following parameters: <ul style="list-style-type: none"><li>■ drda.input (created in phase 5a).</li><li>■ drda.config (a component of the StorHouse/RM software).</li></ul>
5c	gen_sth utility	Executes the sth_config script and saves the output to \$BUILD/config.sh.

## Updating an existing system with gen\_dev

You must explicitly execute `gen_dev` whenever you need to modify the physical configuration of an existing StorHouse system. For example, you may need to add a new library to an existing system or add a new media type or more drives to an existing library.

For existing systems, `gen_dev` ignores existing R and J facility system files and rebuilds the `bconfig` file only. After you execute `gen_dev` for an existing system, you may need to use J or R facility utilities to reconfigure the J or R system files. For example, you'd use the `jgen` utility to update the J facility files if you were adding a library device, but you wouldn't have to update the R files. For more information about `jgen`, see the internal document *J Facility Utilities*, publication number J7-1.1 Rel 3.0. For information about procedures that use `gen_dev`, see the *StorHouse Installation and Upgrade Guide*.

## Preparing to execute gen\_dev

Before you explicitly execute `gen_dev`, you must manually update the `config_dev` file (and, depending on the type of change you're making to the StorHouse configuration, the `spx_maint.inp` file) with new device information. Once you have modified these files, you can run `gen_dev` to rebuild the customer's `bconfg` file.

## gen\_dev tasks

Table 23-2 provides a detailed list of the `gen_dev` tasks for an existing StorHouse system.

**Table 23-2: gen\_dev process for an existing system**

Phase	Performed by	Description
1	gen_dev utility	<p>Performs the <code>gen_dev</code> check function by comparing device information in the <code>config_dev</code> file and the <code>spx_maint.inp</code> file.</p> <ul style="list-style-type: none"> <li>■ If the device information is consistent, deletes the <code>bconfg</code> (base configuration) system file and proceeds to phase 2.</li> <li>■ If the device information is inconsistent, terminates processing and reports an error.</li> </ul>
2	gen_dev utility	Executes the <code>gen_b</code> script.
2a	gen_b script	Creates two temporary UNIX script files, <code>bconfg_maint.sh</code> and <code>bldv_maint.sh</code> , using the <code>config_dev</code> file as input.

Table 23-2: gen\_dev process for an existing system (continued)

Phase	Performed by	Description
2b	bconfig_maint script	<ul style="list-style-type: none"> <li>■ Invokes the bconfig_maint utility.</li> <li>■ Creates a new and shadow copy of the bconfig (device configuration) system file.</li> <li>■ Executes the bconfig_maint commands from the config_dev file.</li> </ul>
2c	bldv_maint script	<ul style="list-style-type: none"> <li>■ Invokes the bldv_maint utility.</li> <li>■ Regenerates the bldv system file.</li> <li>■ Executes the bldv_maint commands from the config_dev file.</li> </ul>

## Format

The gen\_dev command format is:

gen\_dev [check]

Argument	Description
[no argument]	<p>Checks for consistency between the device information in the config_dev and spx_maint.inp files.</p> <ul style="list-style-type: none"> <li>■ If the information in the two files is consistent, gen_dev proceeds to update the Base facility files to reflect the new StorHouse configuration.</li> <li>■ If the information is inconsistent, gen_dev reports an error and terminates processing.</li> </ul>
check	<p>Checks for consistency between the device information in the config_dev and spx_maint.inp files and ensures that there are no duplicate device names. Displays the message “configuration files are OK” if the data in the files is consistent.</p> <p>Note that running gen_dev with the check argument does not update the Base facility system files.</p>

## Examples

The following examples show how to execute `gen_dev` with and without the `check` argument.

### Example 1

The following command checks for consistency between the `config_dev` and `spx_maint.inp` files, then exits the `gen_dev` utility without updating device information:

```
gen_dev check
```

If the information in the two files matches, `gen_dev` issues the following message:

```
configuration files are OK.
```

### Example 2

The following command checks for consistency between the `config_dev` and `spx_maint.inp` files. If the files are consistent, it updates the Base facility files to reflect the new StorHouse configuration and sends the output of the command to a log file (`gen_dev.log`):

```
gen_dev >& gen_dev.log
```

## genqmm utility

The Generate Quality Maintenance Manager (genqmm) utility generates a set of StorHouse system files that describe default values for all possible devices and media/recording types in a StorHouse system.

During the StorHouse installation process, the build\_sxm utility executes genqmm automatically to build the initial qmm system files. Subsequently, you run genqmm (from the StorHouse operating system prompt) to reinitialize the system files *only* when a new version of the StorHouse software contains a modified version of the genqmm script file *and* you are upgrading an existing StorHouse system to that version.

If you modify the configuration of an existing StorHouse system, such as add a new library or media type, update the existing qmm system files using the qm\_maint utility rather than replace the files using genqmm. By updating the existing files, you preserve any site-specific information you previously added to the files.

## Files generated by genqmm

The following table describes the StorHouse system files generated by genqmm.

File name	Description
qmmdeflt	Default parameters file, which contains initial threshold values for device types (for example, HITACHI and FILENET) and media types (for example, OAB).
qmmparam	Customized parameters file, which contains the site's local parameters. All devices and media types specified in this file must also exist in the bconfg file. A record in this file for a particular device or type of device overrides the corresponding record in the default parameters file.
qmmmedia	Media statistics file, which tracks statistics for each surface of each physical volume used by StorHouse.
qmmdevsm	Device file, which tracks statistics for devices other than media (drives, libraries, accessors, exchange stations, and slots). Statistics are accumulated for the life of the device as 'permanent' statistics.

## Format

genqmm

## gzip utility

The gzip utility is a UNIX program that compresses, or reduces, the size of a UNIX file. In addition, gzip decompresses a UNIX file that was previously compressed using gzip. You can use this utility to create a fixes file containing replacement software files to be installed on a customer's StorHouse system using either the asd or AUTOMV utility (refer to Chapter 3, "asd utility" or Chapter 4, "AUTOMV utility," for information on using those utilities).

When you use gzip, the utility replaces the file you specify with a compressed version and appends a suffix of .gz to the file name. For example, if you create an asd fixes file named my\_file.asd, gzip compresses the file and renames it as follows:

```
my_file.asd.gz
```

The degree of compression depends on what's in the file, although 50 percent compression for text files is typical.

## Format

```
gzip [-dv] {filename}
```

Option/Argument	Description
-d	Decompresses a UNIX file that was previously compressed using gzip.
-v	Reports the percentage of disk space gzip saved by compressing the file.
<i>filename</i>	Path (if the file is not located in the operator's home directory) and file name of the file to be compressed.  If you are compressing a software fixes file, the file name should adhere to the following naming conventions: <ul style="list-style-type: none"><li>■ asd utility: <i>filename.asd</i></li><li>■ AUTOMV utility: <i>AUTOMV.TAR.filename</i></li></ul>

---

## Sample output

The following sample output results when you issue the gzip command with the verbose option for an asd fixes file.

```
gzip -v my_file.asd
```

```
++my_file.asd:      69.9%
```

```
++ gzip test: rate = 38.893 KB/second
```

```
-- replaced with my_file.asd.gz
```



## **maint utility**

The System Maintenance (maint) utility provides a user-friendly means of performing maintenance on a StorHouse system. You can use the maint utility to perform the following tasks:

- Change the status of a library or level F device from up to down or vice versa
- Enable or force a library rebuild, for a single library or all libraries at once
- Set a level F device to INITIALIZE DEVICE mode
- Enable migrations
- Abort, drain, or disable migrations or backups

You can execute the maint utility in either interactive or batch mode. When you execute the utility interactively, it displays a menu with a list of available options, a set of online instructions, and a maint> command prompt (see the figure on page 26-2).

```
System Maintenance Utility
Checking system availability...
System is available.
Select function:
  1) Toggle library device status
  2) Enable library rebuild
  3) Force library rebuild
11) Toggle level F device status
12) Set level F device into INITIALIZE DEVICE mode
21) Enable migrations
22) Abort/disable migrations/backups
23) Drain/disable migrations/backups
Enter entry number for function desired, then press ENTER.
Press just the ENTER key to exit this menu.
maint>
```

**Main menu of the maint utility**

## Format

maint

At the maint> prompt, type the option number that corresponds to the function you want to perform, then press **Enter**. Or, press **Enter** without specifying an option to exit the utility.

After you select an option from the main menu, the maint utility either performs the associated function or displays a list of libraries or device names from which you can choose. To select a device, type either the option number or the device name and press **Enter**.

Option	Function	Description
1	Toggle library device status	Alternates the status for a library device between offline and online. Use this option once to put the device in an offline state before you perform maintenance on the device, then again to return the device to an online state following maintenance.
2	Enable library rebuild	<p>Enables automatic library device rebuild for the specified library if it needs it at initialization. For example, a rebuild would be necessary if the volume directory for the device had been corrupted or destroyed. (This option sets the AUTO_REBUILD_Lxx system parameter to TRUE.)</p> <p>When you perform a rebuild, StorHouse mounts every volume in the specified library device and reads the label of each volume. It then updates the bldv file to reflect the actual volume IDs in the library.</p>
3	Force library rebuild	<p>Forces StorHouse to rebuild the volume directory for the specified library at initialization regardless of whether StorHouse deems it necessary. Use this option if you have reason to believe the volume directory does not reflect the actual contents of the library.</p> <p><b>Warning:</b> Depending on factors such as the device type and number of volumes, the library rebuild process can be very time-consuming. In addition, you can't cancel the process once you initiate it. Therefore, be certain that a library rebuild is absolutely necessary before you proceed. The maint utility displays a warning message and prompts you to confirm whether you want to proceed.</p>
11	Toggle level F device status	Alternates the status for a level F device between offline and online. Use this option once to put the device in an offline state before you perform maintenance on the device, then again to return the device to an online state following maintenance.

Option	Function	Description
12	Set level F device into INITIALIZE DEVICE mode	<p>Causes StorHouse to ignore the specified level F device during the normal recovery process that takes place when StorHouse comes up.</p> <p>The normal recovery process can't recover data from a broken magnetic disk unit. This, in turn, prevents StorHouse from coming up. Use this option to allow StorHouse to bypass recovery of the magnetic disk unit and come up.</p> <p><b>Warning:</b> Using this option causes a loss of all customer data on the level F device. The maint utility displays a warning message and prompts you to confirm whether you want to proceed.</p>
21	Enable migrations	<p>Enables the StorHouse system to migrate files from the performance buffer to the library devices. Use this option to restore the system's ability to perform migrations if you previously disabled migration using option 22 or 23. (This option sets the MIG_FROM system parameter to TRUE.)</p>
22	Abort/disable migrations/backups	<p>Immediately stops in-progress requests to transfer or copy files from the performance buffer to the library devices. In addition, this option disables the system's ability to perform future migrations until you re-enable them using option 21. (This option sets the MIG_FROM system parameter to FALSE.)</p>

Option	Function	Description
		<p><b>Note:</b> If the customer is using non-erasable, or Write-Once-Read-Many (WORM), media, using this option rather than option 23 potentially wastes space on the customer's optical device. The portion of the file extents that were already transferred when you aborted the migration or backup remain on the customer's StorHouse system in unusable form.</p> <p>FileTek recommends that you use option 22 <i>only</i> when you can't wait for in-progress requests to complete and you're willing to risk wasting media. Use option 23 in all other circumstances.</p>
23	Drain/disable migrations/backups	<p>Allows in-progress transfer or copy requests to complete but removes any additional pending (queued) migration and backup requests. In addition, this option disables the system's ability to perform future migrations until you re-enable them using option 21. (This option sets the MIG_FROM system parameter to FALSE.)</p> <p><b>Note:</b> This option is available only on a StorHouse 5.2 system or higher.</p>

## Sample output

The following sections illustrate the sample output that results from executing the maint utility using three different options.

### Sample 1

The following sample output results when you execute the maint utility and select option 1, Toggle library device status, from the main menu.

To select a device, → Select library device:

type either the option number or the device name, then press Enter. For example, type 1 or L00, then press Enter.

1) L00	ONLINE
2) L00A00	ONLINE
3) L00D00	ONLINE
4) L00D01	ONLINE
5) L00E00	ONLINE
6) L01	ONLINE
7) L01A00	ONLINE
8) L01D00	ONLINE
9) L01D01	ONLINE
10) L01D02	ONLINE
11) L01D03	ONLINE
12) L01E00	ONLINE
13) L02	OFFLINE
14) L02A00	OFFLINE
15) L02A01	OFFLINE
16) L02D00	OFFLINE
17) L02E00	OFFLINE

Enter entry number for device to be toggled, then press ENTER.  
Or enter library device name, then press ENTER.  
Press just the ENTER key to exit this menu.

maint>

## Sample 2

The following sample output results when you execute the maint utility and select option 2, Enable library rebuild, from the main menu.

Select library:

- 0) All libraries
- 1) L00     ONLINE
- 2) L01     ONLINE
- 3) L02     OFFLINE

Enter entry number for library to be auto-rebuild enabled, then press ENTER.  
Or enter "All" or library name, then press ENTER.  
Press just the ENTER key to exit this menu.

maint>

## Sample 3

The following sample output results when you execute the maint utility and select option 23, Drain/disable migrations/backups, from the main menu.

\*\*\*\*\*

\* Migration has been disabled, remember to re-enable it.

\*\*\*\*\*





## make\_bkups utility

The Make Backups (make\_bkups) utility backs up the system disk (boot device) on a StorHouse server. Use this utility to create a shadow copy of the root file system after you add a new library or otherwise reconfigure an existing StorHouse system.

The make\_bkups utility creates backup copies of all the standard operating system (os) disk partitions, if they exist:

- / (root)
- /usr
- /usr/openwin
- /var
- /opt

In addition, make\_bkups creates a copy of either the operator's home directory (/filetek/operator) or the contents of /filetek (except the smd directory), depending on the command option you choose. Finally, make\_bkups runs installboot on the backup root partition (to install boot block) and modifies the copy of the /etc/vfstab (file system table) file that resides on the backup disk to make the backup disk bootable in the event the boot disk becomes unusable.

You must log in to the StorHouse server operating system as superuser (using the root account) to execute the make\_bkups utility. From the root directory, either specify the complete path name when you type the make\_bkups command, or change to the /filetek/bin directory then execute ./make\_bkups.

## Before you use make\_bkups

If you plan to execute the make\_bkups utility using the -f (fast) option (see a description of this option in “Format”), ensure that the source and target disks are identical in size before you run the utility. In addition, ensure that the StorHouse system is in a quiesced (idle) state prior to running the make\_bkups utility with this option.

## Format

/filetek/bin/make\_bkups [-fa]

Option	Description
[no option]	<p>Performs a file-level backup of the system disk and the operator's home directory (/filetek/operator). This form of backup (which is the default method):</p> <ul style="list-style-type: none"><li>■ Copies new versions of files to the existing backup disk</li><li>■ Doesn't delete obsolete files</li><li>■ Is relatively slow</li><li>■ May report a lot of errors, which you can ignore</li></ul> <p>Execute the make_bkups command with no options if you are running StorHouse in multi-user mode.</p>

Option	Description
-f[ast]	<p>Performs a block-level backup of the entire system disk partition (copies not only the files, but the file system information) and a file-level backup of the operator's home directory. This option completely overwrites the operating system data on the backup disk, thereby deleting obsolete files. However, it doesn't delete obsolete files in /filetek/operator.</p> <p>This form of backup is faster than the default method. However, this option is available only when the os and bkup partitions are the same size.</p>
-a[l ]	<p>Performs a file-level backup of the contents of /filetek to /filetek2 (with the exception of the /filetek/smd directory).</p> <p><b>Note:</b> This option doesn't delete obsolete files in /filetek2, regardless of whether you use it in conjunction with the -f option.</p>

## Examples

This section presents two examples of how to use the make\_bkups utility.

### Example 1

The following command performs a block-level backup of the entire system disk partition (which deletes obsolete operating system files on the backup disk) and a file-level backup of the operator's home directory:

```
/filetek/bin/make_bkups -f
```

## **Example 2**

The following command performs a block-level backup of the entire system disk partition and a file-level backup of /filetek to /filetek2 (with the exception of the /filetek/smd directory):

```
/filetek/bin/make_bkups -af
```

## meta\_rstr utility

The metadata restore (meta\_rstr) utility recovers metadata for one or more StorHouse databases from backup files. Use this utility to restore the system tables and other system components for a database in the unlikely event that the magnetic disks containing the database directories fail. Recovery of the last valid version of the metadata is critical to the client's operation because, when the metadata files for a database become corrupt or are destroyed, users are unable to access data in the database.

As an alternative to using the meta\_rstr utility, you can restore the metadata for a database using the StorHouse/Admin component of Control Center. Refer to the *StorHouse/Admin Database Administrator's Quick Reference*, publication number 900150, or the Control Center online help for StorHouse/Admin for more information.

Note that you can restore *only* metadata that was backed up previously using the metadata backup (meta\_bkup) utility or the metadata backup feature of StorHouse/Admin.

## How meta\_rstr works

The meta\_rstr utility copies the most recent metadata backup file (version 0) for each database you specify from StorHouse into a temporary directory. It

subsequently restores the metadata files into the appropriate database directories. The actual steps performed by meta\_rstr are described in the following table.

#### Steps performed by the meta\_rstr utility

Step	Description
1	Verifies that the \$STHROOT and \$STHDBS environment variables exist and point to directories.
2	Verifies that metadata directories exist for the specified database(s).
3	Verifies that StorHouse is up and that the system parameters SQL_BKUP_ACCOUNT and SQL_BKUP_GROUP are defined. If these system parameters aren't defined, the utility aborts.
4	Verifies that the database is DOWN (e.g., set by the dbdown utility). This action ensures that StorHouse engines are unable to use the incomplete metadata during the extraction phase.
5	Locates the StorHouse/SM file containing the backed up metadata for the specified database(s) and retrieves the latest version of this file into a temporary directory.
6	Extracts the contents of the metadata backup file (using the UNIX tar utility) and restores the metadata into \$STHDBS.
7	When the metadata files for the database(s) have been restored to the state they were in at the time of the metadata backup, meta_rstr deletes the temporary files, records the successful completion of the utility in the StorHouse administration log, and writes a message to standard output (stdout) indicating that meta_rstr completed successfully.

The dbup utility must be used to make the database accessible.

The meta\_rstr utility writes status messages to standard output (stdout), which is usually defined as your terminal. Errors are reported to both stdout and to the StorHouse administration log. If the utility encounters a severe (fatal) error, it displays a message describing the error at the StorHouse console as well. Refer to “Correcting severe error conditions” on page 28-4 for a list of fatal error messages, explanations, and suggested actions.

## Before you use meta\_rstr

Before you use the meta\_rstr utility, ensure that:

- The StorHouse system on which you plan to execute metadata restore is operational and has StorHouse/RM software on it. In addition, if the restore is part of a general disaster recovery, ensure that the StorHouse/SM files have already been recovered and the StorHouse/SM system is fully operational.
- You are familiar with Chapter 10 of the *StorHouse Database Administration Guide*, which discusses metadata backup.
- Either you or the StorHouse database administrator previously set up the metadata backup environment as described in Chapter 10 of the *StorHouse Database Administration Guide* and backed up the metadata before the metadata files became corrupt or were destroyed.
- The database to be restored exists. If so, use the UNIX mv command to rename it to *database\_name*.replaced, where *database\_name* is the database you want to restore. Then use the UNIX mkdir command to create a new *database\_name.dbs* directory. These steps enable you to revert to the prior version of the metadata if you subsequently decide to abort the metadata recovery process.

If the database to be restored does not exist (for example, someone inadvertently deleted it), you need only create the directory \$STHDBS/*database\_name.dbs*.

- You DOWN the database using the dbdown utility:

```
dbdown -d database_name
```

Note that this will terminate user sessions.

## Format

meta\_rstr [-n] {*database\_name*} [*database\_name* ...]

Argument	Description
-n	Option that indicates that you are running meta_rstr in a non-interactive mode.
<i>database_name</i>	Name of the StorHouse database(s) whose metadata you want to restore. Enclose database names that contain lowercase characters in double quotes.

## Correcting severe error conditions

If the meta\_rstr utility encounters a severe (fatal) error during execution, it displays a message at the StorHouse console. Fatal errors result when the meta\_rstr utility is unable to finish executing due to a serious problem.

The remainder of this chapter describes the format of fatal error messages, lists and describes the messages, and provides a problem determination checklist to assist you in diagnosing and resolving the problems that produce the error messages.

## Format of error messages

Each meta\_rstr error message includes:

- A prefix of RMBR
- A three-digit numeric return code (for example, 001)
- The message text
- An explanation of the message



The meta\_rstr utility also provides a “User Response” section for each error message. Because this section is intended for use by customers and often contains the text “Contact FileTek Customer Support,” it has been replaced in this manual with a “Support Response” section. As the name implies, this replacement section provides suggested responses for support personnel.

Additionally, this manual provides a “Problem Determination” section for some messages which, when used in conjunction with the Problem determination checklist described in the following section, provides additional assistance in diagnosing and resolving error conditions.

## **Problem determination checklist**

Many of the fatal error messages produced by meta\_rstr can be attributed to one or more basic problems. These problems, and the suggested actions to resolve them, are described in the “Problem determination checklist” shown on page 28-6.

When an error message provides a “Problem Determination” section, consult the appropriate section of the checklist. The item number(s) in the “Problem Determination” section for a message correspond to the item numbers in the checklist.

**Problem determination checklist**

Item	Action
1	<p>Ensure that you logged in to the StorHouse server using the UNIX operator account. This account sets up the operating environment that is necessary for a successful execution of the meta_rstr utility. If you used an account other than operator, exit the StorHouse system, log in as operator, and reexecute the meta_rstr utility.</p> <p>If a customer requires that you log in using an account other than operator for security reasons, verify that the account permissions and other data for the account you are using are identical to that of the operator account. In addition, it is recommended that you source the operator account's .cshrc file by executing the following command:</p> <pre>source ~operator/.cshrc</pre> <p>Also, check all files you create to ensure that the operator:sm account can access them.</p>
2	<p>Verify that the StorHouse/RM software resides on the system and is installed correctly by performing the following steps:</p> <ol style="list-style-type: none"> <li>1. Ensure that the .cshrc.rm file, which defines environment variables for StorHouse/RM, exists and hasn't been damaged. If necessary, recreate the file.</li> <li>2. Enter the following command at the StorHouse operating system (UNIX) prompt to determine whether the StorHouse/RM environment variables are defined: <pre>setenv</pre> </li> <li>3. Verify that the following environment variables appear in the list (x corresponds to the release level of the software): <pre>STH_RELEASE=vxry STHROOT=/filetek/sth/vxry STHDBS=/filetek/sth/sthdb STHLOG=/filetek2/general/sth STHSQLNW=sqlnw</pre> <p>Note that the release may also contain a StorHouse/SM release id. For example, STH_RELEASE may have a value such as v3r2_v5r4, where v3r2 is the StorHouse/RM release and v5r4 is the StorHouse/SM release.</p> </li> </ol>

**Problem determination checklist (continued)**

Item	Action
	<p>4. Ensure that the directory defined for the \$STHROOT environment variable exists and that the release level specified for the software is correct. In addition, ensure that the other environment variables are set up correctly.</p> <p>5. If one or more of the environment variables hasn't been defined or contains an incorrect value (for example, the software release defined for \$STHROOT is different from the release defined for \$STH_RELEASE), add the variable or correct its value, respectively.</p>
3	<p>Verify that the database directory (\$STHDBS/database_name.dbs with the correct case and spelling) exists, is owned by operator with UNIX group sm, and has read, write, and execute privilege for the owner.</p> <p>If the directory has been removed or destroyed, use the UNIX mkdir command to recreate it (remember that database names are case sensitive and must have a suffix of .dbs). You do not have to create any files in the directory.</p>
4	<p>Locate the .cshrc.rm file and verify that it is not damaged or corrupt. If necessary, recreate the file.</p>
5	<p>If you reconstructed a StorHouse system that had been destroyed, do the following:</p> <ul style="list-style-type: none"> <li>■ Make sure you understand the nature of the problem that necessitated the restoration.</li> <li>■ Review the steps you took to reconstruct StorHouse/SM to ensure that you rebuilt it correctly. If necessary, correct any errors you made in rebuilding StorHouse/SM.</li> <li>■ Ensure that you reset the values for the StorHouse/SM system parameters to those used by the customer prior to the disaster. If necessary, change the values for the parameters from the default values created during the installation process to those used by the customer prior to the reconstruction of the system.</li> <li>■ Verify whether the customer performed a metadata backup prior to the time the StorHouse system failed. If not, no data exists for restoration by the meta_rstr utility.</li> </ul>

**Problem determination checklist (continued)**

Item	Action
6	<p>During execution, the meta_rstr utility locks the specified database(s) to prevent users from accessing “dirty” data during the metadata restore procedure. If an error occurs that prevents meta_rstr from locking the database, do the following:</p> <ol style="list-style-type: none"> <li>1. Execute stopsm to shut down the StorHouse system.</li> <li>2. Execute cleanipcs to release StorHouse shared resources (including locks) to UNIX and display error messages at your terminal.</li> <li>3. If cleanipcs completes successfully, execute startsm to restart the system, then run meta_rstr again. If cleanipcs exits with errors, reboot the system. If necessary, contact the FileTek software development group for assistance following the reboot.</li> </ol>
7	<p>The meta_rstr utility executes the UNIX tar utility to extract the contents of the metadata backup tar file. If the tar utility is unable to complete successfully, the error may be due to any of the following problems:</p> <ul style="list-style-type: none"> <li>■ The account you used to log in to the StorHouse server doesn’t have the necessary permissions.</li> <li>■ The system has insufficient memory.</li> <li>■ The file system is full.</li> <li>■ The hardware (disk) is defective.</li> </ul>

If you’ve performed all suggested problem determination steps in this table and you’re still unable to resolve the problem, request assistance from the FileTek software development group. Be sure to provide them with detailed information on problem symptoms and resolution steps.

## List of error messages

This section lists and describes fatal error messages that might result during execution of the meta\_rstr utility. Messages are arranged in ascending order by numeric return code. In the message text, information between backward slashes (\) represents variables.

In most cases, the “Support Response” section of a message provides sufficient information to enable you to resolve a meta\_rstr error condition. Messages that require more extensive analysis and effort to correct include a “Problem Determination” section, which references one or more items in the “Problem determination checklist” on page 28-6.

**RMBR001    StorHouse/RM software not installed**

**Explanation:** You tried to run the metadata restore utility on a StorHouse system on which StorHouse/RM software isn't installed or wasn't set up correctly.

**Problem Determination:** Item 2.

**RMBR002    No database name specified**

**Explanation:** The metadata restore command did not contain a database name. At least one database name is required.

**Support Response:** On the metadata restore command, specify the name of the database you want to restore. Database names are case sensitive, so if the name contains lowercase characters, enclose it in double quotes.

**RMBR003    Invalid command line option**

**Explanation:** You entered a command argument that began with a hyphen (-).

**Support Response:** Check the command line. Be sure the only arguments following the metadata restore command are valid database names and do not start with a hyphen.

**RMBR004    SM System Parameter not defined \parameter\_name\**

**Explanation:** A required StorHouse system parameter doesn't have a value. In order to run the metadata restore utility, the following system parameters require values: SQL\_BKUP\_ACCOUNT and SQL\_BKUP\_GROUP.

**Support Response:** Set a value for the identified parameter\_name. Refer to Appendix C, “System parameters for StorHouse/RM” in the StorHouse *Database Administration Guide* for more information about setting system parameters.

**RMBR005    Environment variable not defined \variable\_name\**

**Explanation:** A required environment variable—\$STHDBS or \$STHROOT—doesn't have a value.

**Problem Determination:** Items 1, 2.

**RMBR006    Database does not exist \database\_name\**

**Explanation:** The database name supplied on the command line refers to a database that doesn't exist in StorHouse.

**Support Response:** Verify that the database directory (\$STHDBS/*database\_name.dbs*) exists. Resubmit the metadata restore command with the correct database name. Be sure to type the exact database name. Database names are case sensitive. Enclose names with lowercase letters in double quotes.

**Problem Determination:** Item 3.

**RMBR010    Error from 'tar' utility \errno\**

**Explanation:** The UNIX utility that copies the metadata files and range index files from a temporary file to the database directory encountered an error. The errno is a UNIX error number.

**Support Response:** Try to determine the cause of the error based on the text of the UNIX error message.

**Problem Determination:** Item 7.

**RMBR011      Error opening 'tar' file \filename errno\**

**Explanation:** The metadata restore utility couldn't open the identified temporary file. The errno is a UNIX error number.

**Support Response:** Try to determine the cause of the error based on the text of the UNIX error message. Also, verify whether the temporary file identified in the message text exists. If it doesn't, the StorHouse software contains errors and you must contact the FileTek software development group for assistance.

**RMBR012      Error returned from SM operation \operation return\_code\**

**Explanation:** StorHouse either couldn't open or read from the metadata backup file, or the StorHouse account specified on the SQL\_BKUP\_ACCOUNT system parameter is invalid or doesn't have the correct privilege. The return\_code is the StorHouse return code.

**Support Response:** Refer to the StorHouse *Messages and Codes Manual* for more information about the return\_code, then contact the customer's StorHouse system administrator for help resolving the StorHouse account or file error.

**Problem Determination:** Item 5.

**RMBR013      Authentication failed for SQL\_BKUP\_ACCOUNT \account\**

**Explanation:** The metadata restore utility couldn't obtain the password for the account specified on the SQL\_BKUP\_ACCOUNT system parameter or the account is invalid.

**Support Response:** Contact the customer's StorHouse system administrator. If the system parameter contains an incorrect value or the account is set up incorrectly, it's unlikely that the metadata backup was successful.

**Problem Determination:** Item 5.

**RMBR014      Error writing 'tar' file \filename errno\**

**Explanation:** The UNIX utility that copies the metadata backup file from StorHouse to the temporary file encountered an error. The errno is a UNIX error number.

**Support Response:** Try to determine the cause of the error based on the text of the UNIX error message.

**Problem Determination:** Item 7.

**RMBR015      Unable to access database directory \directory\**

**Explanation:** The metadata restore utility was unable to access the database UNIX directory.

**Problem Determination:** Items 1, 2, 3, 4.

**RMBR016      SM Message \message\_text\**

**Explanation:** An error has been reported from a StorHouse API operation. The “message\_text” contains the message returned by StorHouse. The RMBR016 error message is usually accompanied by other messages reporting the failing operation type and return code, for example, RMBR012.

**Support Response:** Refer to the “user response” for the accompanying messages(s).

**RMBR021      Error getting temporary directory \directory\**

**Explanation:** The file etc/rdbtemp.data in the RM root directory could not be processed. This file is used to locate the directory to be used for the temporary file built during metadata backup processing.

**Support Response:** Verify that the \$STHROOT/etc directory exists and can be referenced for reading by the operator account. Verify that the file rdbtemp.data exists and can be read. List the contents of this file (with more or cat) and validate that:



- The first line of the file contains two decimal digits, which indicate the number of lines in the rest of the file
- The remaining lines are names of directories that exist and can be referenced for writing by the operator account.

**RMBR023      Restore of database \database\ Completed with failure.**

**Explanation:** The metadata restore operation did not complete successfully. The metadata for “database” is not usable.

**Support Response:** Correct the problems indicated by other meta\_rstr error messages and re-run the restore.

**RMBR024      Database is still up. \database\. Bring database down with “dbdown”**

**Explanation:** A database must be set to the “down” state before it can be restored.

**Support Response:** Run the “dbdown” utility (directly from a UNIX prompt or through the StorHouse “run” command) to set the database to the downed state. Note that this will kill any current user connections to that database.

**RMBR025      Unable to extract backed up database \database sqlcode\**

**Explanation:** The StorHouse file containing the tar image of the database could not be retrieved from StorHouse and written to a temporary UNIX file. Prior error messages will indicate the specific failure that occurred.

**Support Response:** Problem resolution is determined by the preceding error messages.

**Problem Determination:** Item 7.



## **movevol utility**

The Move Volume (movevol) utility tests whether an optical library device and its drives are working properly. Use this program as a verification test following maintenance on a library. You can test multiple libraries in a StorHouse system concurrently by issuing the movevol command multiple times in succession and specifying a different library each time.

### **How movevol works**

The movevol utility enables you to test a single drive or all available drives within a library by moving all or a specified number of volumes from one location to another within the same library. If you think that more extensive testing of a library is necessary, you can specify that movevol repeats the test multiple times. As each volume is loaded in a drive, StorHouse reads the volume label and verifies that the drive is functioning properly.

### **Balancing the load of volumes to be moved**

When you're testing multiple drives in a library, the movevol utility attempts to balance the load of volumes to be moved among all available drives. To ensure that all drives in a library are functioning correctly, you should move at least as many volumes as there are drives in a library. For example, if the library contains four drives, you should move a minimum of four volumes to ensure that movevol exercises each drive.

## Reserving one or more library drives

Ordinarily, you should avoid running the movevol utility during production use. However, if you must perform the movevol test during normal business hours, you can specify that the movevol utility “reserve” one or more drives in the library you want to test to prevent users from accessing those drives until movevol completes. Reserving a drive eliminates the possibility that users might experience performance problems caused by competing for the same drive that movevol is using.

When you test a specific drive rather than all drives in a library, the movevol utility automatically reserves the drive before it begins to move volumes. Regardless of whether you reserved library drives or the movevol utility reserved a drive automatically, movevol removes each drive from a reserved state after it finishes executing.

## Monitoring the movement of volumes

After you issue the movevol command, the UNIX command prompt returns immediately and movevol begins executing in the background. Although movevol does not display messages to inform you of its progress, you can monitor the movement of the volumes using the Q (RQ Queues) or L (RL Queues) options of the rd utility. Refer to Chapter 32, “rd utility,” for more information on that utility.

## Format

Note the following before you use the movevol command:

- The status of a library device must be UP when you issue the movevol command.
- If the .cshrc.local file in the \$HOME directory for the UNIX operator account defines *move* as an alias for movevol, you can use the short form of the command.
- The movevol utility uses positional command syntax which means that, if you specify a given argument, you must also specify values for all of the arguments that *precede* it. You may, however, omit values for arguments that *follow* the argument you want to specify. When you omit values for succeeding arguments, the movevol utility uses the default values shown in the following table.

movevol [*library ID*] [*max volumes*] [*repeat count*] [*reserve drives*] [*drive to use*]

Argument	Description	Default
[no argument]	Moves each volume in library device L00 once and tests any or all available drives without reserving them.	–
<i>library ID</i>	ID of the library device that you want to test. Valid values are hexadecimal digits from 0 through f or F. For example, for library L01, specify 1.	0
<i>max volumes</i>	Maximum number of volumes that you want to move within a library. Valid values are: <ul style="list-style-type: none"><li>■ A zero value indicates that you want to move all volumes in the specified library.</li><li>■ A value greater than zero indicates that movevol should move only the specified number of volumes.</li></ul>	0

Argument	Description	Default
<i>repeat count</i>	<p>Number of times movevol should repeat the volume move process after the initial execution. Valid values are:</p> <ul style="list-style-type: none"> <li>■ A zero value indicates that you want to execute the movevol test only once.</li> <li>■ A value greater than zero indicates the number of times movevol should repeat after the initial execution. For example, if you specify 1, movevol will execute twice.</li> </ul> <p><b>Note:</b> When you specify a number greater than zero for this argument, movevol ignores the value you specified for the max volumes argument and moves all volumes in the specified library the specified number of times.</p>	0
<i>reserve drives</i>	<p>Integer that indicates whether you want to reserve the drives in the specified library device while movevol executes to prevent users from accessing them. Valid values are:</p> <ul style="list-style-type: none"> <li>■ A zero value indicates that you don't want to reserve the drive(s) in the specified library device.</li> <li>■ A value greater than zero indicates that movevol should reserve either all available drives or the specified drive(s) in the specified library device.</li> </ul>	0
<i>drive to use</i>	<p>ID of the specific drive that you want to test. For example, for drive L00D01, specify 1. Valid values are:</p> <ul style="list-style-type: none"> <li>■ A null value (space) indicates that you want to test any or all available drives.</li> <li>■ A hexadecimal number from 0 through f or F indicates that you want to test the drive with that ID.</li> </ul> <p><b>Note:</b> When you specify a particular drive to test, movevol automatically reserves the drive, regardless of the value you specified for the reserve drives argument.</p>	Null

## Examples

This section presents four examples of how to use the movevol utility.

### Example 1

The following command moves each volume in library device L00 once and exercises any or all available drives without reserving them:

```
movevol
```

### Example 2

The following command moves six volumes in library L01 once and exercises any or all available drives without reserving them:

```
movevol 1 6
```

### Example 3

The following command moves all volumes in library L01 three times each and exercises any or all available drives without reserving them:

```
movevol 1 0 2
```

### Example 4

The following command reserves drive D01 and moves each volume in library L01 once:

```
movevol 1 0 0 1 1
```





## port utility

The port utility simplifies the task of configuring internal a and b serial ports for a StorHouse system. These ports, which are located on the CPU board, allow users to log in to a StorHouse system from either a modem or a terminal. Initially, use the port utility following the installation of a new StorHouse system to configure these ports. Subsequently, use this utility when you need to modify the configuration of the ports used by StorHouse.

The port utility provides a “user-friendly” front end to the UNIX Port Monitor Administration (pmadm) program. In addition, port provides default values for most of the command arguments to further simplify the port configuration process. The port utility provides an alternative to configuring ports using either the UNIX pmadm program or the admintool program that is provided with the Sun® Solaris™ operating system.

You must log in to the StorHouse server operating system as superuser (using the root account) to execute the port utility.

## Format

port {add {port} {hardwired | dial-in | dial-out | bidirectional} | remove {port} | list [port]}

Command/ Argument	Description	Default
add {port}	Adds the specified port to the configuration for the StorHouse system.	—
hardwired	Specifies that the port you are adding is for a hardwired terminal (a terminal that's connected to the StorHouse system without using a modem).	hard_label=19200 hard_term=vt100
dial-in	Specifies that the port you are adding is for a modem with dial-in capability only. This option is rarely used.	dial_label=19200m dial_term=dialup
dial-out	Specifies that the port you are adding is for a modem with dial-out capability only. This option is rarely used.	dial_label=19200m dial_term=dialup
bidirectional	Specifies that the port you are adding is for a modem with bidirectional capability. A modem that is configured for bidirectional use can both place outgoing calls and receive incoming calls.	dial_label=19200m dial_term=dialup
remove {port}	Removes the specified port from the configuration for the StorHouse system.	—
list [port]	Displays details such as port status (enabled or disabled) for either the specified port or for all ports configured for the StorHouse system.	—
port	Value (either a or b) that identifies the port for which you want to view details or configure.	—

## Examples

This section provides two examples of using the port utility.

### Example 1

The following command:

```
port add a bidirectional
```

adds bidirectional port number “a” to the configuration for the StorHouse system.

### Example 2

The following command:

```
port list
```

lists all ports configured for the StorHouse system.



## qchsend utility

The qchsend utility issues a Call Home report to FileTek headquarters from a StorHouse system. You can use it to test the Call Home error reporting software in the following circumstances:

- After you install a new StorHouse system
- After you make changes to an existing StorHouse system
- When the Call Home software is not functioning properly

Call Home automatically identifies specific error conditions at a customer's site, reports them to FileTek's Support Center at headquarters or another destination, and notifies FileTek customer support personnel, as appropriate. Call Home responds to events and conditions such as predictive maintenance, critical disk space shortages, and unrecoverable device errors. See the *System Administrator's Guide*, publication number 900007, for a list of the events and conditions that generate a Call Home.

Even if the Call Home feature is not activated on a StorHouse system (that is, the CALL\_HOME system parameter is set to FALSE), you can force a call home using the qchsend utility.

To use the qchsend utility, first log on to the customer's StorHouse system. Then execute the qchsend utility, either from a shell script or as a standalone command from the StorHouse operating system (UNIX) prompt.

## Format

qchsend [-fp] [*text*]

Option	Description	Default
-f	Forces a Call Home, even if the CALL_HOME system parameter is FALSE.	
-p	Issues a page to FileTek customer support personnel.	
[ <i>text</i> ]	Transmits the specified message text in the Call Home report.	CALL HOME TEST— PLEASE IGNORE.

## Examples

This section presents two examples of how to run the qchsend utility. Both examples issue a page to FileTek customer support personnel.

### Example 1

The following command issues a Call Home to FileTek headquarters (assuming the CALL\_HOME system parameter is TRUE), issues a page to FileTek customer support personnel, and transmits the default message text described in the table above:

```
qchsend -p
```

### Example 2

The following command forces a Call Home, issues a page to FileTek customer support personnel, and transmits the specified message text:

```
qchsend -fp call home test from abc company
```

## Sample Call Home report

The following sample Call Home report results when you issue the qchsend command without any options:

From site: newsite  
Received at: 28-Apr-1999:08:49:00 EST

Site ID: NEWSITE  
Contact info: (555)-555-5555  
Occurred at: 28-APR-1999:05:42:23 PST  
Event type: Generic CALL HOME message, no paging (128)

This is the default  
event type. —————>

This is the default  
message text. —————>

Error text: CALL HOME TEST -- PLEASE IGNORE.  
Process: QCHSEND (4681)





## rd utility

The rd utility is the Resource Management Facility's Display Processor for StorHouse.

The primary responsibility of rd is to accept commands, examine internal StorHouse data structures, and present a real-time display of system activities. The rd utility is a valuable tool for demonstrating, describing, debugging, and testing StorHouse.

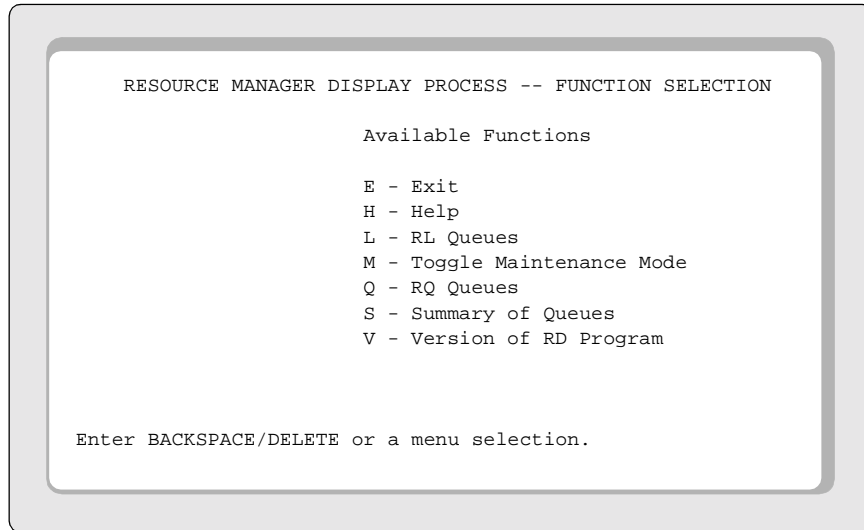
You can use the rd utility to produce any of the following displays:

- **RL Queues:** Displays the queues used by the RL (Library Device Manager) process
- **RQ Queues:** Displays the queues used by the RQ (Queue Manager) process
- **Summary:** Summary of buffer usage by the RQ process

Additionally, rd supports the following functions:

- **Exit:** Exits the rd utility
- **Help:** Displays an online reference for using rd
- **Maintenance:** Toggles maintenance mode on and off
- **Version:** Identifies the version number of the rd program

When you execute the rd utility, it displays a main menu with a list of available commands (see the figure on page 32-2). After you select a command, rd performs the associated function. Following completion of a command (other than Exit), the utility returns to the main menu.



**Main menu of the rd utility**

## Format

rd
















### Keystroke commands

The rd utility does not follow the standard UNIX command format; that is, a line consisting of a command followed by parameters, separated by delimiters. Instead, rd uses single *keystroke commands*, some of which are preceded by an optional number. The meaning of a keystroke command depends on the context in which it is used. However, most of the keystroke commands perform the same function throughout rd.







You can enter an alphabetic command in either uppercase or lowercase letters. Note that the command is not displayed on the screen as you type. If you type an invalid command, the display beeps and no action is taken.

Table 32-1 lists and describes the general types of keystroke commands accepted by the rd utility.

**Table 32-1: Keystroke commands accepted by the rd utility**

Command	Description
Alphabetic key	Used to select a menu choice on the rd main menu. Also used in conjunction with numeric keys in certain commands.
Numeric key	Used to supply a value when it precedes one of the following commands: t, r, c,  ,  ,  , or  .
	Updates the data on the screen immediately whenever rd is in a periodic screen update mode (RL Queues display, RQ Queues display, or Queues Summary display).
[number of seconds] t	<p>Changes the elapsed time between screen updates whenever rd is in a periodic screen update mode. For example:</p> <p>10t</p> <p>causes the data on the screen to update every 10 seconds.</p>
[num_rows   num_cols]  or  -P [num_rows   num_cols]  or  -N [num_rows   num_cols]  or  -B [num_rows   num_cols]  or  -F	<p>Moves the “window” that rd displays if the amount of data to be displayed exceeds the size of the terminal screen.</p> <p>This command also moves the reverse-video selection bar  that appears when you press  on either the RL or RQ display screens. This bar allows you to select a buffer to display. You can move the bar one row or column at a time using either the arrow or control keys alone (the num_rows   num_cols option is not available for the selection bar).</p> <p>The rd utility supports VT-100 and VT-52 style arrow keys. For terminals that do not support either of these styles of arrow keys, rd allows you to use the designated control key equivalents.</p>

**Table 32-1: Keystroke commands accepted by the rd utility (continued)**

Command	Description
	Generally takes you to the next lower level of operation, further down the path you are following. For example, if one volume ID in a set is highlighted, pressing  displays the corresponding volume buffer.
 Bksp or 	Generally returns you to the previous level of operation, back up the path from which you came. For example, if a volume or request buffer is currently displayed, pressing  Bksp or  returns you to the previous screen, where you can either select another volume or request buffer, or return to an even earlier screen.

## Main menu commands



The rd main menu shown in the figure on page 32-2 is displayed when you type rd from the StorHouse operating system (UNIX) command prompt and press . The main menu lists all the functions available with the rd utility as well as their corresponding keystroke commands. Select a menu choice by typing the command that corresponds to the function you want to perform (do not press  following the command).

Table 32-2 lists and describes the commands in the main menu.

**Table 32-2: Keystroke commands in the rd main menu**


Command	Function	Description
E	Exit	Clears the screen, terminates rd execution, and returns control to the operating system.
H	Help	Allows you to view the online reference for using the rd utility. The information is organized as a set of topics, each with sub-topics. <ul style="list-style-type: none"> <li>■ To view the help on a particular topic, type the name of the topic and press .</li> </ul>

Table 32-2: Keystroke commands in the rd main menu (continued)

Command	Function	Description
		<ul style="list-style-type: none"> <li>■ To display a list of the available top-level topics, type a "?" and press <b>Enter</b>.</li> <li>■ To exit Help, press <b>Enter</b> at the topic prompt.</li> </ul>
L	RL Queues	<p>Periodically examines the internal RL Queues and displays them on the screen. One line is displayed for every library device in the system. The first column shows the device name of the library device. The remaining columns show the volume IDs of the volumes in the library device drives.</p> <p>During the display, you can enter any of the keystroke commands defined in Table 32-3, "Keystroke commands for the RL and RQ Queues" on page 32-7.</p>
M	Toggle maintenance mode	<p>Toggles the maintenance mode flag and displays its new value. The display freezes until you press either <b>←Bksp</b> or <b>Delete</b>.</p> <p>When maintenance mode is on, certain commands display more information than usual. This information is generally not useful.</p>
Q	RQ Queues	<p>Periodically examines the internal RQ Queues and displays them on the screen. The top row of the display shows the volume IDs of all active volume buffers. In maintenance mode, even "idle" volume buffers are displayed. Subsequent rows show the types of all active request buffers, one row for every request buffer priority.</p> <p>During the display, you can enter any of the keystroke commands defined in Table 32-3, "Keystroke commands for the RL and RQ Queues" on page 32-7.</p> <p>A sample RQ Queues Display window is shown under "Sample output" on page 32-10.</p>
S	Summary of queues	<p>Displays a detailed summary of buffer usage in the RQ process. Five different summaries are displayed:</p> <ul style="list-style-type: none"> <li>■ The Buffer Summary shows the number of total buffers, vol (volume) buffers, req (request) buffers, and free buffers.</li> </ul>

Table 32-2: Keystroke commands in the rd main menu (continued)

Command	Function	Description
		<ul style="list-style-type: none"> <li>■ The Active Volume Summary shows the number of volumes in a drive, in a slot, and on the shelf, as well as the number of volumes that are moving between a library and a shelf.</li> <li>■ The Label Summary shows the number of various types of label requests. The StorHouse software records an internal label on the media used by the system. A label request is generated each time a blank volume is migrated into the free pool, data is written to a volume for the first time, and a volume is erased.</li> <li>■ The Request Summary shows the number of each type of request. A request is a system action in response to and in support of a user command. For example, when a user opens or moves a file, an internal request is generated.</li> <li>■ The Transfer Summary shows the number of total transfers, active transfers, suspended transfers, and waiting transfers, and breaks out the waiting transfers by reason for the wait. These values are also broken out by low and high priority requests. Transfers include all user requests to copy data from one location on a StorHouse system to another, for example, from the network to the performance buffer, from the performance buffer to permanent storage, and so on.</li> </ul> <p>Values are periodically refreshed.</p> <p>A sample Queues Summary Display window is shown under "Sample output" on page 32-11.</p>
V	Version of rd program	Displays the version of rd being executed. The display freezes until you press either <b>←Bksp</b> or <b>Delete</b> .

## RL and RQ Queues display commands

In addition to providing access to main menu functions, keystroke commands enable you to perform various functions during display of the RL and RQ Queues. With the exception of the *s*, *d*, and *v* commands, which are available only for the RQ Queues, the following keystroke commands are available for both displays.

**Table 32-3: Keystroke commands for the RL and RQ Queues**

Command	Description
<span>Spacebar</span>	Updates screen data immediately.
<code>[number of seconds] t</code>	Changes the elapsed time between screen updates. For example:  10t  causes the data on the screen to update every 10 seconds.
<code>[num_rows   num_cols]</code> <span>↑</span> or <span>Ctrl</span> -P <code>[num_rows   num_cols]</code> <span>↓</span> or <span>Ctrl</span> -N <code>[num_rows   num_cols]</code> <span>→</span> or <span>Ctrl</span> -B <code>[num_rows   num_cols]</code> <span>←</span> or <span>Ctrl</span> -F	<p>If the amount of data to be displayed exceeds the size of the terminal screen, rd uses a "window" to display a portion of the data. You can move the window vertically or horizontally by entering the number of rows or columns to be moved, followed by one of the arrow keys (<span>↑</span>, <span>↓</span>, <span>→</span>, or <span>←</span>). If you do not enter a number, the window shifts one row or column.</p> <p>When you press <span>Enter↵</span> on either the RL or RQ display screens, a reverse-video selection bar <span>■</span> appears. This bar allows you to select a buffer to display. You can move the bar one row or column at a time using either the arrow or control keys alone (the <code>num_rows</code>   <code>num_cols</code> option is not available for the selection bar).</p>
<code>[row_num] r</code>	To move the upper-left corner of the window to a particular row, enter the row number, followed by the letter <i>r</i> . If you do not specify a number, the window moves to row 1.

**Table 32-3: Keystroke commands for the RL and RQ Queues (continued)**






Command	Description
<code>[col_num] c</code>	To move the upper-left corner of the window to a particular column, enter the column number, followed by the letter c. If you do not specify a number, the window moves to column 1.
<code>Enter ↵</code>	<p>Freezes the display. Pressing this key also causes a reverse-video selection bar to appear on the screen.</p> <ul style="list-style-type: none"> <li>■ To view detailed information for a specific volume or request buffer, use the arrow keys (<code>↑</code>, <code>↓</code>, <code>→</code>, or <code>←</code>) to position the bar over the desired volume ID (or request buffer, for the RQ Queues). Then press <code>Enter ↵</code>.</li> <li>■ To update the values on the Volume Buffer or Request Buffer display, press <code>Enter ↵</code>.</li> <li>■ To return to the frozen display, press either <code>←Bksp</code> or <code>Delete</code>. Then select another volume or request buffer to display, or press <code>←Bksp</code> or <code>Delete</code> to unfreeze the display and return to monitoring mode.</li> </ul>
<code>←Bksp</code> or <code>Delete</code>	Terminates system monitoring performed by the RL and RQ Queues.
s	Displays the source volume buffer, if any, from the Request Buffer display of the RQ Queues.
d	Displays the destination volume buffer, if any, from the Request Buffer display of the RQ Queues.
v	Displays either the source or destination volume buffer, if any, from the Request Buffer display of the RQ Queues.




## Line editing commands

Whenever rd is in line input mode, you can perform line editing tasks using the following keys:

**Table 32-4: Line editing commands**

Command	Description
 or 	Erases the last character entered.
 -U	Erases the entire line.
 or  -P	Retrieves the last line entered, which you can then add to or edit.

## Refreshing the display

Whenever rd is in input mode, you can use -R to refresh the display.

## Sample output

The following window appears when you select the Q (RQ Queues) command on the rd main menu:

```
RESOURCE MANAGER DISPLAY PROCESS -- RQ QUEUES DISPLAY 11:16:26
```

```
          #1      #2      #3      #4      #5      #6      #7  
VOL      :  C10001CX
```

```
REQ 07:
```

```
REQ 06:
```

```
REQ 05:
```

```
REQ 04:
```

```
REQ 03:
```

```
REQ 02:
```

```
REQ 01:
```

```
REQ 00:
```

```
Enter RETURN, BACKSPACE/DELETE, or a Screen-Positioning Command.
```

**RQ Queues Display window**

The following window appears when you select the S (Summary of Queues) command on the rd main menu:

```

RESOURCE MANAGER DISPLAY PROCESS -- QUEUES SUMMARY DISPLAY 11:16:26
Buffer Summary                                Active Volume Summary
  Total:                                1000                In a Drive:                0
  Volume:                                7 (7 idle)           In a Slot:                0
  Request:                               0                   On the Shelf:             0
  Free:                                 993                   Moving:                   0
Label Summary                                Request Summary
  Initialize:                            0                   Transfer Extent:          0
  Relabel:                               0                   Remove Extent:            0
  Erase:                                 0                   Recover Volume:           0
Transfer Summary                             Label Volume:           0
  All Low High                             Move Volume:             0
  Total:                                0 0 0                 Export Volume:            0
  Transferring:                         0 0 0                 Import Volume:            0
  Suspended:                           0 0 0                 Remove Volume:            0
  Waiting:                             0 0 0                 Clean Drive:              0
  Operator:                             0 0 0                 Reload Drive:             0
  Drive:                               0 0 0                 Unload Drive:             0
  Volume:                              0 0 0                 Reset Library:            0
  System:                              0 0 0                 Device Status:            0
Enter BACKSPACE/DELETE.

```

**Queues Summary Display window**



## **rebuild\_disk utility**

The `rebuild_disk` utility rebuilds partitions that you previously backed up (using the `make_bkups` utility) on a specified StorHouse system disk. Use this utility to install the StorHouse operating system on a new drive following the replacement of a broken disk. Note that `rebuild_disk` destroys any existing data on the destination device.

After you execute the `rebuild_disk` utility, either reboot the StorHouse server if you replaced the os disk or exit to multi-user mode if you replaced any other disk.

### **Before you use rebuild\_disk**

Before you run the `rebuild_disk` utility, you must make the target disk usable (that is, you must format, partition, and label the disk using the UNIX format utility). In addition, ensure that:

- An entry exists in the `/etc/vfstab` (file system table) file for the destination drive
- The source drive is accessible (that is, the necessary source partitions are mounted)
- No disk partitions are mounted for the destination drive

The `rebuild_disk` utility uses the boot drive as the source of the software to be installed on the target drive. If you are using a hot spare drive to replace the os or

bkup disk, edit the `/etc/vfstab` file and specify the device name for the hot spare where the defective disk is referenced before you execute this utility.

## How to execute rebuild\_disk

You must log in to the StorHouse server operating system as superuser (using the root account) to execute the `rebuild_disk` utility. From the root directory, either specify the complete path name when you type the `rebuild_disk` command, or change to the `/filetek/bin` directory then execute `rebuild_disk`.

In addition, it is highly recommended that you execute this utility in single-user mode (`b -sw` or `reboot --sw`). If you run `rebuild_disk` in multi-user mode and/or redirect stdout/stderr, you run the risk of copying a file system that is in flux due to activity by other users.

## Format

`/filetek/bin/rebuild_disk {os | bkup | pri | sec}`

Argument	Description
os	Rebuilds the operating system disk from the boot disk (including <code>/filetek</code> if it resides on the os disk).
bkup	Rebuilds the backup disk from the boot disk (including <code>/filetek2</code> if it resides on the bkup disk).
pri	Rebuilds the primary disk ( <code>/filetek</code> ) from the secondary disk ( <code>/filetek2</code> ).
sec	Rebuilds the secondary disk ( <code>/filetek2</code> ) from the primary disk ( <code>/filetek</code> ).

## recds utility

The recds (records) utility produces a short form dump for a subset of user log record types for one or more days. You can use the output of this utility to monitor StorHouse activity. In addition, you can identify problems with the system by searching on the phrase retcode in the recds output file.

You can dump records for the current date, the prior day's date, a selected date, or a range of dates. You can also specify which record types appear in the output file by using the select parameter. (See the *User Log Format*, publication number 900028, for more information about user log data record types.) If you do not specify record types, information for the following record types is dumped by default: 9, 10, 11, 12, 13, 20, 21, and 22. If multiple user logs exist for the specified date, the recds utility dumps the appropriate records from all the logs.

In order to dump records for a specific date or range of dates from the user log, the StorHouse NEWLOG /USER command must have been issued subsequent to the specified time period. The NEWLOG command closes the current temporary log file, opens a new one, and writes the closed file to StorHouse user storage. The recds utility then accesses the copy of the log file that resides in user storage. (See the *Command Language Reference Manual*, publication number 900005, for more information on the NEWLOG command.)

To ensure that FileTek customer support representatives can run the recds utility whenever necessary, support personnel typically schedule the NEWLOG command to run daily (shortly before midnight) on every installed StorHouse system.

## Format

```
recds "{today | yesterday | week | month |
first=yyyyjjj},[last=yyyyjjj],[select=(n:n:n)]"
```

You can use either the *yyyyjjj* or *yyyymmdd* format when specifying a date.

**Note:** You can also execute the recds utility in interactive mode, and by using the RUN command from the StorHouse command prompt (?).

Argument	Description
today	Specifies that the user log with the current date is the first log file to process.
yesterday	Specifies that the user log with the prior calendar day's date is the first log file to process.
week	Specifies that the first user log file to process is dated one week (7 calendar days) prior to the current date and the last user log file to process is yesterday's date.
month	Specifies that the first user log file to process is dated one month (30 calendar days) prior to the current date and the last user log file to process is yesterday's date.
first=yyyyjjj or first=yyyymmdd	Calendar year and date of the first user log file to process if other than today, yesterday, week, or month. If you use the Julian date <sup>a</sup> format (yyyyjjj), specify a number from 001 to 365 or 366 following the year.



Argument	Description
last=yyyyjjj or last=yyyymmdd	<p>Calendar year and date of the last user log file to process when used in conjunction with the “first” argument. If you omit this argument, the default is last=first.</p> <p>If you use the Julian date format, specify a number from 001 to 365 or 366 following the year.</p>
select=(n:n:n)	<p>Specifies the type of user log records to process. You can specify any number of record types. If you omit this argument, the default is record types 9:10:11:12:13:20:21:22, where:</p> <p>9=commands, 10=open, 11=close, 12=dismount, 13=movement, 20=mount, 21=message, 22=copy</p> <p><b>Note:</b> The recds utility ignores console commands even if you select record type 9.</p>

- a. The Julian date is a number assigned in sequence to each day of the year, starting with Julian date 001 for January 1 and ending with Julian date 365 (or 366 for a leap year) for December 31.

## Examples

This section presents three examples of how to dump record types from the user log.

### Example 1

The following command dumps all supported record types from the user log with the prior calendar day's date:

```
recds "yesterday"
```

### Example 2

The following command dumps record types 10, 13, and 20 from all user logs dated January 1, 1999 through March 28, 1999:

```
recds "first=1999001,last=1999087,select=(10:13:20)"
```

### Example 3

The following command dumps record type 9 from the user log dated March 13, 1999:

```
recds "first=1999072,select=(9)"
```

## Sample output

The following recds command:

```
recds "yesterday"
```

results in the sample output below (the output is not shown in its entirety to conserve space):

```
recds (4.2a)
```

```
DATE-RANGE is: 1999192 thru 1999192
```

```
(cmd) start time=10-JUL-1999:23:59:00 end time=10-JUL-1999:23:59:00  
NEWLOG
```

```
(cmd) start time=10-JUL-1999:23:59:01 end time=10-JUL-1999:23:59:01  
mon /all /int=0
```

```
(cmd) start time=10-JUL-1999:23:59:01 end time=10-JUL-1999:23:59:02  
sh dev *
```

```
(close) U1999191235900 SERVICE end=10-JUL-1999:23:59:02  
PUT SEQ ufid=2102.1080963 nropns=0 size(kb)=2180 et(sec)=2
```

```
(cmd) start time=10-JUL-1999:23:59:00 end time=10-JUL-1999:23:59:02  
PUT/ASCII "U1999191235900"/GROU=SERVICE/NOEDC/  
FSET=USERLOG/VSET=USERLOG
```

```
(cmd) start time=10-JUL-1999:23:59:05 end time=10-JUL-1999:23:59:05  
mon /all /int=0
```

```
(cmd) start time=10-JUL-1999:23:59:05 end time=10-JUL-1999:23:59:06  
sh dev *
```

```
(cmd) start time=10-JUL-1999:23:59:14 end time=10-JUL-1999:23:59:14  
mon /all /int=0
```

```
(cmd) start time=10-JUL-1999:23:59:14 end time=10-JUL-1999:23:59:14  
sh dev *
```

```
(cmd) start time=10-JUL-1999:23:59:39 end time=10-JUL-1999:23:59:40  
mon /all /int=0
```

```
(cmd) start time=10-JUL-1999:23:59:40 end time=10-JUL-1999:23:59:40  
sh dev *
```

```
(cmd) start time=11-JUL-1999:00:00:00 end time=11-JUL-1999:00:00:00  
BACKUP
```

```

(cmd) start time=11-JUL-1999:00:00:01 end time=11-JUL-1999:00:00:01
mon /all /int=0
(cmd) start time=11-JUL-1999:00:00:01 end time=11-JUL-1999:00:00:01
sh dev *
(cmd) start time=11-JUL-1999:00:00:05 end time=11-JUL-1999:00:00:05
mon /all /int=0
(cmd) start time=11-JUL-1999:00:00:05 end time=11-JUL-1999:00:00:06
sh dev *
(mount) time=11-JUL-1999:00:00:11 vol=OEB2E0B18FE:A vset=ERASE
sdev=L01D00 ufid=2102.1080905
(cmd) start time=11-JUL-1999:00:00:14 end time=11-JUL-1999:00:00:14
mon /all /int=0
(cmd) start time=11-JUL-1999:00:00:14 end time=11-JUL-1999:00:00:14
sh dev *
(mount) time=11-JUL-1999:00:00:15 vol=OEB2DD8A6F6:A vset=USERLOG
sdev=L01D01 ufid=2102.1080963
(mount) time=11-JUL-1999:00:00:20 vol=OEB2DD89993:B vset=SOURCE
sdev=L01D02 ufid=2102.1080907
(dism) time=11-JUL-1999:00:00:24 vol=OEB2E0B18FE:A vset=ERASE
(mount) time=11-JUL-1999:00:00:24 vol=OEBHIT00003:B vset=ES9000
art time=11-JUL-1999:00:02:05 end time=11-JUL-1999:00:02:05
mon /all /int=0
(cmd) start time=11-JUL-1999:00:02:05 end time=11-JUL-1999:00:02:06
sh dev *
(cmd) start time=11-JUL-1999:00:02:14 end time=11-JUL-1999:00:02:14
mon /all /int=0
(cmd) start time=11-JUL-1999:00:02:14 end time=11-JUL-1999:00:02:14
sh dev *
(cmd) start time=11-JUL-1999:00:02:39 end time=11-JUL-1999:00:02:40
mon /all /int=0
(cmd) start time=11-JUL-1999:00:02:40 end time=11-JUL-1999:00:02:40
sh dev *
(cmd) start time=11-JUL-1999:00:03:01 end time=11-JUL-1999:00:03:01
mon /all /int=0
(cmd) start time=11-JUL-1999:00:03:01 end time=11-JUL-1999:00:03:02
sh dev *
(cmd) start time=11-JUL-1999:00:03:05 end time=11-JUL-1999:00:03:05
mon /all /int=0

```

## Possible error conditions and resolutions

If you execute the recds command and the utility returns the following message with no data:

DATE-RANGE is: 1999101 thru 1999101

the probable cause is one of the following:

- NEWLOG was not executed subsequent to the specified date or range of dates.

Resolution: Sign on to StorHouse using the system account and issue the following command at the StorHouse command prompt (?):

NEWLOG /USER

then sign off from StorHouse and execute the recds command again.

- No data records of the type specified exist for the designated time period.

Resolution: No action is required. Specify a different set of record types, if desired.



## set\_timezone utility

The `set_timezone` utility designates the time zone in which a StorHouse system will be operating so that all time-related data (such as timestamps in the StorHouse user and administration logs) will be accurate.

You can set the time zone for a new StorHouse system during installation of the Solaris operating system on the StorHouse server. If you want to change the time zone for an existing StorHouse system (for example, a customer moves the system to a location in a different time zone), run the `set_timezone` utility to reset the time zone to the correct zone.

You must log in to the StorHouse server operating system as superuser (using the root account) to set the time zone for a StorHouse system. The time zone setting does not take effect until you reboot the system.

### Selecting a time zone

The UNIX operating system provides a set of files that correspond to each of the 24 standard time zones as well as a number of time zones unique to various countries. These files, which are listed in alphabetical order by column below, are located in the `/usr/share/lib/zoneinfo` directory. When you use the `set_timezone` utility, the value you specify for the `Zone` argument (and the `Subzone` argument, if required) corresponds to one of these UNIX files:

Australia *	Factory	Libya	Portugal
Brazil *	GB	MET	ROC

CET	GB-Eire	MST	ROK
CST6CDT	GMT	MST7MDT	Singapore
Canada *	Greenwich	Mexico *	Turkey
Chile *	HST	Mideast *	UCT
Cuba	Hongkong	NZ	US *
EET	Iceland	NZ-CHAT	UTC
EST	Iran	Navajo	Universal
EST5EDT	Israel	PRC	W-SU
Egypt	Jamaica	PST8PDT	WET
Eire	Japan	Poland	Zulu
Etc *	Kwajalein		

An asterisk next to a time zone in the list signifies that the time zone is made up of subzones. For example, the US time zone is composed of subzones such as Eastern, Central, and Pacific. The Etc zone contains subzones for various offsets from Greenwich mean time (GMT), for example, GMT -02:00 and GMT +07:00. When you issue the set\_timezone command for such a time zone, you must designate the appropriate subzone within the time zone.



## Format

set\_timezone [*Zone* [*Subzone*]]

Argument	Description
[no argument]	Displays a list of available time zones from which you can choose (see Sample 1 on page 35-4).
<i>Zone</i>	<p>Time zone in which the StorHouse system will be operating. The name you specify must begin with a capital letter.</p> <p>If you specify only a time zone for a geographic area that has multiple subzones, the set_timezone utility displays a list of subzones from which you must choose (see Sample 2 on page 35-5).</p>
<i>Subzone</i>	Subzone in which the StorHouse system will be operating. The name you specify must begin with a capital letter.

## Examples

This section presents two examples of how to run the set\_timezone utility.

### Example 1

The following command sets the time zone for a StorHouse system to the Pacific subzone within the US time zone:

```
set_timezone US Pacific
```

## Example 2

The following command sets the time zone for a StorHouse system to Greenwich mean time (GMT):

```
set_timezone GMT
```

## Sample output

The following sections illustrate the sample output that results from executing three different set\_timezone commands.

### Sample 1

The following sample output results when you issue the set\_timezone command with no arguments:

Usage: set\_timezone [<zone> [<subzone>]]

Available zones:

Australia *	Factory	Libya	Portugal
Brazil *	GB	MET	ROC
CET	GB-Eire	MST	ROK
CST6CDT	GMT	MST7MDT	Singapore
Canada *	Greenwich	Mexico *	Turkey
Chile *	HST	Mideast *	UCT
Cuba	Hongkong	NZ	US *
EET	Iceland	NZ-CHAT	UTC

EST	Iran	Navajo	Universal
EST5EDT	Israel	PRC	W-SU
Egypt	Jamaica	PST8PDT	WET
Eire	Japan	Poland	Zulu
Etc *	Kwajalein		

Note: Zones marked with a "\*" contain subzones.

Use "set\_timezone <zone>" to see its subzones.

## Sample 2

The following sample output results when you issue the set\_timezone command with a value of US for the zone argument and don't specify a subzone:

Zone US requires a subzone.

Available subzones in the US zone:

Alaska	Central	Hawaii	Pacific
Aleutian	East-Indiana	Michigan	Pacific-New
Arizona	Eastern	Mountain	Samoa

## Sample 3

The following sample output results when you issue the set\_timezone command with a value of US for the zone argument and Pacific for the subzone argument:

The time zone has been changed to US/Pacific.  
It is now Thu Oct 28 13:55:34 PST 1999.  
Please reboot to complete the change.



## shelf utility

The shelf utility monitors I/O station activity for each optical library in a StorHouse system for a specified day or date range. The utility opens each user log file associated with the specified time period and reads all data records with a record code number of 13 (volume movement). It then reports the number of volumes that were moved into and out of each library in a StorHouse system per hour during the specified time period.

A UNIX cron job runs a StorHouse procedure called daily at 12:15 a.m. on every customer's StorHouse system. This procedure executes a number of internal StorHouse utilities, including the shelf utility, and logs the output from shelf in the customer's `hw_log.yymmdd` file. The output is identical to the shelf report that is produced when you run the shelf utility (refer to the sample report on page 36-3). The CS server at FileTek headquarters polls customer sites, retrieves the `hw_log` report, and stores it on the FileTek CS server in `~ps/sites/site_name`, where *site\_name* is the name of the customer site.

### Format

```
shelf {today | yesterday | week | month | first=yyyyjjj},[last=yyyyjjj]
```

You can use either the *yyyyjjj* or *yyyymmdd* format when specifying a date.

**Note:** You can also execute the shelf utility in interactive mode.

Argument	Description
today	Specifies that the user log with the current date is the first log file to process.
yesterday	Specifies that the user log with the prior calendar day's date is the first log file to process.
week	Specifies that the first user log file to process is dated one week (7 calendar days) prior to the current date and the last user log file to process is yesterday's date.
month	Specifies that the first user log file to process is dated one month (30 calendar days) prior to the current date and the last user log file to process is yesterday's date.
first=yyyymmdd or first=yyyymmdd	Calendar year and date of the first user log file to process if other than today, yesterday, week, or month. If you use the Julian date <sup>a</sup> format (yyyymmdd), specify a number from 001 to 365 or 366 following the year.
last=yyyymmdd or last=yyyymmdd	Calendar year and date of the last user log file to process when used in conjunction with the "first" argument. If you omit this argument, the default is last=first.  If you use the Julian date format, specify a number from 001 to 365 or 366 following the year.

- a. The Julian date is a number assigned in sequence to each day of the year, starting with Julian date 001 for January 1 and ending with Julian date 365 (or 366 for a leap year) for December 31.

## Sample report

The following sample shelf report results when you execute the shelf utility with the yesterday argument. The first section of the report lists the total number of shelf volumes mounted for each library device, as well as the total for all libraries in the StorHouse system. The second section of the report identifies the number of input volumes and output volumes mounted per hour for each library device.

SHELF (5.1)

DATE-RANGE is: 1999308 thru 1999308

Total IOSTATION activity by library:

This number is the → L00 = 14  
sum of the input  
volumes (10) and  
output volumes (4)  
for library L00.

L01 = 1  
L02 = 0

total for all libraries = 15

IOSTATION activity by hour by library:

	in/out		
	L00	L01	L02
0000-0059	00/00	00/00	00/00
0100-0159	00/00	00/00	00/00
0200-0259	00/00	00/00	00/00
0300-0359	00/00	00/00	00/00
0400-0459	00/00	00/00	00/00
0500-0559	00/00	00/00	00/00
0600-0659	00/00	00/00	00/00
0700-0759	00/00	00/00	00/00
0800-0859	00/00	00/00	00/00
0900-0959	00/00	00/00	00/00
1000-1059	01/01	00/00	00/00
1100-1159	06/00	00/00	00/00
1200-1259	03/01	00/00	00/00
1300-1359	00/00	00/00	00/00
1400-1459	00/00	01/00	00/00
1500-1559	00/00	00/00	00/00

During the time → period 10 a.m. to 10:59 a.m., one volume was moved into and one volume was moved out of library L00.

1600-1659	00/00	00/00	00/00
1700-1759	00/00	00/00	00/00
1800-1859	00/00	00/00	00/00
1900-1959	00/02	00/00	00/00
2000-2059	00/00	00/00	00/00
2100-2159	00/00	00/00	00/00
2200-2259	00/00	00/00	00/00
2300-2359	00/00	00/00	00/00



## startsm utility

The startsm utility starts the StorHouse software manually (normally StorHouse system power-up invokes startsm).

The startsm utility does the following:

- Executes the cxclean utility, which cleans up shared system resources (shared memory, locks, and mailboxes) used by StorHouse/SM.
- Determines whether StorHouse/RM is installed on the system. If so, executes the cleanipcs utility, which cleans up shared system resources used by that software.
- Creates a “save” directory, which contains miscellaneous files from the previous execution of startsm.
- Installs any pending asd fixes provided the JUSTDOIT file exists in the SUUCP\_SPOOL/asd directory.
- Starts the qchc (Call Home), bmon\_nvm, bmon\_dev, and bmon\_ctcs processes. (The bmon\_nvm process monitors the battery status of non-volatile memory (NVM) boards, the bmon\_dev process monitors the UNIX system log for device errors, and the bmon\_ctcs process monitors the serial port of a Polaris model 8900 Sbus board.) These processes continue to run after the StorHouse software ultimately stops. A subsequent startsm will stop them, then restart them.

- Creates a log file, `$$SMD_GENERAL/sm.log`, in which the first line is the date and time the StorHouse software was started (this information is used by the timeup utility).
- Starts the System Control (SC) process, which controls the startup and shutdown of the StorHouse software.
- Starts the diskfree utility, which monitors the minimum amount of available disk space defined for each file system used by the StorHouse operating system. If the StorHouse software crashes, a subsequent startsm will stop diskfree, then restart it.
- Creates the file `$$SMD_GENERAL/stoppids`, which contains a list of processes (by process ID) that should be killed by stopsm when the system is shut down.
- Executes the site-specific startsm script `$HOME/startsm.local` after the StorHouse software is up if the script exists and has execute permission. The current directory for the startsm.local script is `$$SMD_GENERAL` and its standard output (stdout) and standard error (stderr) is the startsm.log file.

The UNIX command prompt does not return immediately after you issue the startsm command, nor is the StorHouse software necessarily up when the prompt returns.

Only one startsm process can run at a time.

## Format

You can execute startsm interactively or as part of the boot process by using a shell script.

- To invoke the startsm utility interactively from the UNIX prompt, enter the startsm command with no options.
- To invoke the startsm utility as part of the boot process, use the following command format:

startsm {-b}

Option	Description
-b	Specifies that startsm is being invoked as part of the boot process. In order to use this option, the \$SMD_GENERAL environment variable must be defined on the StorHouse system and its value must represent a valid directory path. In addition, startsm checks for the existence of the smc file, which is used by StorHouse to control system startup, in the \$SMD_PRIMARY directory.



## sthtest utility

The sthtest utility verifies that you successfully installed a StorHouse/RM system.

The sthtest utility is comprised of two UNIX scripts or *procedures*: one to set up a test database table and validate the StorHouse/RM software (sth\_val\_prep) and one to remove the test data created by sth\_val\_prep following a successful StorHouse validation (sth\_val\_del). These procedures are described in the following sections.

### How the sth\_val\_prep procedure works

The sth\_val\_prep procedure runs in interactive mode. After you execute sth\_val\_prep from the StorHouse operating system (UNIX) prompt, it displays a series of instructions, or prompts, to which you respond. As the procedure runs, it displays status messages to inform you of its progress.

The following sections describe the input to and output from the sth\_val\_prep procedure, as well as the steps that sth\_val\_prep performs. In addition, these sections describe how to use the sth\_val\_prep procedure, including preliminary tasks you must complete.

## Input to sth\_val\_prep

The following predefined parameter files serve as input to the sth\_val\_prep procedure:

File name	Description
sth_val_prep.ftp	Static FTP Data Loader file. This file contains both the control statements necessary to load data into a StorHouse user table and the actual data records to be loaded. The FileTek FTP Data Loader uses this file to create a test tablespace named SM2 and a database table named PUBLICATIONS, and to load data into the test table.
sth_val_prep.ctrl	Static control file. This file contains the contents of a table named PUBLICATIONS, which represents the expected output from a sth_val_prep execution. The sth_val_prep procedure compares the data in the spool file it creates (sth_val_prep.spool) with the data in the control file to ensure that the contents are identical.

The input files are located in the same directory as the procedure files (/filetek/operator/sthtest). The sth\_val\_prep procedure uses these files each time you execute it—you never need to modify the files.

## Steps performed by sth\_val\_prep

The following table describes the steps performed by the sth\_val\_prep procedure.

**Steps performed by the sth\_val\_prep procedure**

Step	Description
1	Creates a StorHouse account with the ID you specify and assigns database administrator (DBA) privilege to the account.
2	Creates a StorHouse database with the name you specify.
3	Saves the account ID and database name you specify in a temporary file, sth_val_prep.data, for subsequent use by the sth_val_del procedure in removing the test data.

**Steps performed by the sth\_val\_prep procedure (continued)**

Step	Description
4	Creates a user tablespace named SM2 and a test table named PUBLICATIONS, and loads data into the table using the FileTek FTP Data Loader and its input file (sth_val_prep.ftp). The data is stored in the StorHouse performance buffer.
5	Checks the SYSADM.SYSTABLES system table to ensure that an entry exists for the PUBLICATIONS test table.
6	Performs a SELECT operation on the test table and directs the output to a spool file, sth_val_prep.spool.
7	Compares the results in sth_val_prep.spool with those in the control file, sth_val_prep.ctrl, which contains the expected output.
8	Displays the message "StorHouse validation procedure completed successfully (remember to execute sth_val_del to clean everything up)" if the results of the comparison are identical. Otherwise, sth_val_prep displays the message "Control and spool files different."

## Before you use sth\_val\_prep

Before you execute the sth\_val\_prep procedure, you must create the spool file identified in step 6 in the preceding table. The size of the file is not important; it can be zero or more bytes.

To create the spool file, type the following command from the /filetek/operator/sthtest directory:

```
touch sth_val_prep.spool
```

If you neglect to create the spool file prior to executing sth\_val\_prep, the sth\_val\_prep procedure creates the file and assigns root ownership to it. Unfortunately, this prevents you from accessing it with the operator account and causes sth\_val\_prep to fail.

When you execute the sth\_val\_prep procedure, you must specify the name of an existing volume set, file set, and group name for the StorHouse account you create. Therefore, prior to running sth\_val\_prep, ensure that they already exist on the StorHouse system.

## Format for sth\_val\_prep

Unlike most of the other StorHouse utilities or procedures described in this manual, you can't execute sth\_val\_prep from the operator's home directory (/filetek/operator). You must execute this procedure from /filetek/operator/sthtest.

Type the following command from the directory path specified above and press **Enter** (make sure you type ./ preceding the command to instruct UNIX to look for the sth\_val\_prep procedure in the current directory):

```
./sth_val_prep
```

The sth\_val\_prep procedure displays a series of prompts, which are described in the following table. The procedure supplies default values for all but the last argument in the table. The default value for each prompt is enclosed in parentheses following the prompt. To accept the default value for a given prompt, press **Enter** without specifying a value. To specify a value other than the default value, type the desired value and press **Enter**.



**Prompts displayed by the sth\_val\_prep procedure**

Argument	Description	Default
account ID	New StorHouse account ID, which serves as the user table owner for the test table. If the default account ID supplied by sth_val_prep already exists on the StorHouse system you're testing, you must specify a different account ID.	STHDBA
volume set	Name of the volume set to which all table, hash index, and value index segments associated with the test user table are assigned. If you accept the default volume set name, it must already exist on the StorHouse system you're testing.	SYSTEM
file set	Name of the file set to which all table, hash index, and value index segments associated with the test user table are assigned. If you accept the default file set name, it must already exist on the StorHouse system you're testing.	SERVICE
group	Name of the file access group to which all table, hash index, and value index segments associated with the test user table are assigned. If you accept the default group name, it must already exist on the StorHouse system you're testing.	STH
database name	Name you want to assign to the test database. The name must not already exist in the current StorHouse UNIX file system.	–

## Files created by sth\_val\_prep

The sth\_val\_prep procedure creates the output files described in the following table. (In the case of sth\_val\_prep.spool, the procedure just adds the contents of the database it creates to the empty file you created in “Before you use sth\_val\_prep” on page 38-3.)

File name	Data provided
sth_val_prep.spool	Spool file that contains the output from issuing the SQL SELECT statement on the test database. The sth_val_prep procedure compares the contents of this file with the contents of the static control file (sth_val_prep.ctrl) to ensure that they are identical.
sth_val_prep.out	File that contains the output from the execution of the sth_val_prep procedure, including any error messages that resulted from executing sth_val_prep.
sth_val_prep.data	Temporary file in which sth_val_prep stores the StorHouse account ID and the name of the test database you created. The sth_val_del procedure subsequently uses this file to remove the test data.

Upon successful completion, the sth\_val\_prep procedure deletes the first two output files described above (sth\_val\_del deletes the last file in the table). If sth\_val\_prep encounters errors during execution, it saves the output files in /filetek/operator/sthtest.

## Sample output from sth\_val\_prep

The following sample output results when you execute the sth\_val\_prep procedure. In this example, the default account ID (STHDBA) already existed on the StorHouse system so an alternate value was specified.

```

Enter the account id (default = STHDBA):  STHDBA2
Enter the volume set (default = SYSTEM):
Enter the file set (default = SERVICE):
Enter the group (default = STH):          ]
                                           ] When you accept the
                                           ] default value for a
                                           ] prompt, no text
                                           ] appears in the output
                                           ] following the colon in
                                           ] the prompt.

CREATING STHDBA2 ACCOUNT . . .
      ACCOUNT CREATED.

Enter the database name: sthtest

CREATING STHTEST DATABASE . . .
      DATABASE CREATED.

DATA LOADER:  CREATING SM2 TABLE SPACE AND LOADING
PUBLICATIONS TABLE . . .
      TABLE SPACE CREATED AND TABLE LOADED.
+ [ ! -f sth_val_prep.ctrl ]
+ isql -u STHDBA2 -a STHDBA2 filetek:T:alpha2:STHTEST
SET ECHO ON
SPOOL "sth_val_prep.spool"
SELECT * FROM PUBLICATIONS;
SPOOL OFF
EXIT
+ check_return_code 0 SELECT FAILED FOR PUBLICATIONS. y
+ [ 0 -ne 0 ]
+ grep -i error sth_val_prep.out
+ [ ! -f sth_val_prep.spool ]
+ diff sth_val_prep.ctrl sth_val_prep.spool
OUTDIFF=
+ [ 0 -ne 0 ]
+ delete_files
+ rm -f sth_val_prep.spool sth_val_prep.tmp sth_val_prep.out

```

```
+ echo \n  STORHOUSE VALIDATION PROCEDURE COMPLETED  
SUCCESSFULLY (Remember to execute\n    sth_val_del to clean  
everything up).\n
```

```
STORHOUSE VALIDATION PROCEDURE COMPLETED SUCCESSFULLY  
(Remember to execute  
    sth_val_del to clean everything up).
```

## How the sth\_val\_del procedure works

The sth\_val\_del procedure removes the test data created by sth\_val\_prep following a successful StorHouse validation.

### Steps performed by sth\_val\_del

The following table describes the steps performed by the sth\_val\_del procedure.

Step	Description
1	Retrieves the account ID and database name from sth_val_prep.data (a temporary file created by sth_val_prep), then deletes the temporary file.
2	Deletes the account ID you created using sth_val_prep.
3	Drops the test table and tablespace.
4	Removes the test database.

## Format for sth\_val\_del

As with the sth\_val\_prep procedure, you must execute sth\_val\_del from /filetek/operator/sthtest. Type the following command and press **Enter** (↵):

```
./sth_val_del
```

The sth\_val\_del command has no arguments or prompts; it executes immediately.

## File created by sth\_val\_del

The sth\_val\_del procedure creates an output file called sth\_val\_del.out, which contains the output from the execution of the sth\_val\_del procedure. Upon successful completion, the sth\_val\_del procedure deletes the output file. If sth\_val\_del encountered errors during execution, it saves the output file in /filetek/operator/sthtest.

## Sample output from sth\_val\_del

The following sample output results when you execute the sth\_val\_del procedure:

```
REMOVING STHDBA2 ACCOUNT . . .  
ACCOUNT REMOVED.  
  
DROPPING PUBLICATIONS TABLE . . .  
TABLE DROPPED.  
  
DROPPING SM2 TABLE SPACE . . .  
TABLE SPACE DROPPED.  
  
REMOVING STHTEST DATABASE . . .  
DATABASE REMOVED.
```



## stopsm utility

The stopsm utility shuts down the StorHouse software. Use this utility prior to performing maintenance on a StorHouse system such as installing software deliveries using the asd utility.

If the System Control (SC) process (which controls the startup and shutdown of the StorHouse software) is running, the stopsm utility sends a shutdown request to SC and waits one minute for a response before initiating the StorHouse software shutdown process. The stopsm utility then does the following:

- Waits up to five additional minutes for the StorHouse software shutdown process to complete.
- Kills the processes listed in the \$SMD\_GENERAL/stoppids file, which is created by the startsm utility. In addition, stopsm stops the diskfree utility, which monitors the minimum amount of available disk space defined for each file system used by the StorHouse operating system.
- Executes the cxclean utility, which cleans up shared system resources (shared memory, locks, and mailboxes) after the StorHouse software stops.
- Returns the UNIX command prompt when the StorHouse software shutdown process is complete.

## Format

stopsm





## sxm utility

The `sxm` utility performs a set of services for internal StorHouse utilities that are invoked from the StorHouse operating system (UNIX) command prompt. Specifically, `sxm` performs the following services:

- Locates the specified utility
- Performs any required setup tasks for the utility
- Runs the utility

## How to use `sxm`

You can execute the `sxm` utility either indirectly by using a UNIX script or directly as part of the command format when you execute an internal StorHouse utility.

### Executing `sxm` indirectly

Ordinarily, you execute `sxm` indirectly when you execute an *imposter module* for a StorHouse utility. An imposter module is a UNIX script that contains a single line of code, which executes both `sxm` and the utility for which the module was created. There is an imposter module for each StorHouse/SM utility that requires `sxm` services.

Imposter modules are located in `$OPER`. Because the directories defined for `$OPER` are included in the path for the UNIX operator account, you can execute

a StorHouse utility from any directory regardless of the location of the actual utility script.

The name of an imposter module always corresponds to the name of the StorHouse utility for which it was created. For example, qchsend represents the imposter module for the qchsend utility.

## Executing sxm directly

An imposter module is not available for the StorHouse/RM cleanipcs utility because a StorHouse system may contain more than one version of cleanipcs. Although you can execute cleanipcs by explicitly including the sxm command in the command format for the utility, FileTek recommends that you do not. Instead, change to the directory that contains the version of cleanipcs you want to execute, then execute the cleanipcs command. Refer to Chapter 10, “cleanipcs utility,” for more information on how to execute cleanipcs.

## Format

`sxm pgm_name [arg ...]`

Argument	Description
<i>pgm_name</i>	Name of the internal StorHouse utility that you want to run.
<i>arg</i> ...	Argument list that you want to pass to the <i>pgm_name</i> program.

## syscreate utility

The syscreate utility creates a StorHouse database. The utility does the following:

- Creates system tables and indexes (or *metadata*) for a database
- Grants SELECT privilege on system tables to the PUBLIC account (this means that all users who are authorized to access this database can query the system tables)
- Grants DBA, RESOURCE, and SCAN privileges to the SYSADM account (this means that the SYSADM account has all database privileges on this database)

When you create a StorHouse database, StorHouse/RM creates a database directory—or system tablespace—on the UNIX file system and a set of metadata for the new database. Database directories are stored under the path /filetek/sth/sthdb (\$STHDBS). The directory name is comprised of the name you assign to the database plus a suffix of .dbs.

You must adhere to StorHouse naming conventions when you create a StorHouse database. In addition, if you use StorHouse in conjunction with a local database, you must follow the naming rules for your local database. Refer to the *StorHouse Database Administration Guide*, publication number 900108, for more information.

As an alternative to using the syscreate utility, you can use the StorHouse/Admin component of Control Center to create a database. Refer to *Getting Started with StorHouse/Admin*, publication number 900135, for more information.

## Format

syscreate {*database\_name*} [-v]

Argument	Description
{ <i>database_name</i> }	Name you want to assign to the StorHouse database you are creating. The name must follow the naming rules described in the <i>StorHouse Database Administration Guide</i> and must not already exist in the current StorHouse system.
-v[erbose]	Displays (at your terminal) each step that syscreate performs as it runs.

---

## timeup utility

The timeup utility performs a status check to determine whether the StorHouse software is running. You can also use this utility to determine:

- How long the StorHouse operating system has been running since the system was last booted
- How long the StorHouse software has been running since the last time it was started

Executing this utility can be a useful step in the troubleshooting process for StorHouse software.

The timeup utility displays the current date and time, the date and time the StorHouse server was last booted (when UNIX was started), and the date and time the StorHouse software was last started. If the StorHouse software is not up, the utility displays the message “SM Software is not running.”

### Format

timeup

## Sample output

The following sample output is displayed if the StorHouse software is running successfully.

Current date/time:      Wed Mar 31 12:57:39 1999

Storage Machine booted:    Mar 21 20:21

SM Software started:   Tue   Mar 23 14:37:29 1999

## List of terms

<b>asd</b>	Automated Software Delivery utility, which is an internal StorHouse utility that is used to install FileTek StorHouse software enhancements and fixes on existing StorHouse systems.
<b>Basic Device Support (Base, or B) Facility</b>	The StorHouse software facility that provides a mechanism for accessing all user data storage devices and network devices used for data transfer. The facility operates in such a way that the specific device characteristics for a device type are transparent to the rest of the StorHouse software. In addition, the Base Facility logs errors generated by these devices.
<b>bconfig</b>	A StorHouse Base Facility system file (base configuration), which maintains device configuration information for use by various StorHouse operations.
<b>bldv (memory)</b>	The library device volume system file, which maintains device and volume information for a StorHouse library. The bldv file keeps track of information such as which volume resides in which slot of a library device. This file is not used for “intelligent libraries” such as ACSLS.
<b>bldv_init</b>	An internal StorHouse utility that initializes the library device volume (bldv) system file.
<b>bmon_ctcs</b>	An internal StorHouse utility that monitors the serial port of a Polaris model 8900 Sbus board. This utility logs all serial port output to a bmon/ <i>N</i> ib.out file, where <i>N</i> is the board number. You execute the bmon_ctcs utility using the ctc <sub>s</sub> _trace start command from the StorHouse operating system (UNIX) prompt.

<b>bmon_dev</b>	An internal StorHouse utility that monitors the UNIX system log for device errors and issues a Call Home if it identifies errors.
<b>bmon_nvmm</b>	An internal StorHouse utility that periodically monitors the battery status of non-volatile memory (NVM) boards and issues a Call Home if the batteries are low or aren't functioning properly.
<b>build_sxm</b>	An internal StorHouse utility that configures a new StorHouse system and initializes the permanent storage area.
<b>Call Home</b>	A feature within the StorHouse software that enables a StorHouse server to notify headquarters automatically of system problems. This feature is used to transport initial diagnostic information prior to a trouble ticket being opened. <i>See also</i> Clientele and trouble ticket.
<b>cfcreate</b>	An internal StorHouse utility that creates a StorHouse system file. You can execute cfcreate from the StorHouse operating system (UNIX) prompt or from within a UNIX shell script.
<b>cfdelete</b>	An internal StorHouse utility that deletes a StorHouse system file. You can execute cfdelete from the StorHouse operating system (UNIX) prompt or from within a UNIX shell script.
<b>Clientele</b>	The FileTek trouble ticketing and tracking system, which tracks CPRs through the system until they are resolved and subsequently closed.
<b>config_dev</b>	A device configuration template file that you edit and tailor for each customer's StorHouse system. The config_dev file serves as input to the gen_dev utility.
<b>CPR</b>	A FileTek Customer Problem Report. Also referred to as a trouble ticket. Clientele opens a CPR automatically each time a Call Home is received. In addition, a customer support representative can create a CPR manually.
<b>.cshrc</b>	Static file (script) that the StorHouse system executes when you log on to the StorHouse operating system with the operator account. This file



	sets up the operating environment (environment variables, aliases, and so on) for the current UNIX session.
<b>.cshrc.local</b>	A file that contains local site changes to StorHouse/SM environment variables. Commands in this file are executed after those in the <code>.cshrc</code> and <code>.cshrc.rm</code> files.
<b>.cshrc.rm</b>	A file that defines the StorHouse/RM environment variables. Commands in this file are executed after those in the <code>.cshrc</code> file.
<b>cxclean</b>	An internal StorHouse utility that cleans up shared system resources (shared memory, locks, and mailboxes). In addition, <code>cxclean</code> removes the temporary files created to manage the system resources.
<b>cxstatus</b>	An internal StorHouse utility that displays the status of the shared system resources (shared memory, locks, and mailboxes).
<b>diskfree</b>	An internal StorHouse utility that uses entries in the <code>vfstab</code> (file system table) file to monitor the minimum amount of available disk space defined for each file system used by the StorHouse operating system. If the amount of available space for a file system is below the threshold defined, <code>diskfree</code> issues a Call Home report with a “critical disk space shortage” error message.
<b>gen_dev</b>	An internal StorHouse utility that creates a new device configuration file ( <code>bconfig</code> ) for a StorHouse system or updates the existing one. The device information contained in the <code>bconfig</code> file includes things like device, drive, and media types as well as the number of slots and available free pool volume sets in each library device. The <code>gen_dev</code> utility also creates the library device volume ( <code>bldv</code> ) system file, which maintains library device and volume information for a StorHouse system.
<b>gzip</b>	A UNIX program that compresses, or reduces, the size of a UNIX file or a tar file. The program appends a suffix of <code>.gz</code> to the file name.

**A**

**List of terms**

---

jgen

**jgen**

An internal StorHouse utility that initializes and modifies the Storage Allocation Manager (J Facility) files. It creates and opens J files, and allows you to create free pools, volume sets, file sets, and empty volumes.

**JUSTDOIT**

A file that you create and add to the \$UUCP\_SPOOL/asd directory on a customer's StorHouse system when you are ready to install StorHouse asd software delivery files containing enhancements or fixes. Each time the startsm utility is executed, it checks for the existence of the JUSTDOIT file and installs pending software fixes if it finds it.

**logical link system  
services control point  
(L-SSCP)**

A unidirectional communication path between an IBM MVS host system and a StorHouse system that is established when an application requests a StorHouse-host link. Several L-SSCP sessions can be associated with one P-SSCP. Each L-SSCP session requires two device numbers. *See also* physical link system services control point and system services control point.

**L-SSCP**

*See* logical link system services control point.

**non-volatile memory**

Permanent storage used by the StorHouse software to store information that must be recoverable across system shutdowns, crashes, and power losses, and must also be accessed and updated quickly.

**NVM board**

*See* Non-volatile memory.

**ODU**

Optical disk unit.

**performance buffer**

The portion of storage level F (magnetic disk) that contains performance copies of files waiting to be migrated or backed up to their primary file sets. The performance buffer is a file set named \$\$BUFFER in the volume set named MAGDISK.

**physical link system  
services control point  
(P-SSCP)**

The capability that handles allocation control operations in communications between an IBM MVS host system and a StorHouse system. Software on the host and on the StorHouse system implements the P-SSCP capability. One P-SSCP exists for each image. *See also* logical link system services control point and system services control point.

<b>program temporary fix</b>	A software maintenance fix for an IBM MVS host environment.
<b>P-SSCP</b>	<i>See</i> physical link system services control point.
<b>PTF</b>	<i>See</i> program temporary fix.
<b>qm_maint</b>	An internal StorHouse utility that you use to maintain the Quality Maintenance Manager (qmm) system files generated by the genqmm utility.
<b>redo_pb</b>	A UNIX shell script, provided with the StorHouse software, that you use to replace the performance buffer on an existing StorHouse system. You must modify the script to include site-specific details before you execute it.
<b>redo_pb.add_only</b>	A UNIX shell script, provided with the StorHouse software, that you use to add a new level F device to an existing StorHouse system. You must modify the script to include site-specific details before you execute it.
<b>Resource Management (R) Facility</b>	The StorHouse software facility that controls data transfer, volume management, and some aspects of file storage.
<b>RL process</b>	The Library Device Manager process, which is part of the StorHouse Resource Management Facility. This process is responsible for handling library devices that are attached to a StorHouse system, under the direction of the Resource Management Facility RQ process. One RL process exists per library device.
<b>RQ process</b>	The Queue Manager process, which is part of the StorHouse Resource Management Facility. This process queues various requests (for example, volume movement, transfer, extent removal, and recovery) to special-purpose processes.
<b>rxt_show</b>	An internal StorHouse utility that you use to display information about every record of the Resource Management Facility transfer table (Rxt). The rxt_show utility does not require any input.

<b>sct_init</b>	A customer support procedure that initializes the system control table, which is stored in permanent storage (usually non-volatile memory).
<b>smc file</b>	A file that is used by StorHouse to control system startup. FileTek customer support installs the initial copy of the smc file in the \$BUILD directory during StorHouse installation. The smc file contains an ordered list of processes to start, StorHouse system files to recover, and startup messages for the system operator. The startsm utility checks for the existence of the smc file; if the file doesn't exist, the system can't start up.
<b>software enhancement</b>	One or more new or replacement StorHouse software files that either enhance an existing feature or add a new feature. A customer support representative generally groups software enhancements into tar files, compresses them, and installs them on a customer's StorHouse server using either the asd utility or the AUTOMV utility.
<b>software fix</b>	One or more StorHouse replacement files that correct errors in the StorHouse software. A customer support representative generally groups software fixes into tar files, compresses them, and installs them on a customer's StorHouse server using either the asd utility or the AUTOMV utility.
<b>Solaris</b>	A UNIX operating system product from SunSoft, a division of Sun Microsystems corporation.
<b>SPR</b>	A System Problem Report, which is a formally submitted request for improved function for a given piece of FileTek software. An SPR is generated internally by FileTek personnel.
<b>spx</b>	The system parameters system file, which is created by the build_sxm utility for a newly installed StorHouse system. This file contains the generic set of StorHouse system parameters and their definitions.
<b>spx_maint</b>	The System Parameter File Maintenance utility, which is an internal StorHouse utility that you use to display, add, delete, and modify system

	parameters. System parameters reside in the system parameters (spx) system file.
<b>spx_maint.inp</b>	The maintenance file for the system parameters (spx) system file. This file contains site-specific modifications to StorHouse system parameter information.
<b>SSCP</b>	<i>See</i> system services control point.
<b>startsm</b>	An internal StorHouse utility that starts the StorHouse software manually. You can execute startsm interactively or as part of the boot process by using a shell script.
<b>stopsm</b>	An internal StorHouse utility that shuts down the StorHouse software.
<b>Storage Allocation Manager (J) Facility</b>	The StorHouse software facility that manages the allocation of storage for user data such that the specific characteristics of the storage media do not have to be known by other StorHouse facilities.
<b>sw_log</b>	A StorHouse software report that is generated every day on all customers' StorHouse systems by the StorHouse daily procedure. The sw_log file describes the state of the system, including the current asd delivery level and a list of fixes files that are pending installation. The report also includes error information extracted from the StorHouse administration log.
<b>sw_log.local</b>	A file containing a script that executes the AUTOMV utility on a daily basis. You can set up this script to execute site-specific procedures.
<b>System Control (SC) process</b>	A StorHouse process, which is started by the startsm utility, that controls the startup and shutdown of the StorHouse software.
<b>system file</b>	A StorHouse internal file that resides on magnetic disk and contains programs, directories, account data, or other data used to control the system.

**A**

**List of terms**

---

system services control point (SSCP)

**system services  
control point (SSCP)**

A component within a subarea network for managing the configuration, coordinating network operator and problem determination requests, and providing directory services and other session services for end users of the network. Multiple SSCPs, cooperating as peers with one another, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its own domain. In this manual, SSCP refers to FileTek's direct connect network. *See also* physical link system services control point and logical link system services control point.

**tar**

The UNIX tape archive utility for backing up UNIX files to tape. It is also a software-packaging tool for combining multiple files in a single file called a tar file.

**trouble ticket**

A FileTek Customer Problem Report (CPR).

**uucp utility**

A standard UNIX program used to copy files from one UNIX system to another.

**ZID**

Zero installation defects.

**ZUD**

Zero unscheduled downtime, which is the primary objective of the StorHouse software support organization.

# **StorHouse directory structure**

This appendix describes the StorHouse directory structure, that is, the environment variables and directories that make up a StorHouse system. Use this chapter as a reference to familiarize yourself with the location of the directories that you need to work with to support a StorHouse system.

## **Environment variables**

An environment variable is a variable that defines an aspect of your working environment such as your home directory or other frequently used directory path. The value for a variable can also be a number or a character string. You can easily identify an environment variable because it is preceded by a \$ symbol, which instructs UNIX to translate the text following the symbol into the value defined for the variable.

Environment variables are set during the login procedure. Therefore, the value for a variable depends on the UNIX account you use to log in. For example, the \$HOME environment variable is defined as /filetek/operator for the operator account. When you log in to the StorHouse server using the UNIX operator account, the .cshrc and .cshrc.local files (and the .cshrc.rm file, if StorHouse/RM is installed on the system) set up the environment variables for that account. Setting up the environment variables at login allows a user of the operator account to run most of the StorHouse utilities regardless of their location on the StorHouse server.

An environment variable that represents a directory path serves as a shortcut that simplifies the command format when you issue a UNIX command. Environment

## B

## StorHouse directory structure

## Environment variables

variables are generally shorter and easier to remember than complete directory paths. For example, rather than specifying a long pathname such as `/usr/spool/uucppublic/filetek` when you issue the UNIX change directory (`cd`) command, you can specify the environment variable `$UUCP_SPOOL`.

The following table lists and describes the most important environment variables that are defined on a StorHouse system for the UNIX operator account. The variables are listed in alphabetical order.

**Table B-1: Environment variables—UNIX operator account**

Variable name	Value
<code>\$BUILD</code>	<code>/filetek/operator/build_sxm</code>
<code>\$FILETEK_ETC</code>	<code>/filetek/etc</code>
<code>\$G</code>	<code>/filetek2/general</code>
<code>\$HOME</code>	<code>/filetek/operator</code>
<code>\$HQ_MAIL</code>	<code>ps</code>
<code>\$HQ_TRANSPORT</code>	<code>internet, uucp, or none</code>
<code>\$LD_LIBRARY_PATH</code>	<code>/opt/SUNWsymon/lib:/filetek/v5rx/o/osta:/usr/openwin/lib:/etc/raid/bin</code>
<code>\$LOGNAME</code>	<code>operator</code>
<code>\$MAIL</code>	<code>/var/mail/operator</code>
<code>\$MANPATH</code>	<code>/usr/share/man:/opt/SUNWsymon/man</code>
<code>\$ND_PATH</code>	<code>/etc/raid/bin</code>
<code>\$OBIN</code>	<code>/filetek/operator/bin</code>
<code>\$OPER</code>	<code>/filetek/v5rx/o/oper</code>
<code>\$OP_MODE</code>	<code>DEBUG or PRODUCTION</code>
<code>\$OPT_FAST_SHFILE _VOL</code>	<code>TRUE</code>
<code>\$OSTA</code>	<code>/filetek/v5rx/o/osta:/filetek/sth/v2rx/bin:/filetek/sth/v2rx/lib</code>



**Table B-1: Environment variables—UNIX operator account (continued)**

Variable name	Value
\$P	/filetek/smd
\$PATH	/filetek/operator/bin:/filetek/bin:/usr/bin:/usr/ccs/bin:/usr/sbin:/usr/platform/SUNW,Ultra-Enterprise/sbin:/filetek/v5rx/o/oper:/filetek/v5rx/o/operator/bin:/sbin:/etc/raid/bin:/usr/openwin/bin:/opt/SUNWsymon/sbin:/opt/SUNWsymon/bin
\$PRI_DISK	/filetek
\$PS_DEVICE	NVM, NVM1, NVM2, or DISK
\$S	/filetek2/smd
\$SEC_DISK	/filetek2
\$SHELL	/bin/csh
\$SM_DLKNAM	CMD_LINK
\$SM_OPER	local0
\$SM_RELEASE	v5rx
\$SM_TLKNAM	1200
\$SMCONFIG	"/filetek/operator/SMCONFIG"
\$SMD_GENERAL	/filetek2/general
\$SMD_PRIMARY	/filetek/smd
\$SMD_SEARCH	/filetek2/general:/filetek/smd:/filetek2/smd
\$SMD_SECONDARY	/filetek2/smd
\$SP_HOST	SM_SP
\$STH_RELEASE	v2rx
\$STHDBS	/filetek/sth/sthdb
\$STHLOG	/filetek2/general/sth
\$STHMACHINE	hostname of current StorHouse system
\$STHROOT	/filetek/sth/v2rx

B

StorHouse directory structure

Environment variables

Table B-1: Environment variables—UNIX operator account (continued)

Variable name	Value
\$STHSQLNW	sqlnw
\$SYMONHOME	/opt/SUNWsymon
\$THIS_HOST	SM_SP
\$USER	operator
\$UUCP_PATH	fttk
\$UUCP_SPOOL	/var/spool/uucppublic/filetek
\$VRAM_CACHE	100
\$VRAM_RELEASE	v1.x
\$WOR	/filetek/v5rx
\$WOSTA	/filetek/v5rx/o/osta

## StorHouse/SM directory structure

The following table lists and describes the directories that you need to be familiar with on a StorHouse 5.x system. Directory paths appear in alphabetical order. The *x* in v5r*x* corresponds to the release level of the StorHouse software (for example, v5r1 or v5r2). If an environment variable exists for a directory, it is noted in the Environment variable column of the table.

**Table B-2: Directory structure for a StorHouse 5.x system**

Directory path	Description	Environment variable
/filetek	Primary disk (file system), which contains StorHouse-related files.	\$PRI_DISK
/filetek/asd.old	Contains old StorHouse software files that were replaced by the asd utility. The files remain in this directory for a minimum of 30 days, at which time asd removes them.	
/filetek/bin	Contains general FileTek executable files such as StorHouse operating system configuration, backup, and restore scripts and other shell scripts.	
/filetek/etc	Contains general FileTek configuration files.	\$FILETEK_ETC
/filetek/etc/polaris	Contains configuration files associated with the Polaris ESCON Direct Connect Interface.	
/filetek/general	Backup place-holder for /filetek2/general directory.	
/filetek/operator	Home directory for the UNIX operator account.	\$HOME

**B****StorHouse directory structure**

StorHouse/SM directory structure

**Table B-2: Directory structure for a StorHouse 5.x system (continued)**

Directory path	Description	Environment variable
/filetek/operator/bin	Contains site-specific executables. The files in this directory are not part of the standard StorHouse software.	\$OBIN
/filetek/operator/build_sxm	Contains site-specific copies of files for building and configuring the local StorHouse system.	\$BUILD
/filetek/operator/run	Contains scripts for the StorHouse Command Language RUN command.	
/filetek/smd	Contains the primary copies of the StorHouse system files.	\$SMD_PRIMARY, P
/filetek/uucppublic	UUCP spooling directory for FileTek use. The system is normally configured to use this directory and its subdirectories in place of /usr/spool/uucppublic/filetek.	\$UUCP_SPOOL
/filetek/uucppublic/asd	Directory to which the uucp utility transmits asd tar files containing software fixes to be installed on the customer's StorHouse system.	
/filetek/uucppublic/automv	Directory to which the uucp utility transmits AUTOMV tar files containing software fixes to be installed on the customer's StorHouse system.	
/filetek/uucppublic/callhome	Directory from which the uucp utility transmits Call Home files to FileTek headquarters.	

**Table B-2: Directory structure for a StorHouse 5.x system (continued)**

Directory path	Description	Environment variable
/filetek/v5rx	StorHouse release ("world") directory, which contains all the operational StorHouse software. With the exception of software delivery files, do not modify files in this directory tree.	\$WOR
/filetek/v5rx/o/build_sxm	Build directory, which contains files for building and configuring StorHouse systems.	
/filetek/v5rx/o/oper	Operator directory, which contains StorHouse executables used to operate and maintain a system.	\$OPER
/filetek/v5rx/o/operator	Contains miscellaneous operator files such as the standard UNIX hidden files and a couple of configuration files.	
/filetek/v5rx/o/operator/bin	Contains operator executables used to support a StorHouse system.	\$OBIN
/filetek/v5rx/o/operator/run	Contains scripts that are set up during installation to be executed using the StorHouse Command Language RUN command.	
/filetek/v5rx/o/osta	Contains the executables, system file definition files, and help files that make up the basic StorHouse system.	\$WOSTA

**Table B-2: Directory structure for a StorHouse 5.x system (continued)**

Directory path	Description	Environment variable
/filetek2	Secondary disk (file system), which contains backup copies of files on /filetek, secondary copies of shadowed system files, and non-shadowed system files.	\$SEC_DISK
/filetek2/asd.old	Backup directory.	
/filetek2/bin	Backup directory.	
/filetek2/etc	Backup directory.	
/filetek2/general	Contains non-shadowed system files. The /filetek file system contains this directory but does not contain a copy of its contents.	\$SMD_GENERAL, G
/filetek2/general/save_YYYYYY	“Save” directories created by prior startsm executions, where YYYYYY is the date the directory was created in Julian format (for example, 00339). Each save directory contains the operator log and other output files from the previous execution of the StorHouse system.	
/filetek2/operator	Backup directory.	
/filetek2/operator/bin	Backup directory.	
/filetek2/operator/run	Backup directory.	
/filetek2/smd	Directory that contains the secondary copies of the shadowed StorHouse system files.	\$SMD_SECONDARY, S
/filetek2/uucppublic	Backup place-holder.	

**Table B-2: Directory structure for a StorHouse 5.x system (continued)**

Directory path	Description	Environment variable
/filetek2/uucppublic/asd	Backup place-holder.	
/filetek2/uucppublic/automv	Backup place-holder.	
/filetek2/uucppublic/callhome	Backup place-holder.	
/filetek2/v5rx	Backup directory.	
/filetek2/v5rx/build_sxm	Backup directory.	
/filetek2/v5rx/oper	Backup directory.	
/filetek2/v5rx/operator	Backup directory.	
/filetek2/v5rx/operator/bin	Backup directory.	
/filetek2/v5rx/operator/run	Backup directory.	
/filetek2/v5rx/osta	Backup directory.	
/usr/kernel/drv	Contains loadable drivers for the StorHouse operating system, including drivers that are provided with the StorHouse software.	
/usr/spool/uucppublic/filetek	UUCP spooling directory for FileTek use. This directory is normally a symbolic link to /filetek/uucppublic, but the link can be omitted and the asd, automv, and callhome subdirectories used if the site administrator prefers not to have this link.	\$UUCP_SPOOL
/usr/spool/uucppublic/filetek/asd	Directory to which the uucp utility transmits asd tar files containing software fixes to be installed on the customer's StorHouse system.	\$UUCP_SPOOL/asd

**B****StorHouse directory structure**

StorHouse/RM directory structure

**Table B-2: Directory structure for a StorHouse 5.x system (continued)**

Directory path	Description	Environment variable
/usr/spool/uucppublic/filetek/automv	Directory to which the uucp utility transmits AUTOMV tar files containing software fixes to be installed on the customer's StorHouse system.	\$UUCP_SPOOL/automv
/usr/spool/uucppublic/filetek/callhome	Directory from which the uucp utility transmits Call Home files to FileTek headquarters.	\$UUCP_SPOOL/callhome

## StorHouse/RM directory structure

The following table lists and describes the directories that you need to be familiar with on a StorHouse/RM v2r $x$  system. Directory paths appear in alphabetical order. The  $x$  in v2r $x$  corresponds to the release level of the StorHouse/RM software (for example, v2r1 or v2r2). If an environment variable exists for a directory, it is noted in the Environment variable column of the table.

**Table B-3: Directory structure for a StorHouse/RM 2.x system**

Directory path	Description	Environment variable
/filetek	Primary disk (file system), which contains StorHouse-related files.	\$PRI_DISK
/filetek/sth/sthdb	Contains StorHouse/RM database directories.	\$STHDBS
/filetek/sth/v2rx	Contains the StorHouse/RM software.	\$STHROOT
/filetek/sth/v2rx/bin	Contains StorHouse/RM executable programs.	



**Table B-3: Directory structure for a StorHouse/RM 2.x system**

Directory path	Description	Environment variable
/filetek/sth/v2rx/etc	Contains a StorHouse/RM configuration file called rdbtemp.data (and drda.config, if DRDA support is installed).	
/filetek/sth/v2rx/lib	Contains library files for the StorHouse/RM software.	
/filetek/sth/v2rx/include	Contains header files for building programs.	
/filetek2	Secondary disk (file system), which contains backup copies of files on /filetek, secondary copies of shadowed system files, and non-shadowed system files.	\$SEC_DISK
/filetek2/sth	Backup directory.	
/filetek2/sth/sthdb	Backup place-holder for /filetek/sth/sthdb directory.	
/filetek2/sth/v2rx	Backup directory.	
/filetek2/sth/v2rx/bin	Backup directory.	
/filetek2/sth/v2rx/etc	Backup directory.	
/filetek2/sth/v2rx/lib	Backup directory.	
/filetek2/sth/v2rx/include	Backup directory.	

**B**

**StorHouse directory structure**

---

StorHouse/RM directory structure

# Index

## Symbols

! (comment character) 19-2  
# (comment character) 19-2  
\$ (character that precedes environment variables) B-1  
\* (wild card character) 4-1  
.cshrc file B-1  
.cshrc.local file B-1  
.cshrc.rm file B-1  
? (StorHouse command prompt) 1-10

## A

ACSL 23-4  
add\_accounts utility  
    description 2-1  
    format 2-3  
    properties of StorHouse accounts 2-2  
asd utility  
    adding a JUSTDOIT file 3-3  
    description 3-1  
    error handling 3-6  
    examples 3-6  
    format 3-4

guidelines for creating a fixes file 3-5  
identifying replacement software to be installed 3-3  
including an optional setup script in a fixes file 3-2  
output 3-8  
    interpreting log data 3-8  
    sample log file 3-10  
overview of the asd process 3-1

asterisk (\*) wild card character 4-1

AUTOMV utility  
    description 4-1  
    format 4-2  
    guidelines for creating a fixes file 4-3  
    including an optional shell script in a fixes file 4-2  
    overview of the AUTOMV process 4-1

## B

balance utility  
    balancing the load of volume mounts 5-1  
    description 5-1  
    format 5-2  
    rules used by the load balancing process 5-2  
    using the RUN command 5-2

Basic device support (B) software facility 23-4

bconfig system file 23-1, 23-11

bmdi utility  
    description 6-1  
    format 6-1  
    sample output 6-2

braces in commands 1-6

build\_sxm utility

    before you use build\_sxm 7-3

    description 7-1

    format 7-3

    high-level tasks 7-1

## C

Call Home error reporting software 31-1, 37-1

CALL\_HOME system parameter 31-1

cfg\_modem utility

    description 8-1

    enabling or disabling remote access 8-3

    examples 8-5

    format 8-4

    implementing modem security 8-2

    input to 8-2

cfg\_net utility

    description 9-1

    examples 9-4

    format 9-3

    input to 9-2

cleanipcs utility

    description 10-1

    format 10-2

commands, general

    command-line format 1-8

    format conventions 1-6

    interactive format 1-8

    redirecting output 1-11

    RUN command format 1-10

    special symbols 1-6

config\_dev file

    description 23-2

    modifying 23-3

conventions

    notational xxviii

    utility command formats 1-6

cpshow utility

    description 11-1

    format 11-1

    sample output 11-2

cpstop utility

    description 12-1

    examples 12-2

    format 12-1

ctest utility

    before you use ctest 13-4

    description 13-1

    format 13-6

    how ctest works 13-1

        files created by 13-3

        primary steps performed by 13-2

        where ctest writes data 13-4

    possible error conditions 13-8

    sample output 13-10

ctestsu utility

    description 14-1

    format 14-3

    how ctestsu works 14-1

        files created by 14-1

        where ctestsu writes data 14-2

    sample output 14-3

cxclean utility

    description 15-1

    format 15-1

## D

- daily procedure 3-3, 4-1, 36-1
  - dbdown utility
    - description 16-1
    - examples 16-2
    - format 16-1
    - sample output 16-2
  - dbup utility
    - description 17-1
    - examples 17-2
    - forcing a database online 17-1
    - format 17-2
    - sample output 17-3
  - dc\_maint utility
    - description 18-1
    - examples 18-5
    - format 18-1
    - sample output 18-6
  - directory structure, StorHouse
    - environment variables for the UNIX operator account B-1
    - StorHouse/RM 2.x systems B-10
    - StorHouse/SM 5.x systems B-5
  - doit utility
    - description 19-1
    - format 19-2
    - rules for command parsing 19-2
    - sample procedure for non-VRAM files 19-14
    - sample procedure for VRAM files 19-13
  - doit.basic utility
    - description 20-1
    - format 20-2
    - sample output 20-3
  - dumpulog utility
    - description 21-1
    - examples 21-4
    - format 21-2
    - possible error conditions and resolutions 21-7
    - sample output 21-5
    - using the RUN command 21-2
- ## E
- environment variables, UNIX operator account
    - description B-1
    - list of B-2
  - error messages
    - ctest utility 13-8
    - dumpulog utility 21-7
    - meta\_rstr utility 28-8
    - recds utility 34-7
  - exclamation point in commands 19-2
  - executing StorHouse utilities from the
    - StorHouse command prompt (?) 1-10
    - StorHouse operating system (UNIX) prompt 1-7
- ## F
- fixes file, guidelines for creating 3-5, 4-3
  - follow utility
    - description 22-1
    - examples 22-3
    - format 22-2
  - format conventions for commands 1-6

## G

### gen\_dev utility

- build\_sxm tasks 23-5
- description 23-1
- examples 23-12
- format 23-11
- high-level tasks for
  - a new StorHouse system 23-5
  - an existing StorHouse system 23-5
- input files
  - list of 23-2
  - modifying 23-3

### genqmm utility

- description 24-1
- files generated by 24-2
- format 24-2

### getting online help from the

- StorHouse command prompt (?) 1-12
- StorHouse operating system (UNIX) prompt 1-12

### guidelines

- asd utility 3-5
- AUTOMV utility 4-3

### gzip utility

- description 25-1
- format 25-2
- sample output 25-2

## H

### help, getting from the

- StorHouse command prompt (?) 1-12
- StorHouse operating system (UNIX) prompt 1-12

hexadecimal number system 19-2

## I

installing a new StorHouse system 23-4

invoking StorHouse utilities from the

- StorHouse command prompt (?) 1-10
- StorHouse operating system (UNIX) prompt 1-7

italics in commands 1-6

## J

jbol system file 23-7

jfset system file 23-8

jsset system file 23-8

jsurf system file 23-8

Julian date, definition 5-3, 21-3, 34-3, 36-2

JUSTDOIT file 3-3

jvset system file 23-8

## L

Library Station software 23-4

log files

- asd.log 3-4
- asd\_master.log 3-4

## M

- maint utility
  - description 26-1
  - format 26-2
  - sample output 26-6
- make\_bkups utility
  - before you use make\_bkups 27-2
  - description 27-1
  - format 27-2
  - how to execute 27-2
- meta\_rstr utility
  - before you use meta\_rstr 28-3
  - description 28-1
  - format 28-4
  - severe error messages 28-4
    - format of 28-4
    - list of 28-8
    - using the problem determination checklist to resolve 28-5
  - steps performed by 28-2
- movevol utility
  - description 29-1
  - examples 29-5
  - format 29-3
  - how movevol works
    - balancing the load of volumes to be moved 29-1
    - monitoring the movement of volumes 29-2
    - reserving a library drive 29-2

## N

- NEWLOG /USER command 21-1, 34-1
- notational conventions xxviii

## O

- online help, requesting from the
  - StorHouse command prompt (?) 1-12
  - StorHouse operating system (UNIX) prompt 1-12
- operator account (UNIX) 1-7
- output of a command, redirecting 1-11

## P

- parentheses in commands 1-6
- port utility
  - examples 30-3
  - format 30-2
- pound sign (#) in commands 19-2
- prompts
  - StorHouse command prompt (?) 1-1
  - StorHouse operating system (UNIX) command prompt 1-1

## Q

- qchsend utility
  - description 31-1
  - examples 31-2
  - format 31-2
  - sample report 31-3
- quotation mark in commands 1-6

## R

ract system file 23-8

rd utility

- description 32-1

- format 32-2

- sample output 32-10

rebuild\_disk utility

- before you use rebuild\_disk 33-1

- description 33-1

- format 33-2

- how to execute 33-2

recds utility

- description 34-1

- examples 34-4

- format 34-2

- possible error conditions and resolutions 34-7

- sample output 34-5

- using the RUN command 34-2

redirecting command output 1-11

reinfo system file 23-8

reinfo2 system file 23-8

reports

- Call Home 31-3

- shelf 36-3

- sw\_log 3-3

Resource management (R) software facility 23-4

rexref system file 23-8

root account. *see* superuser access, utilities requiring  
9-1

roper system file 23-8

rpb system file 23-8

rules

- balance utility 5-2

- doit utility 19-2

RUN command

- general format 1-10

- using with the

  - balance utility 5-2

  - dumpulog utility 21-2

  - recds utility 34-2

rvol system file 23-8

## S

set\_timezone utility

- description 35-1

- examples 35-3

- format 35-3

- sample output 35-4

- selecting a time zone 35-1

shelf utility

- description 36-1

- format 36-1

- sample report 36-3

smc file 7-2, 37-3

software fixes file, guidelines for creating 3-5, 4-3

special symbols in commands

- brace 1-6

- colon 1-6

- comma 1-6

- exclamation point 19-2

- parentheses 1-6

- pound sign (#) 19-2

- quotation mark 1-6

- square bracket 1-6

- vertical bar 1-6



- spx\_maint.inp file
  - description 23-2
  - modifying 23-3
- square brackets in commands 1-6
- standard StorHouse accounts
  - OPERATOR 2-2
  - SERVICE 2-3
  - SYSADM 2-3
  - SYSTEM 2-2
  - USER 2-2
- startsm utility
  - description 37-1
  - format 37-3
- sthtest utility
  - description 38-1
  - sth\_val\_del procedure 38-8
    - files created by 38-9
    - format 38-9
    - sample output 38-9
    - steps performed by 38-8
  - sth\_val\_prep procedure 38-1
    - before you use sth\_val\_prep 38-3
    - files created by 38-6
    - format 38-4
    - input to 38-2
    - sample output 38-7
    - steps performed by 38-2
- stopsm utility
  - description 39-1
  - format 39-1
- Storage allocation manager (J) software facility 23-4
- StorHouse command prompt (?) 1-10
- StorHouse directory structure B-1
- StorHouse operating system, logging on 1-7
- StorHouse software facilities
  - Basic device support (B) facility 23-4
  - Resource management (R) facility 23-4
  - Storage allocation manager (J) facility 23-4
- StorHouse utilities
  - description 1-1
  - executing from the
    - StorHouse command prompt (?) 1-10
    - StorHouse operating system (UNIX) prompt 1-7
  - format conventions 1-6
  - list of 1-2
  - redirecting output 1-11
  - using the UNIX operator account 1-7
- StorHouse, description xv
- StorHouse/RM xv
- StorHouse/SM xv
- superuser access, utilities requiring
  - cfg\_net 9-1
  - make\_bkups 27-1
  - port 30-1
  - rebuild\_disk 33-2
  - set\_timezone 35-1
- sw\_log report 3-3
- sw\_log.local file 4-1
- sxm utility
  - description 40-1
  - format 40-2
  - how to use
    - executing sxm directly 40-2
    - executing sxm indirectly 40-1
- syscreate utility
  - description 41-1
  - format 41-2

## Index

---

T

### system files

- bconfig 23-1, 23-7, 23-11
- jbol 23-7
- jfset 23-8
- jsset 23-8
- jsurf 23-8
- jvset 23-8
- ract 23-8
- reinfo 23-8
- reinfo2 23-8
- rexref 23-8
- roper 23-8
- rpb 23-8
- rvol 23-8

list of 1-2

redirecting output 1-11

using the UNIX operator account 1-7

## V

vertical bar in commands 1-6

## W

wild card character 4-1

## T

### tar

definition A-8

tar files 3-1, 4-1

### timeup utility

description 42-1

format 42-1

sample output 42-2

## U

updating an existing StorHouse system 23-9

### utilities, StorHouse

description 1-1

executing from the

- StorHouse command prompt (?) 1-10

- StorHouse operating system (UNIX)

- prompt 1-7

format conventions 1-6