



StorHouse/RFS Concepts

StorHouse/RFS Release 4.0

Publication Number
900159 Rev. G

May 23, 2006

The FileTek logo consists of the word "FileTek" in white, bold, sans-serif font, centered within a teal square.



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Copyright © 2001-2006 FileTek, Inc. As an Unpublished Licensed Work.
Publication Number: 900159 Rev. G

NOTICE: U.S. GOVERNMENT USERS

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 Unpublished licensed work property of FileTek, Inc. Unauthorized use, duplication or distribution prohibited.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

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

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

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

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



Contents

What is StorHouse/RFS? 1

What’s new in StorHouse/RFS release 4.0? 2

 Enhanced security model 2

 Simplified StorHouse table creation 2

 Enhanced collection process 2

 Data integrity checks 2

 User file information reporting 3

 Dynamic timeout support 3

 Safety directory for user file changes 3

 StorHouse/RFS configuration file reorganization 3

 System file date support 4

 StorHouse/RFS utility changes 4

What is StorHouse? 5

 StorHouse/SM 6

 StorHouse/RM 6

 StorHouse/Control Center 6

What hardware is used by StorHouse/RFS? 8



Contents

What are the StorHouse/RFS software components?	10
StorHouse/RFS file system driver.....	10
StorHouse/RFS server.....	10
StorHouse/RFS configuration file utility.....	11
StorHouse/ODBC driver	11
What's a virtual file system?	11
How does StorHouse/RFS collect and store data?	13
Staging the data	14
Staging directories and user directories.....	14
Creating a file in a staging area.....	16
Writing a file to a staging area.....	17
Collecting the data.....	18
Renaming a file	18
Collection sizing information.....	19
Generating file locator data	20
Storing the data on StorHouse.....	21
Storing file data.....	21
Storing file locator data	22
Storing file locator data in a StorHouse collection	22
Storing file locator data in a StorHouse table array.....	22
Configuring intervals for collecting and storing data	24
Accumulating files to a local collection.....	24
Writing StorHouse collections	24
Collection interval example.....	25
How does StorHouse/RFS retrieve data?.....	26
Searching for file versions.....	28
Browsing the virtual file system.....	29
How does StorHouse/RFS implement security?	31
File security.....	31
Securing files on the local system.....	31
Securing files on StorHouse	32
Extended security for directory levels	34
Specifying the number of directory levels	34
Creating and using the security table.....	34
User and group name management.....	35



What is StorHouse/RFS duplexing?	35
Writing data to two StorHouse systems	36
Retrieving data from the secondary StorHouse system	36
How can you monitor activity?	37
Generating statistics	37
Reporting information for a user file	38
Auditing file activity	40
What recovery facilities are available?	41
Recovering a StorHouse collection	41
Recovering file locator data	41
Recovering a corrupt load file	42
Recovering after a StorHouse/RFS server failure	43
How does StorHouse/RFS support compliant storage?	45
Glossary	46
cache directory	46
checkfile utility	46
collection	46
collection definition	47
collection directory	47
collection set	47
collector	47
collector definition	47
.da1 file	47
.da2 file	48
.dad file	48
.dat file	48
file locator data	48
file version	48
isolation directory	48
.ld1 file	48
.ld2 file	49
.ldr file	49
.lds file	49
load file	49



Contents

local collection	49
local search.....	49
local table.....	49
mirror system.....	50
rename directory.	50
retriever	50
rfsmaint utility	50
rfsrestore utility	50
safety directory.....	50
security table	51
staging area	51
staging directory.....	51
storage definition	51
StorHouse.....	51
StorHouse/Relational File System (RFS).....	51
StorHouse/RFS configuration file	51
StorHouse/RFS duplexing	52
StorHouse/RFS file system driver.....	52
StorHouse/RFS server.....	52
StorHouse collection.....	52
StorHouse database.....	52
StorHouse index	52
StorHouse search	52
StorHouse table	52
system definition.....	53
table array.	53
tblgen utility	53
user directory	53
virtual file system	53



What is StorHouse/RFS?

StorHouse/Relational File System (RFS) is the FileTek file system interface that enables organizations to store a virtually unlimited number of files on a StorHouse® system. With StorHouse/RFS, your enterprise can:

- Store any type of *user file*—document, spreadsheet, presentation, e-mail, voice mail, video clip, photo, and so on—on StorHouse
- Retrieve a file and open it in native mode, for instance, open a spreadsheet with Microsoft® Excel or a presentation with PowerPoint®

StorHouse/RFS presents itself to a host as a local or remote volume, or drive. Applications use this *virtual file system* as the storage location for archived items. Files are “archived” to a folder, or directory, on the virtual file system. StorHouse/RFS collects the files and writes them to StorHouse at user-defined intervals. When an application or user requests an archived item, and that item is on the virtual file system, StorHouse/RFS locates the file and returns it through the virtual file system. The actual storage location—local, network, or StorHouse—is transparent to the user.



What's new in StorHouse/RFS release 4.0?

The following features are new in StorHouse/RFS release 4.0.

Enhanced security model. For Windows® systems, StorHouse/RFS now retains original file system security, including security descriptors, for specified levels of subdirectories on the staging disk. As part of this enhancement, the DDLGen utility has been renamed to tblgen and modified to create a security table for storing security descriptors as well as to set the number of subdirectory levels that must always exist. In this case, the KeepSubdirectories value in the StorHouse/RFS configuration file is ignored.

Simplified StorHouse table creation. Previously, creating database tables for StorHouse/RFS involved running the DDLGen utility to create an SQL script file and then using StorHouse/Admin to load the script file. The DDLGen utility, renamed to tblgen utility, now connects to StorHouse and creates the necessary tables. You no longer have to use StorHouse/Admin to load a script file to create database tables for StorHouse/RFS.

Enhanced collection process. To help minimize I/O, StorHouse/RFS release 4.0 collects files by renaming them to their collection file names instead of copying them to a collection directory. Renamed files reside in a rename directory, which StorHouse/RFS creates automatically at the same level as the staging directory. StorHouse/RFS continues to create file locator data in the collection directory.

Data integrity checks. StorHouse/RFS now calculates a cyclic redundancy check (CRC) for files in staging directories, collection directories, and the cache directory. StorHouse/RFS renames any file that failed a CRC check to an isolation directory. A new checkfile utility enables you to locate errors and to request a copy of a file containing



errors. The CRC calculated when the file data is first written is retained throughout the life of the data and rechecked when the archive data is retrieved. No data is ever written to disk or retrieved from disk without the computation and checking of a CRC.

User file information reporting. StorHouse/RFS provides a command that reports metadata or metadata and media location for a user file. You can submit the command interactively or programmatically. Metadata includes staging and local collection information, primary and mirror StorHouse system information, and the status of collections written to StorHouse.

Dynamic timeout support. Timeouts that occur on reads, writes, opens, creates, and closes of StorHouse files are now based on media type. For example, timeouts for disk-based VSETs are now several minutes shorter than in StorHouse/RFS 3.0. Timeouts for tape-based VSETs are now several minutes longer than in StorHouse/RFS 3.0. You can provide a value for the new TimeoutOverride parameter in a system definition to override the default StorHouse/RFS timeout.

Safety directory for user file changes. StorHouse/RFS can store duplicate entries of metadata for file changes (such as file deletes, renames, and security updates) in a safety directory you specify in the StorHouse/RFS configuration file. Should a load, or .ldr, file become corrupt, StorHouse/RFS automatically initiates a recovery process to re-create the entries from the safety directory. StorHouse/RFS deletes the data in the safety directory after successfully writing the collection to StorHouse.

StorHouse/RFS configuration file reorganization. The SYSTEMS and COLLECTIONS sections were removed from the StorHouse/RFS configuration file. System definitions and collection definitions are still used but no longer part of a section. A new storage definition contains parameters for storing file locator data. Several parameters previously in a collection definition are now in the new storage definition. A new



What's new in StorHouse/RFS release 4.0?

Storage parameter in a collection definition identifies the corresponding storage definition. Multiple collection definitions can use the same storage definition.

System file date support. StorHouse/RFS now retains system file dates, which may be changed using standard utilities such as `utime` and `touch`.

StorHouse/RFS utility changes. In addition to the `DDLGen/tblgen` changes and the new `checkfile` utility, the `rfsmaint` utility, used for compacting and deleting StorHouse collections, was enhanced as follows:

- You can run the `rfsmaint` utility from any system that can access the StorHouse/RFS configuration file or a copy of it and that has ODBC and TCP/IP connectivity to StorHouse.
- New age criteria for selecting eligible collections for compaction or deletion includes a since date or newer than days. When combined with the existing `-b` parameter (before date or older than days), you can specify both an upper and lower bound on the dates used to select eligible collections.
- In order to save processing time, you can bypass collections consisting only of user file changes, that is, only of an `.ldr` file. This type of maintenance may be performed less often than regular maintenance.
- You can now process a specific StorHouse collection. This is useful when a user file is inadvertently stored in a StorHouse collection and must be removed for security reasons.
- If your StorHouse/RFS configuration includes a mirror system, you can request to perform the same maintenance on the secondary system.



- Instead of specifying -p 0 to perform an unconditional delete of all eligible collections, you use a new -u option.

What is StorHouse?

StorHouse is FileTek's data storage and management system for capturing, storing, moving, and accessing gigabytes (GB) to petabytes of relational and non-relational enterprise data. StorHouse technology combines industry-leading, scalable storage devices and Open System processors with specialized storage management and relational database management system (RDBMS) software components. StorHouse is a comprehensive system that manages a complete storage hierarchy of the following:

- Redundant array of independent disk (RAID)
- Serial ATA (SATA) RAID
- Massive Arrays of Idle Disks (MAID)
- Write-once-read-many (WORM) and erasable optical disk jukeboxes
- High-performance and high-capacity (WORM and erasable) tape in automated libraries
- Shelf storage

This highly scalable storage architecture offers a virtually limitless capacity to cost-effectively store data online on different media types, depending on user-dictated access and performance requirements. For example, you can store massive amounts of files that are infrequently accessed on tape media and archive compliance-driven data on WORM



What is StorHouse?

optical or tape to support your enterprise's regulatory retention requirements.

The StorHouse software components—StorHouse/SM, StorHouse/RM, and StorHouse/Control Center—provide features to manage and protect your data.

StorHouse/SM

StorHouse/SM, the StorHouse storage management software component, controls the hierarchy of storage devices and provides system-managed storage that optimizes media usage, response time, and storage costs for each application. StorHouse/SM is also responsible for critical system management tasks, such as data migration, backup, and recovery. All storage management features and benefits apply to the user files you store through StorHouse/RFS.

StorHouse/RM

StorHouse/RM, the StorHouse RDBMS software component, works in conjunction with StorHouse/SM to specifically administer the storage, access, and movement of relational data. StorHouse/RM provides direct row-level Structured Query Language (SQL) access to high volumes of detail data on any storage layer, including tape, in the StorHouse storage hierarchy.

StorHouse/RFS uses StorHouse/RM to store file locator data and to provide a relational search capability for archived files. StorHouse/RFS also supports the use of ODBC-compliant databases to provide this relational capability. StorHouse/RFS release 4.0 runs with StorHouse/RM release 3.3, build 03 and higher.



StorHouse/Control Center

StorHouse/Control Center, the StorHouse Windows®-based application for system and database administration, supports an easy-to-use graphical user interface (GUI) that greatly simplifies StorHouse storage and database management tasks. StorHouse/Control Center consists of one or more StorHouse/Control Center servers communicating with StorHouse/Control Center modules over a TCP/IP network. The three modules are:

- *CCAdmin* helps you monitor and manage your StorHouse/Control Center server(s).
- *StorHouse/Admin* helps you perform storage administration and system operation tasks, such as setting up the storage resources for your application data on StorHouse.
- *StorHouse/Performance Monitor* helps you monitor and analyze the performance and real-time system activity of StorHouse.



What hardware is used by StorHouse/RFS?

What hardware is used by StorHouse/RFS?

StorHouse/RFS operates in Windows and UNIX environments. The hardware components in a StorHouse/RFS configuration are:

- Application hosts
- Windows 2000, Solaris™ 2.6 or higher, HP-UX™ 11 or HP-UX 11i, or IBM® AIX® platform(s) to run the StorHouse/RFS server software
- StorHouse system(s)

In a StorHouse/RFS configuration, an application writes and reads user files through the virtual file system using standard NTFS, NFS, or CIFS protocol. Application hosts access the StorHouse/RFS server platform through the established network protocol. An application host can also run on the StorHouse/RFS server platform.

The StorHouse/RFS server software can run on a standalone server, an application host, or on StorHouse. The StorHouse/RFS server platform and StorHouse communicate through a TCP/IP network. Figure 1 illustrates a configuration with separate application hosts that access a standalone StorHouse/RFS server.

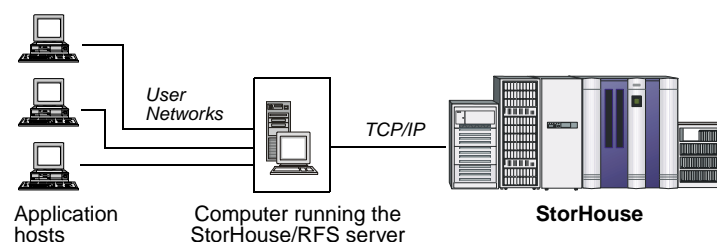


Figure 1: Standalone StorHouse/RFS server

What hardware is used by StorHouse/RFS?



Multiple computers running the StorHouse/RFS server in different locations can communicate with the same StorHouse. Figure 2 shows a configuration with multiple StorHouse/RFS servers running on the same machine as the application.

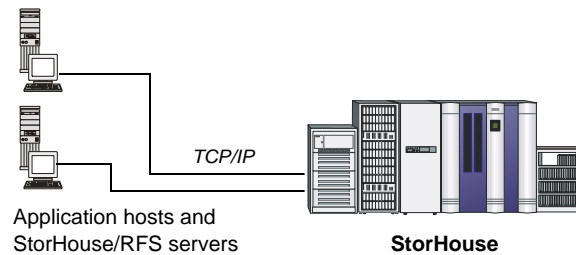


Figure 2: Application hosts on StorHouse/RFS servers

In UNIX environments, the StorHouse/RFS server can run on StorHouse.

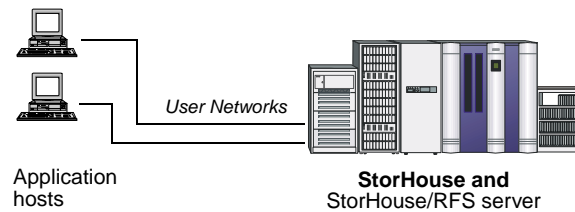


Figure 3: StorHouse/RFS server on StorHouse

For disaster protection, StorHouse/RFS can store and access the same data on two StorHouse systems. This feature is called *StorHouse/RFS duplexing*. See “What is StorHouse/RFS duplexing?” on page 35 for more information.



What are the StorHouse/RFS software components?

What are the StorHouse/RFS software components?

StorHouse/RFS consists of the following software components:

- StorHouse/RFS file system driver
- StorHouse/RFS server
- StorHouse/RFS configuration file utility
- StorHouse/ODBC driver

StorHouse/RFS file system driver

The *StorHouse/RFS file system driver* provides the virtual file system interface between the StorHouse/RFS server and the user application or operating system (OS). This driver enables applications to open files retrieved from StorHouse using their native application, for instance, Microsoft Outlook, Word, and so on.

StorHouse/RFS server

The *StorHouse/RFS server* consists of collector and retriever components.

- A *collector* accesses files from the virtual file system and stores them on StorHouse in a single file, or *collection*, in a modified CD format. You can define multiple collectors for each StorHouse/RFS installation. See page 13 for more information about the collection and storage process.
- The *retriever* locates data on StorHouse or on network or local storage accessible by StorHouse/RFS and returns data in the original file format through the virtual file system. See page 26 for more information about the retrieval process.



StorHouse/RFS configuration file utility

StorHouse/RFS operating parameters are stored in the *StorHouse/RFS configuration file*. Each StorHouse/RFS server has one StorHouse/RFS configuration file.

For a Windows environment, the *StorHouse/RFS configuration file utility* is an application you can use to maintain StorHouse/RFS operating parameters. For a UNIX environment, you can edit the StorHouse/RFS configuration file with any text editor, such as emacs or vi.

Refer to the *StorHouse/RFS Administration Guide*, publication number 900178, for descriptions of the parameters in the StorHouse/RFS configuration file. The *StorHouse/RFS Administration Guide* also describes how to use the configuration file utility.

StorHouse/ODBC driver

The *StorHouse/ODBC driver* is FileTek's ODBC-compliant interface that StorHouse/RFS uses to communicate with StorHouse databases managed by StorHouse/RM. For example, a StorHouse/RFS retriever accesses file locator data through ODBC when searching for files on StorHouse. FileTek provides an ODBC driver appropriate for the client operating system.

What's a virtual file system?

The StorHouse/RFS *virtual file system* is an NFS, NTFS, or CIFS mount or share point. Applications, including operating system utilities, request to create, write, and read file data as well as rename and delete files through the virtual file system just as they would with any mapped Windows drive or NFS mount point. When you install StorHouse/RFS, you select the drive letter or mount point to use as the virtual file system.



What's a virtual file system?

You can organize your files on the virtual file system in any way that best satisfies your site requirements (by user departments, functional requirements, criteria, and so on). For example, you can write all files to one or more folders, or directories, for each user; or you can create separate folders for different applications. Figure 4 shows a virtual file system in a Windows environment. The drive letter is V, and the virtual file system contains two folders for two e-mail archives.

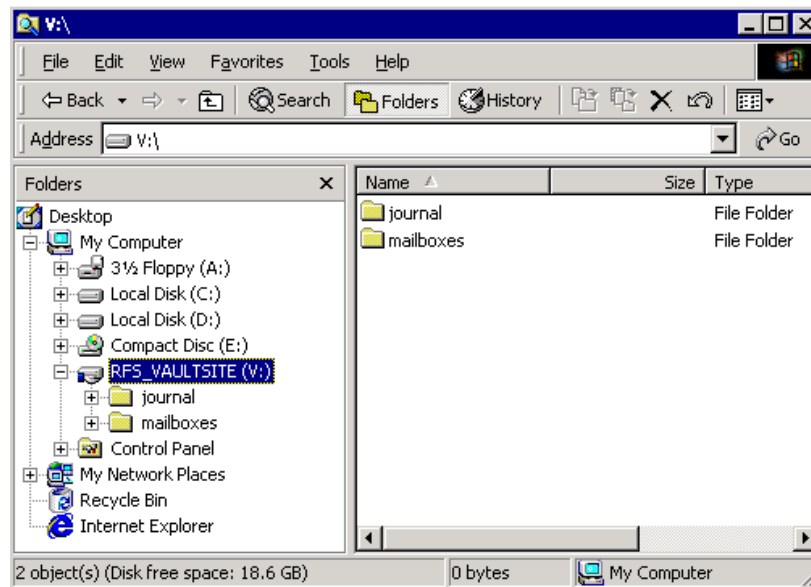


Figure 4: Example of a virtual file system in Windows



Figure 5 shows an example of a virtual file system in a UNIX environment.

```
# df -k /rfs

Filesystem            kbytes    used    avail  capacity  Mounted on
/dev/dsk/c1t0d0s2    192185092 69894668 118446724    38%    /filetek
#
```

Figure 5: Example of a virtual file system in UNIX

How does StorHouse/RFS collect and store data?

A StorHouse/RFS site can archive and access data for multiple *collection sets*, or types of collections, on the virtual file system. The collection process begins when a user or application requests to create a file on the virtual file system. The data is then (1) staged; (2) collected, which includes renaming the staged file and creating associated file locator data; and (3) stored on StorHouse. Finally, (4) file locator data is written to StorHouse tables.



How does StorHouse/RFS collect and store data?

Figure 6 illustrates the StorHouse/RFS collection and storage process.

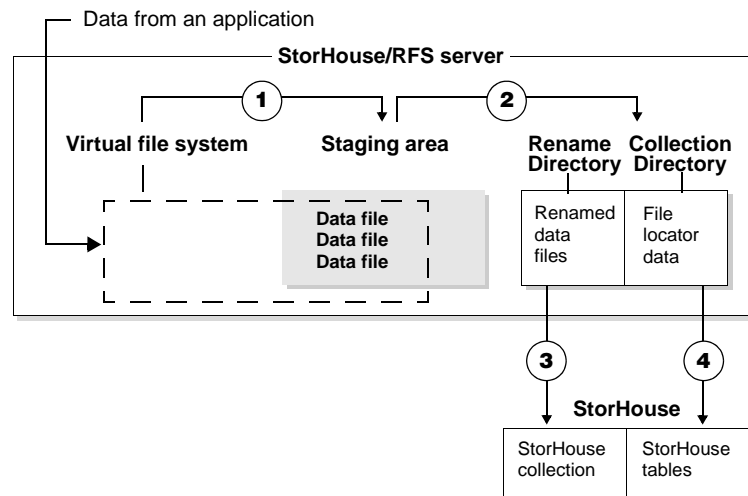


Figure 6: Collecting and storing data

Staging the data

StorHouse/RFS first creates files and writes file data to a *staging area*—a location that holds files before they are collected. You can define multiple staging areas for your enterprise, and you can configure a size limit for each staging area to manage space usage. For performance reasons, this location is typically a local directory on the StorHouse/RFS server platform; however, it can be a network location accessible to the computer running the StorHouse/RFS server.

Staging directories and user directories

A staging area consists of a *staging directory* and a *user directory*. You assign a collector to a staging directory and a user directory. Your enterprise can have any number of staging directories and any number of user directories in the same staging directory. Each staging directory must be

How does StorHouse/RFS collect and store data?



at least one level down from the root level of the virtual file system or drive. Each user directory must be unique in a StorHouse/RFS configuration.

For example, assume you have defined staging areas in a Windows environment for a mailbox e-mail archive and a journal e-mail archive. The staging directory for both archives is called `c:\mailarchive`. The user directory for the mailboxes archive is called `\mailboxes` and the user directory for the journal archive is called `\journal`. Figure 7 illustrates these multiple staging areas.

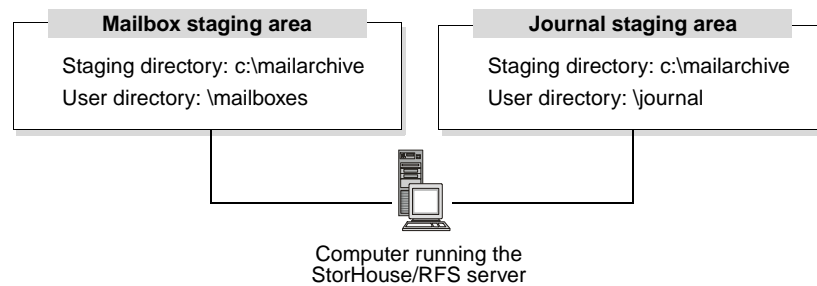


Figure 7: Example of staging areas in Windows

One staging area per StorHouse/RFS configuration may consist of a staging directory only. For example, Figure 8 illustrates the directories, including a staging directory called `stage`, on a UNIX-based StorHouse/RFS server platform.

```
qa2.1> cd /rfs
qa2.2> ls
cache    control  files    log      stage    v4r0
```

Figure 8: Example of a staging area in UNIX

StorHouse/RFS creates a folder in the virtual file system for each user directory. Only the user directory appears in the virtual file system. The staging directory is transparent. For example, if `\mailarchive` is the staging



How does StorHouse/RFS collect and store data?

directory and \mailboxes and \journal are the user directories, then StorHouse/RFS creates folders in the virtual file system for the \mailboxes and \journal user directories. The \mailarchive staging directory does not appear in the virtual file system. Figure 9 illustrates these user directories in a virtual file system. An authorized user or application writes files to these folders.

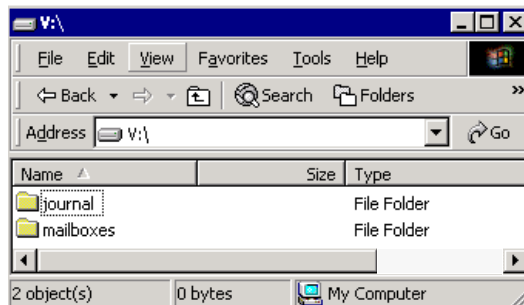


Figure 9: Example of user directories in a virtual file system

Creating a file in a staging area

When the StorHouse/RFS virtual file system receives a request to create a file, it first determines which collector is responsible for the file by comparing the file's path to each collector's user directory. When a match is found, StorHouse/RFS assigns that collector to the file and creates a physical file in the collector's staging area using the complete path of the file. At this point, the file is still empty because the request is only to create a file. StorHouse/RFS returns the handle (name) of the file to the operating system, and all subsequent I/O is done using that file handle.

For example, if a collector named CollectorA is assigned to staging directory /RFS/Collectors and user directory /NewFiles, then when a request is received to create a file called /NewFiles/MyFile.txt, StorHouse/RFS would determine that CollectorA is the responsible collector and create a file with the complete file path name of /RFS/Collectors/NewFiles/MyFile.txt.



Writing a file to a staging area

When StorHouse/RFS receives a request to write data to a file, it:

- Creates a new buffer that will hold the data plus a 16-bit CRC for each 2 KB of data
- Transfers the data to the new buffer with the computed CRC for each 2 KB of data
- Writes the new buffer to the file in the staging area

StorHouse/RFS computes or checks CRCs for all data written to or retrieved from disk. If a write request terminates within a 2 KB segment of the file, then StorHouse/RFS places a CRC at the end of the data even though it does not occupy a full 2 KB on disk. For example, for a write request of 5,000 bytes, one CRC covers bytes 1 to 2048 of user data, another CRC covers bytes 2049 to 4096 of user data, and another CRC covers bytes 4097 to 5000 of user data. If the next write request extends the file, for instance bytes 5001 to 8192, then StorHouse/RFS:

- Retrieves the data for bytes 4097 to 5000
- Computes a CRC and compares it to the one saved with the data
- Appends the additional data
- Computes new CRCs as necessary
- Writes the new data to disk

The file write process continues until the application has completed writing the data.



How does StorHouse/RFS collect and store data?

Collecting the data

Collectors check staging areas almost continuously for files to collect. When a file has remained unmodified for a configured length of time (collector's wait time), the collector:

- Renames the file
- Generates file locator data

Renaming a file

A collector moves a file by renaming it into a *rename directory* at the same level and on the same disk as the collector's staging directory. This causes no additional file I/O, only a directory change. This rename process reduces the risk of errors due to reading and writing processes.

StorHouse/RFS adds the collection name to the renamed file. The format of a collection name is:

`collection_nameyyyymmddhhmmss(C)system_name`

where

- `collection_name` is the name assigned to the collection in the StorHouse/RFS configuration file
- `yyyymmddhhmmss` is the date and time the local collection was created
- `(C)` is a format identifier for a StorHouse/RFS release 4.0 collection (while `(A)` is a format identifier for collections written before release 4.0)
- `system_name` is the name of the system where the collection was created

How does StorHouse/RFS collect and store data?



An example collection name is:

CollectionA20051115083020(C).ELF.FILETEK.COM

StorHouse/RFS places the collection name before the file path and appends the starting logical block number (LBN) to the file name. The *LBN* indicates the position of the data in the final StorHouse collection. For example, if the file RFS/Collectors/NewFiles/MyFile.txt is ready to be collected and the name of the collection is CollectionA20051115083020(C).ELF.FILETEK.COM, then the renamed file is:

CollectionA20051115083020(C).ELF.FILETEK.COM/RFS/NewFiles/MyFile.txt.1

Collection sizing information

Limits for collecting data are as follows.

- The maximum size of a collection is 2 GB; however, you can designate a smaller size.
- StorHouse/RFS can collect a single user file up to 10 terabytes (TB) in size. StorHouse/RFS places any user file larger than the maximum collection size in its own collection. Therefore, the maximum size of a collection with multiple user files is 2 GB. The maximum size of a collection with a single user file is 10 TB.
- The maximum number of files in a collection is 200,000. When this number is reached, StorHouse/RFS closes the current collection and starts a new one.
- You can specify the maximum amount of space a collector can use for renamed files with the maximum collection space parameter in the StorHouse/RFS configuration file.



How does StorHouse/RFS collect and store data?

Generating file locator data

When StorHouse/RFS collects a file, it also:

- Obtains standard file properties, such as file path, file name, and size from the operating system
- Generates StorHouse-related file locator data for each file in the collection
- Stores the file locator data in a single *load file*, or *.ldr file*, in a *collection directory*

StorHouse *file locator data* tells StorHouse/RFS where to locate files on StorHouse. StorHouse/RFS appends a CRC to each entry in the *.ldr file*. Each time StorHouse/RFS reads the file locator data, it recomputes the CRC and compares it to the stored CRC to verify that the file locator data is not corrupt or modified in any way.

Additionally, StorHouse/RFS creates an empty *.dat file* in the collection directory after writing the first file locator data entry to the *.ldr file*. The *.dat file* is renamed during various stages of the write process to indicate which step(s) of the write process have been completed. See “What recovery facilities are available?” on page 41 for more information about the *.dat file*.

The combined data files in a rename directory and their associated file locator data (*.ldr file*) and empty *.dat file* in a collection directory



comprise a *local collection*. Figure 10 depicts the directories and files that make up a local collection.

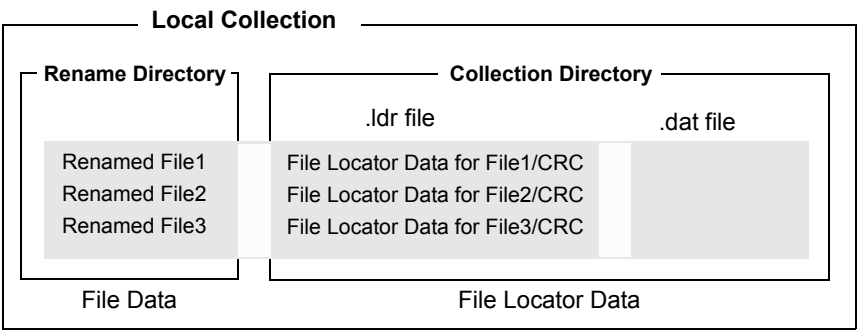


Figure 10: Local collection directories and files

Storing the data on StorHouse

StorHouse/RFS first writes the file data to StorHouse, followed by the file locator data.

Storing file data

At a user-configured interval (maximum write size for the local collection or maximum load interval), StorHouse/RFS creates a *StorHouse collection*—one StorHouse file composed of individual user files—and begins reading the files from the rename directory. StorHouse/RFS reads the files in LBN order, checks the CRCs and if correct, includes them in the data written to StorHouse. Thus, the original CRC that was created when the file system interface first provided the buffer of data to StorHouse/RFS is also written to StorHouse and used in subsequent reads of the archived data.



How does StorHouse/RFS collect and store data?

Any file that is found to be corrupt (fails the CRC check) is flagged and renamed to an *isolation directory* called RFS_ISOLATED. StorHouse/RFS sends an e-mail notification when files have been isolated. You can then run the StorHouse/RFS *checkfile utility* to locate errors and to request a copy of a file containing errors.

If StorHouse/RFS encounters corrupt files during the writing of the collection to StorHouse, it terminates the collection and the load process. On the next write cycle, StorHouse/RFS writes the collection again, this time skipping the corrupt files.

Storing file locator data

StorHouse/RFS stores file locator data in two places on StorHouse:

- StorHouse collection
- StorHouse table array

Storing file locator data in a StorHouse collection. After successfully writing the file data from the rename directory to the StorHouse collection, StorHouse/RFS first writes the .ldr file containing all of the file locator data to the StorHouse collection. StorHouse/RFS reads the .ldr file a line at a time and compares the CRC with the computed CRC. If an entry is correct, StorHouse/RFS writes it to the StorHouse collection. When StorHouse/RFS finishes writing all entries in the .ldr file, it closes the StorHouse collection.

Storing file locator data in a StorHouse table array. Then StorHouse/RFS loads the file locator data into a *StorHouse table* in a *StorHouse database*, again checking the CRC with each line read until the load process for the collection is finally completed. StorHouse/RFS uses a *table array* of up to 255 sets of 32 StorHouse tables to store file locator data for one or more collections. StorHouse/RFS initially creates one set of 32 tables that can contain information for approximately 1 billion files. As required, the software can subsequently create up to a total of

How does StorHouse/RFS collect and store data?



255 sets of 32 tables to store file locator data for approximately 255 billion files.

Figure 11 illustrates the relationship between file locator tables and the individual files in a collection. Each row in a file locator table corresponds to a user file in a specific StorHouse collection.

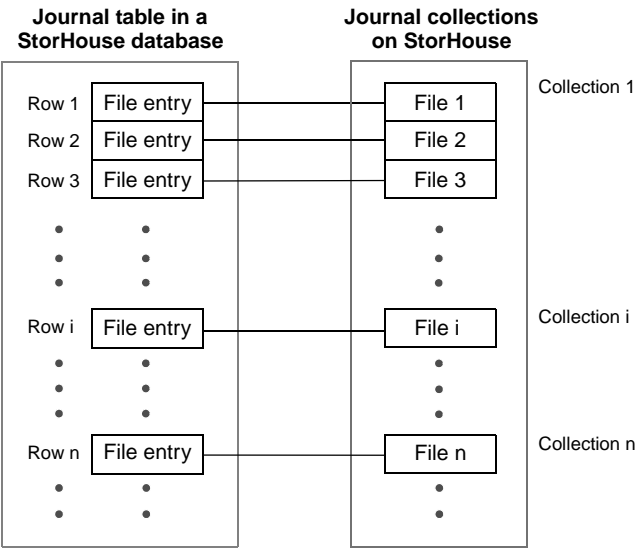


Figure 11: Storing file locator data and StorHouse collections

After StorHouse/RFS writes collections to StorHouse and loads file locator data, it retains local collections in rename and collection directories to enhance search performance. When the maximum collection space limit is exceeded, StorHouse/RFS purges the oldest written collection from those directories but never before first writing the data to StorHouse.



How does StorHouse/RFS collect and store data?

Configuring intervals for collecting and storing data

StorHouse/RFS moves data from a staging area to a rename directory to StorHouse based on parameters—wait time, maximum load interval, and maximum write size—that you define in the StorHouse/RFS configuration file. The purpose of configuring intervals is to transfer data from local storage to StorHouse in a timely manner. This ensures there is always enough local disk space to continue collecting more user files. Some factors to consider when setting these parameters are:

- When should a file in a staging area become eligible for collection?
- How often should StorHouse/RFS write a collection to StorHouse?
- What's the maximum size you want a collection to grow?

Accumulating files to a local collection

A collector renames a file after a specified *wait time*, or number of minutes a file must be idle in a staging area before it is eligible for collection. An *idle file* is one that has not been modified for the specified number of minutes. This wait time prevents files that are in the process of being written from being collected prematurely. A collector checks the last modified time of each file in the staging area, and when the file has aged, the collector renames the file in the rename directory. You define a wait time for each collector.

Writing StorHouse collections

Every 60 seconds, StorHouse/RFS checks for local collections that are ready to be written to StorHouse. StorHouse/RFS transfers a local collection (data and file locator data) to StorHouse when a local

How does StorHouse/RFS collect and store data?



collection is available and one of the following occurs (whichever occurs first):

- The configured time between loads has elapsed
- The configured collection size (maximum 2 GB) has been exceeded
- The maximum number of files (200,000) has been exceeded

The *maximum load interval* is the number of minutes between transfers to StorHouse for each local collection. The interval starts when StorHouse/RFS renames the first file in a local collection. Therefore, this interval is the maximum amount of time the first file will be in a local collection before StorHouse/RFS writes the collection to StorHouse. For instance, if the maximum load interval is 60 minutes and StorHouse/RFS collects the first file 20 minutes after the last collection closed, then StorHouse/RFS closes the current collection and transfers data from the rename and collection directories to StorHouse after 80 minutes. You define this interval for each type of collection, or collection set.

The *maximum write size* is the maximum number of MB a local collection can contain before a collector closes it, that is, no longer accumulates files to the local collection. You define this size for each collection set.

Collection interval example

Assume the following settings:

- The wait time is 1 minute
- The maximum load interval is 720 minutes (twice a day)
- The maximum write size is 1 GB



How does StorHouse/RFS retrieve data?

While StorHouse/RFS checks for completed local collections every 60 seconds, the collector:

- Checks the staging area almost continuously
- Renames eligible files (those that have not been modified for 1 minute) to the local collection for 720 minutes or until the local collection reaches 1 GB

If the collector collects 4 GB of data in a day, then StorHouse/RFS creates 4 collections that day (because the maximum write size is 1 GB). If the collector collects less than 1 GB a day, then StorHouse/RFS may create 2 collections that day (because the maximum load interval is twice a day), depending on when StorHouse/RFS renames the first file in a collection.

How does StorHouse/RFS retrieve data?

You request files archived through StorHouse/RFS using any standard file I/O methods supported by your computer's operating system. Here's what happens when the archived item you request resides on the virtual file system.

1. The application passes the file name to the virtual file system.
2. The StorHouse/RFS retriever conducts a local search and if necessary a StorHouse search.

How does StorHouse/RFS retrieve data?



- A *local search* is a search for a file in a staging directory (if uncollected) or a rename directory (if collected but not yet written to StorHouse).

If the collection has been written to StorHouse but the local files have not yet been removed, StorHouse/RFS reads the data locally rather than on StorHouse. However, if StorHouse/RFS encounters an error during the read of the local data, it invalidates all of the local files for that collection and reads the data from StorHouse.

- A *StorHouse search* is a search that queries a StorHouse table to locate data in a StorHouse collection.
3. When the retriever finds the file, the StorHouse/RFS file system driver notifies the operating system that the file exists, and the operating system tells the application that the file exists.
 4. When a user or application requests to read the file, the StorHouse/RFS file system driver retrieves and returns only the requested data back to the application through the virtual file system. When the file is read, the CRC is recomputed and compared to the one stored on disk. If the CRC is correct, StorHouse/RFS removes it from the data stream before returning the data to the requesting application.

When the file resides on StorHouse, StorHouse/RFS accesses the data in StorHouse collections and caches the returned information in a *cache directory*. Cached data remains in the cache directory for a user-configured amount of space. As StorHouse/RFS reads the data from cache, it recomputes the CRC, compares it to the CRC stored with the data, and then removes the CRC before returning the data to the requesting application. This verifies that the data given to StorHouse/RFS is the same data returned when requested.



How does StorHouse/RFS retrieve data?

Figure 12 illustrates the search process.

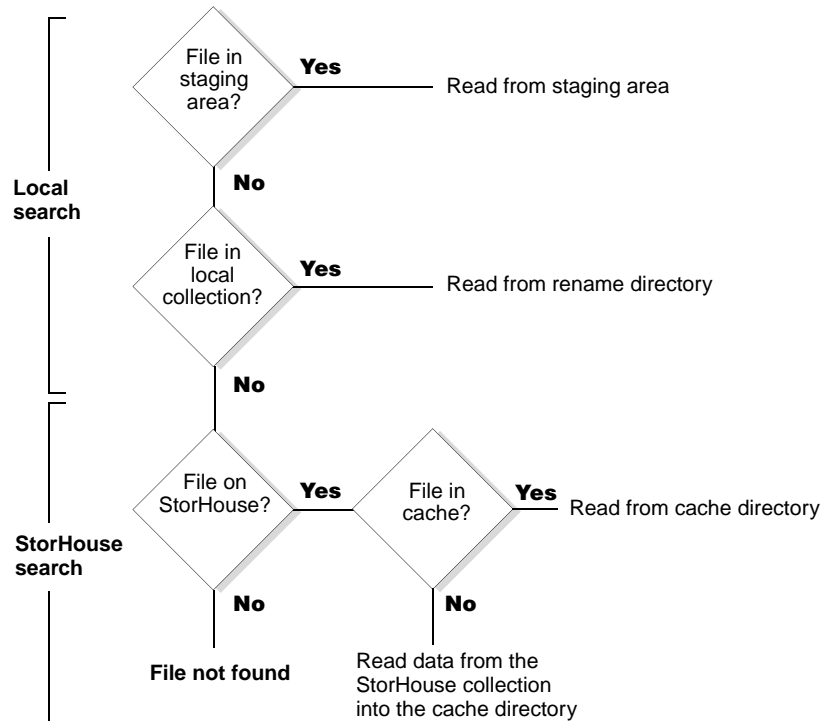


Figure 12: Searching for data

Searching for file versions

A user file archived with the same file name and file path as a previously archived file becomes a new *file version*. Users and applications can search for the current file version and for previous file versions. In the virtual file system, previous file versions are read-only and cannot be overwritten.

Note: StorHouse/RFS creates file versions only for user files that do not have a defined retention period. See “How does StorHouse/RFS support



compliant storage?” on page 45 for more information about setting file retention.

A previous file version name has a suffix of (-n) where n is the version number, for example, (-1), (-2), and so on. A file name without a suffix is the current version, for example, status.txt. A file name with a (-1) suffix is the previous file version, for example, status.txt(-1). A file name with a (-2) suffix is the third version of a file.

A user or application can request a previous file version by appending the suffix to the file name. For example, if a Windows user wanted to access the third version of a file called status.txt in the general directory, he or she could request it by selecting Run from the Start menu and entering the following in the Run dialog box:

```
Notepad v:\general\status.txt(-2)
```

StorHouse/RFS searches for all non-deleted versions of the file (for example, \general\status.txt) in the file locator tables and adds each version found to the virtual directory. For instance, if StorHouse/RFS finds three versions of status.txt, it lists the current version, version (-1), and version (-2) in the virtual directory.

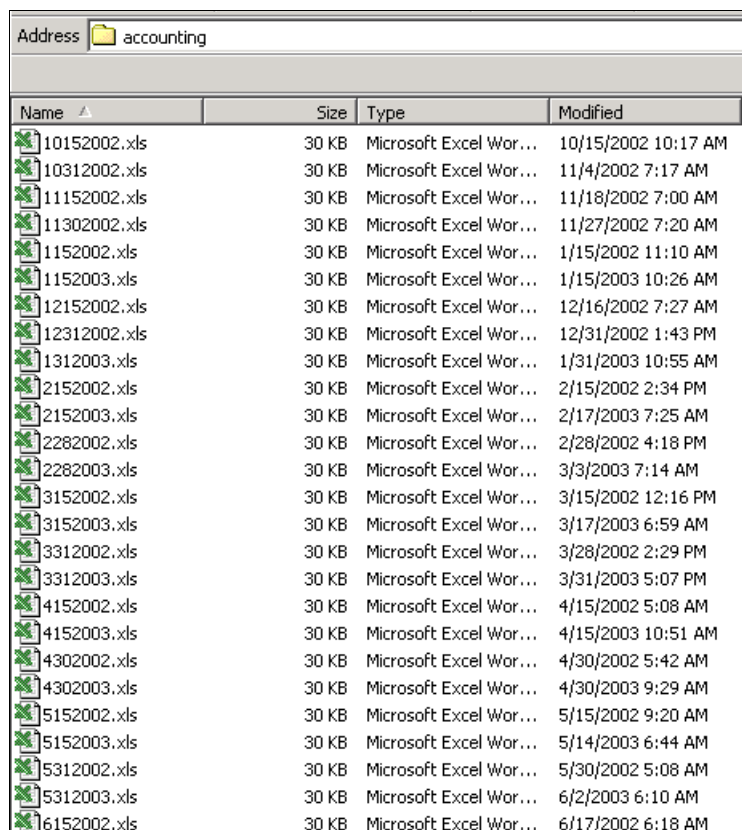
Browsing the virtual file system

StorHouse/RFS enables users to browse the virtual file system like their native file system, for instance, with Windows Explorer or the ls command in UNIX. You can enable or disable browsing for each collection. For example, you can disable browsing for a journal collection



How does StorHouse/RFS retrieve data?

and enable it for a mailbox collection. Figure 13 illustrates the contents of an example user directory called accounting in a virtual file system.



The screenshot shows a file browser window with the address bar set to 'accounting'. The main area displays a list of files with columns for Name, Size, Type, and Modified. Each file is an Excel spreadsheet (.xls) and is 30 KB in size. The files are named with a date in YYYYMM format (e.g., 10152002.xls) and are all of the type 'Microsoft Excel Wor...'. The modified dates range from 10/15/2002 to 6/17/2002.

Name	Size	Type	Modified
10152002.xls	30 KB	Microsoft Excel Wor...	10/15/2002 10:17 AM
10312002.xls	30 KB	Microsoft Excel Wor...	11/4/2002 7:17 AM
11152002.xls	30 KB	Microsoft Excel Wor...	11/18/2002 7:00 AM
11302002.xls	30 KB	Microsoft Excel Wor...	11/27/2002 7:20 AM
1152002.xls	30 KB	Microsoft Excel Wor...	1/15/2002 11:10 AM
1152003.xls	30 KB	Microsoft Excel Wor...	1/15/2003 10:26 AM
12152002.xls	30 KB	Microsoft Excel Wor...	12/16/2002 7:27 AM
12312002.xls	30 KB	Microsoft Excel Wor...	12/31/2002 1:43 PM
1312003.xls	30 KB	Microsoft Excel Wor...	1/31/2003 10:55 AM
2152002.xls	30 KB	Microsoft Excel Wor...	2/15/2002 2:34 PM
2152003.xls	30 KB	Microsoft Excel Wor...	2/17/2003 7:25 AM
2282002.xls	30 KB	Microsoft Excel Wor...	2/28/2002 4:18 PM
2282003.xls	30 KB	Microsoft Excel Wor...	3/3/2003 7:14 AM
3152002.xls	30 KB	Microsoft Excel Wor...	3/15/2002 12:16 PM
3152003.xls	30 KB	Microsoft Excel Wor...	3/17/2003 6:59 AM
3312002.xls	30 KB	Microsoft Excel Wor...	3/28/2002 2:29 PM
3312003.xls	30 KB	Microsoft Excel Wor...	3/31/2003 5:07 PM
4152002.xls	30 KB	Microsoft Excel Wor...	4/15/2002 5:08 AM
4152003.xls	30 KB	Microsoft Excel Wor...	4/15/2003 10:51 AM
4302002.xls	30 KB	Microsoft Excel Wor...	4/30/2002 5:42 AM
4302003.xls	30 KB	Microsoft Excel Wor...	4/30/2003 9:29 AM
5152002.xls	30 KB	Microsoft Excel Wor...	5/15/2002 9:20 AM
5152003.xls	30 KB	Microsoft Excel Wor...	5/14/2003 6:44 AM
5312002.xls	30 KB	Microsoft Excel Wor...	5/30/2002 5:08 AM
5312003.xls	30 KB	Microsoft Excel Wor...	6/2/2003 6:10 AM
6152002.xls	30 KB	Microsoft Excel Wor...	6/17/2002 6:18 AM

Figure 13: Browsing the virtual file system

Although users may be able to browse the virtual file system, access to any listed file is controlled through security measures. See “How does StorHouse/RFS implement security?” on page 31 for more information.



How does StorHouse/RFS implement security?

StorHouse/RFS supports operating system security for individual files as well as for extended security for specified directory levels in the virtual file system.

File security

StorHouse/RFS supports the following levels of security for each user file, or object:

- File owner security – only the user or application that wrote the data can retrieve it
- Group security – users or applications in the same group as the file owner can access the data
- Cross-system, or other, security – users or applications on other StorHouse/RFS systems can or cannot access data

These security levels are enforced for files on the local system and on StorHouse. In the following sections, the term *user* refers to a person or to an application.

Securing files on the local system

Native operating system security—NTFS for Windows or NFS for UNIX—is applied to files that reside in staging areas. For instance:

- If only a file owner has read, write, and execute permission for a file, then only the file owner can archive a file to a staging area and read a file located in a staging area.



How does StorHouse/RFS implement security?

- If all groups have write permission to all staging areas, then all groups can write files to all staging areas.
- If “other” permissions are set to enable users on other StorHouse/RFS systems to read a file and list files in a directory, then other users can read that file in that staging area. In a UNIX environment, “other” permissions are set with UNIX commands. In a Windows environment, “other” permissions are set with a parameter in the StorHouse/RFS configuration file.

Securing files on StorHouse

Once files are collected, StorHouse/RFS ensures that only authorized users can access files and perform other functions, such as changing or deleting files. StorHouse/RFS obtains the following security information and stores the information in the StorHouse database with the file locator data:

- Permissions for file owners, groups, and “other” from the operating system
- For UNIX, user or group name from the operating system
- For Windows, user name and domain name from the operating system
- For Windows, value of the Group parameter (defines a default group) in the StorHouse/RFS configuration file
- For Windows, value of the Permissions parameter (defines cross-system, or “other,” permissions) in the StorHouse/RFS configuration file

How does StorHouse/RFS implement security?



For instance, in a Windows environment:

- If only the file owner is allowed to access a file, then StorHouse/RFS assumes only the file owner, no group, has access to the file on StorHouse.
- If any group is allowed to access a file or directory, StorHouse/RFS assumes group access is allowed, enables the group permissions for the file or directory, and stores the default group name in the file locator data.
- If “other” permissions are set to enable cross-system security, then a file on StorHouse can be shared across StorHouse/RFS systems that have access to the same collection definition. If no “other” permissions are set, then either the group name assigned to the collector must match the group name stored with the file locator data or the requestor must be the file owner.

In a UNIX environment, you do not have to configure any StorHouse/RFS parameters that relate to security.

- If a requestor is the file owner, the requestor has access to the file on StorHouse.
- If a requestor is a member of the file owner’s group, the requestor has access to the file on StorHouse. StorHouse/RFS obtains the primary group name and permissions from the operating system and stores them with the file locator data.
- If a requestor is granted access through “other” permissions, the requestor has access to the file on StorHouse. StorHouse/RFS obtains “other” permissions (set with UNIX commands) from the operating system. You can, however, override the other permissions by setting the Permissions parameter in the StorHouse/RFS configuration file.



How does StorHouse/RFS implement security?

Extended security for directory levels

For Windows environments, StorHouse/RFS can retain security descriptors for specified levels of subdirectories on the staging disk. A user who does not have permission to traverse a directory that is resident on the staging disk will not be able to access files in or below that directory regardless of the security on individual files. You implement extended security in StorHouse/RFS by specifying the number of directory levels to secure and by creating a security table in a StorHouse database.

Specifying the number of directory levels

You specify the number of directory levels by using the StorHouse/RFS *tblgen* utility, which is documented in the *StorHouse/RFS Utilities Reference Manual*, publication number 900181. For example, if you designate five directory levels to secure, then the root and all subdirectories up to five levels below the root would always need to be present on the local staging disk. If directories are missing on the staging disk, then StorHouse/RFS re-creates them and applies the security stored in the security table.

Creating and using the security table

StorHouse/RFS stores the security information for the specified directory levels in a *security table* on StorHouse. You use the *tblgen* utility to create the security table. One security table is required for each table array.

When a collector collects directories, StorHouse/RFS writes an entry in the associated security table to track the complete operating system-specific security and the level of the collected directory relative to the staging root (staging directory). The security table is always current and accessible on the primary or mirror StorHouse/RFS system. StorHouse/RFS updates the security table at collection time rather than waiting until after the collection is written to StorHouse and the file locator data is loaded.



When a user accesses a directory in the virtual file system, StorHouse/RFS reads the security information from the security table and returns it to the operating system exactly as read. StorHouse/RFS does not interpret or understand the security information. StorHouse/RFS then asks the operating system if the user who is trying to access the directory is authorized. The operating system makes all the decisions and returns a yes or no based on this security information.

User and group name management

StorHouse/RFS permanently stores user and group names with the files on StorHouse. If a user or group name changes or is dropped, you can create an *alias* so that the new user or group can still access the files. For example, if Mary's login changes from mjones to msmith, in order to make sure she can access all of her files with both names, you must correlate the old and new name through alias checking. Refer to "Managing aliases for user and group names" in the *StorHouse/RFS Administration Guide* for more information about implementing the aliases feature.

What is StorHouse/RFS duplexing?

StorHouse/RFS can write the same user data and file locator data to separate StorHouse systems that can be geographically dispersed. StorHouse/RFS can also access data on the secondary, or *mirror*, StorHouse system should the primary StorHouse system become unavailable. This disaster protection and data availability feature is called *StorHouse/RFS duplexing*.

Figure 14 illustrates a StorHouse/RFS configuration with a secondary StorHouse system used for StorHouse/RFS duplexing. Any number of



What is StorHouse/RFS duplexing?

StorHouse/RFS servers can communicate with the primary and secondary StorHouse systems.

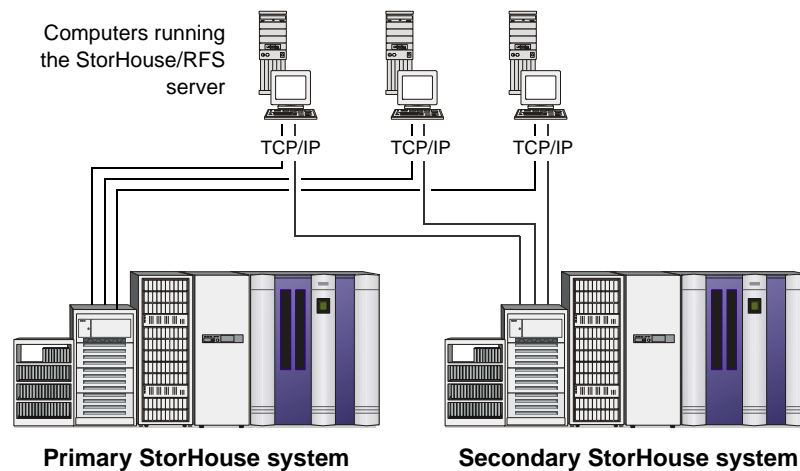


Figure 14: Example of a configuration with duplexing

Writing data to two StorHouse systems

StorHouse/RFS first attempts to write a StorHouse collection and file locator data to the primary StorHouse system. After the data is safely stored, StorHouse/RFS writes the data to the secondary StorHouse system. If the primary StorHouse system is unavailable, StorHouse/RFS writes the data to the secondary StorHouse system and then to the primary StorHouse system when available. StorHouse/RFS always retains the data locally until safely written to both systems.

Retrieving data from the secondary StorHouse system

If the data is not available locally (in a local collection or in a staging area), StorHouse/RFS first attempts to retrieve the data from the primary



StorHouse system. If the primary StorHouse system is unavailable because of connectivity issues (for instance, a network failure), StorHouse/RFS automatically searches for files on the secondary StorHouse system. If both systems remain down after five consecutive search attempts, the request fails.

How can you monitor activity?

You can monitor system-wide activity through statistics as well as obtain information for specific user files through user file reporting and auditing.

Generating statistics

StorHouse/RFS generates statistics to help you monitor local space usage and to assess collection, storage, and retrieval activity. In a duplexing environment, you can store duplicate copies of statistics on the primary and secondary StorHouse systems. At a configured interval, StorHouse/RFS writes statistics to a local file (in HTML, XML, and/or text formats) that you can check at any time. You can also store statistics in a StorHouse database for historical analysis.

Just some of the statistics that StorHouse/RFS provides at each statistics interval are as follows:

- Number of collections written/not written to a collection directory
- Number of files collected/not yet collected
- Average time in seconds for a search to complete
- Number of files returned during a search
- Amount of data written to and read from StorHouse



How can you monitor activity?

- Number of loads completed
- Number of StorHouse writes completed
- Free space in the staging areas, cache directory, and rename directories

Reporting information for a user file

StorHouse/RFS provides a command to generate a report, in text or XML format, identifying the metadata and/or the metadata and media location for a user file in the virtual file system. You can issue the command as a parameter to a utility such as Notepad on Windows or cat on UNIX. Or you can run it programmatically by using the encoded file name with the `fopen()` and `fread()` functions. The command syntax consists of the virtual drive letter or mount point name, the `SM_LOCAL` required keyword, a command verb, and the complete file path name.

The user file reporting command syntax for Windows is:

```
"virtual_drive_letter:\SM_LOCAL\command_verb^complete_file_path"
```

The user file reporting command syntax for UNIX is:

```
"mount_point/SM_LOCAL/command_verb^complete_file_path"
```

Note the following:

- The `^` symbol is used to indicate all path delimiters positioned after the command verb.
- The `virtual_drive_letter` is typically `v` and the `mount_point` is typically `rfs`.
- The program name that will open and read the file (such as Notepad or cat) must precede the command.



Figure 15 illustrates a sample metadata and media location report in XML format for the file `RSFILE_00000010.TXT`. This report was generated by the following command:

```
Notepad "V:\SM_LOCAL\XMLFIND^GENERAL^dates^RFSFILE_00000010.TXT"
```

In Figure 15, each *instance* (information between the `<INSTANCE>` and `</INSTANCE>` tags) represents one occurrence of a user file in the StorHouse database.

```
FILE>\general\dates\rfsfile_00000010.txt
<SYSTEM>
<TYPE>PRIMARY</TYPE>
<DNS>alpha2</DNS>
<INSTANCE>
<STATUS>ACTIVE</STATUS>
<MODIFIED>2005-02-04 10:27:54</MODIFIED>
<SIZE>100000000</SIZE>
<OWNER>ELF.FILETEK</OWNER>
<LBN>59</LBN>
<COLLECTION>DIRECT20050203170548(A).DEV-ELF.FILETEK.COM</
COLLECTION>
<VOLUME>TDA"010161":A</VOLUME>
<LIB>L00</LIB>
<MEDIA>WRITTEN</MEDIA>
</INSTANCE>
<INSTANCE>
<STATUS>ACTIVE</STATUS>
<MODIFIED>2005-02-04 11:29:13</MODIFIED>
<SIZE>100000000</SIZE>
<OWNER>ELF.FILETEK</OWNER>
<LBN>59</LBN>
<COLLECTION>DIRECT20050203170548(A).DEV-ELF.FILETEK.COM</
COLLECTION>
<VOLUME>TDA"010161":A</VOLUME>
<LIB>L00</LIB>
<MEDIA>WRITTEN</MEDIA>
</INSTANCE>
<INSTANCE>
<STATUS>ACTIVE</STATUS>
<MODIFIED>2005-02-04 11:10:55</MODIFIED>
<SIZE>100000000</SIZE>
```



How can you monitor activity?

```
<OWNER>ELF.FILETEK</OWNER>
<LBN>59</LBN>
<COLLECTION>DIRECT20050203170548(A).DEV-ELF.FILETEK.COM</
COLLECTION>
<VOLUME>TDA"010161":A</VOLUME>
<LIB>L00</LIB>
<MEDIA>WRITTEN</MEDIA>
</INSTANCE>
<INSTANCE>
<STATUS>ACTIVE</STATUS>
<MODIFIED>2005-02-04 10:54:14</MODIFIED>
<SIZE>100000000</SIZE>
<OWNER>ELF.FILETEK</OWNER>
<LBN>59</LBN>
<COLLECTION>DIRECT20050203170548(A).DEV-ELF.FILETEK.COM</
COLLECTION>
<VOLUME>TDA"010161":A</VOLUME>
<LIB>L00</LIB>
<MEDIA>WRITTEN</MEDIA>
</INSTANCE>
</SYSTEM>
</FILE>
```

Figure 15: Sample metadata report in XML format

For more information about command syntax, execution instructions, and report fields, refer to Chapter 7, “Operations,” in the *StorHouse/RFS Administration Guide*.

Auditing file activity

When a user deletes or updates a file on the virtual file system, StorHouse/RFS captures those file attributes and stores them along with the date in the StorHouse database with the other file locator data. You can query the StorHouse database with StorHouse/Admin should you need to audit file activity.



What recovery facilities are available?

StorHouse/RFS includes automatic recovery features as well as recovery facilities for any problems that may occur outside of the StorHouse/RFS server.

Recovering a StorHouse collection

Should a StorHouse collection become unreadable or should the media break or become misplaced, you can recover the data from a backup or archive copy. You use standard StorHouse recovery procedures to recover StorHouse collections. Refer to the following publications for more information about recovering StorHouse data:

- *StorHouse System Administrator's Guide*, publication number 900007, Chapter 10, "Recovering User Files and System Files"
- *StorHouse/RFS Administration Guide*, Chapter 5, "Storage management"

Recovering file locator data

Should the magnetic disks containing the file locator data on StorHouse fail, you can restore the StorHouse tables and other system files in a database's system tablespace. The following utilities recover file locator data in StorHouse tables:

- StorHouse/RM metadata restore utility
- StorHouse/RM redo journaling utilities
- StorHouse/RFS rfsrestore utility



What recovery facilities are available?

The StorHouse/RM *metadata restore utility* restores the data from a metadata backup file on StorHouse. Refer to Chapter 11, “Metadata restore,” in the *StorHouse Database Administration Guide*, publication number 900108, for more information about the metadata restore utility.

The StorHouse/RM *redo utilities* available with StorHouse/RM release 3.3 and later capture transactions (for instance, inserts and updates) since the last metadata backup and apply, or replay, the transactions when needed. With these utilities, you can recover to the last committed transaction. Refer to Chapter 12, “Redo journaling,” in the *StorHouse Database Administration Guide* for more information about redo journaling utilities.

The StorHouse/RFS *rfsrestore utility* recovers file locator data for StorHouse collections created since the last metadata backup for an unjournaled database or since the last journal archive or cycle for a journaled database. A FileTek customer support representative runs this utility. Refer to “The rfsrestore utility,” publication number 900182, for more information about this FileTek-run utility.

Recovering a corrupt load file

A load, or .ldr, file contains two types of entries:

- File locator data generated for renamed files in the current collection
- Modifications, such as file deletes, renames, and security changes, to files that belong to other collections

In the event that a .ldr file becomes corrupt, StorHouse/RFS automatically initiates a rebuild process to re-create the file. In a *safety directory* that you specify in the StorHouse/RFS configuration file, StorHouse/RFS creates a *.lds file* containing the modifications. The rebuild process recovers the modifications from the .lds file and the file locator data from the renamed files. Any activity in StorHouse/RFS that was waiting for the .ldr file is suspended during the rebuild process and



continues after the rebuild is complete. Once the collection is safely written to the primary and (if implemented) secondary StorHouse systems, StorHouse/RFS deletes the .lds file in the safety directory.

Recovering after a StorHouse/RFS server failure

If StorHouse/RFS stops working for any reason during the collection or write process, it can fully recover automatically to its previous state. Here's how this recovery works.

As previously described, a local collection consists of data files in a rename directory and two components in a collection directory:

- Empty data file, or .dat file
- Load file, or .ldr file

Once the collection is safely written to StorHouse, StorHouse/RFS changes the .dat file extension to *.dad*. Once the file locator data is safely inserted into the StorHouse table, StorHouse/RFS changes the .ldr file extension to *.ldd*. StorHouse/RFS deletes these files plus all of the renamed files for the collection (provided they have been successfully written to StorHouse) from the local system whenever disk space is needed.

The file extensions are significant because StorHouse/RFS uses them to determine what part of the writing process has completed successfully. In the event of a StorHouse/RFS server failure, StorHouse/RFS fully recovers its previous state by reading all .ldr and .ldd files at startup. It then has a complete replica of the in-memory tables at the time of failure and proceeds normally from that point.

If your installation has two StorHouse systems and has implemented StorHouse/RFS duplexing, additional file extensions are used for the



What recovery facilities are available?

secondary StorHouse system. Here's how it works. Should one or both of the systems be unavailable during the write process, StorHouse/RFS retains the data locally and writes it at the first opportunity. When StorHouse/RFS duplexing is enabled, this process is controlled by the use of file extensions as follows.

- After writing a collection to the primary StorHouse system, StorHouse/RFS changes the *.dat* file extension to *.da2*.
- If the primary StorHouse system is unavailable, StorHouse/RFS writes the collection to the secondary StorHouse system and changes the *.dat* file extension to *.da1*.
- After successfully writing the collection to both StorHouse systems, StorHouse/RFS changes the file extension to *.dad*.

The same method of writing files based on extensions applies to the load file containing file locator data.

- After writing the file locator data to the primary StorHouse system, StorHouse/RFS changes the load file extension from *.ldr* to *.ld2*.
- If the primary StorHouse system is unavailable, StorHouse/RFS writes the file locator data to the secondary StorHouse system and changes the load file extension to *.ld1*.
- After successfully writing the file locator data to both StorHouse systems, StorHouse/RFS changes the load file extension to *.ldd*.

Files continue to reside on the StorHouse/RFS server until they have been successfully written to both StorHouse systems.



How does StorHouse/RFS support compliant storage?

StorHouse supports compliance-driven media, such as WORM tape. StorHouse/RFS also provides a retention feature to help comply with regulatory requirements. The retention feature works as follows.

You can define a file retention period to prohibit users from overwriting or deleting a file for the specified number of days. Any attempt to delete, modify attributes, or overwrite a file when the retention period has not expired fails with access denied.

When a user or application requests to delete, modify attributes, or overwrite a file, StorHouse/RFS determines the expiration date by adding the number of days in the retention period to the file's last modified time. If the last modified time plus the retention period is less than the current time, the retention has expired. For example, if a user archives a file with a last modified time of noon on February 15 and the retention period is 10 days, then the retention expires at noon on February 25. A user may delete, modify attributes, or overwrite a file only after the expiration.

Retention data is stored in StorHouse tables and written at the same time the file locator data is written. If StorHouse/RM is down, any request to delete or overwrite a file fails because StorHouse/RFS must check the database to determine the retention period.

Note the following information about retention periods.

- You set a retention period for a collection set. However, the retention for each user file in a collection set may expire at different times based on each file's last modified time.



Glossary

- A retention period may be minimum of 1 day and a maximum of 65,000 days. You can also set the retention period to “forever,” which means a file may not be deleted or overwritten.
- You can change the retention period at any time. The new value applies only to uncollected files. In other words, any file collected after the change has the new retention and any file collected before the change has the old retention.
- You can also remove the retention requirement later, that is, change a retention period to 0 days. In this case, StorHouse/RFS may create file versions and allow file versions to be deleted with the exception of the original file version.
- Files without retention requirements may be deleted immediately after they are collected.
- StorHouse/RFS creates file versions only for files without retention requirements.

Glossary

The remainder of this document defines StorHouse/RFS terms.

cache directory. The location where StorHouse/RFS places data that it reads from StorHouse.

checkfile utility. A StorHouse/RFS utility that determines the location of a CRC error.

collection. A group of user files stored in a single file. A collection residing on local or network storage accessible to StorHouse/RFS is called a local collection. A collection written to StorHouse is called a StorHouse



collection. A collection also consists of corresponding file locator data. For a local collection, file locator data is stored in a separate file in a collection directory. For a StorHouse collection, file locator data is stored in a table array and within the StorHouse collection. *See also* local collection and StorHouse collection.

collection definition. A profile that defines collection options. Each collection set has a collection definition. You create collection definitions with the StorHouse/RFS configuration file utility (for Windows) or by editing the StorHouse/RFS configuration file (for UNIX).

collection directory. A location where StorHouse/RFS accumulates file locator data for a local collection.

collection set. A group of StorHouse collections. For example, if an e-mail collection is written to StorHouse once a day, then after a month, the e-mail collection set would consist of 30 or 31 StorHouse collections, each consisting of data and associated file locator data.

collector. The StorHouse/RFS server component that checks a staging area for files to collect and accumulates those files into collections. You define a collector by creating a collector definition. You can define multiple collectors. *See also* collector definition.

collector definition. A profile that defines each collector, including where a collector looks for files to collect. You create collector definitions with the StorHouse/RFS configuration file utility (for Windows) or by editing the StorHouse/RFS configuration file (for UNIX).

.da1 file. A renamed .dat file in a StorHouse/RFS duplexing configuration. StorHouse/RFS changes the .dat file extension to .da1 after first writing a collection to the secondary StorHouse system when the primary StorHouse system is unavailable.



Glossary

.da2 file. A renamed .dat file in a StorHouse/RFS duplexing configuration. StorHouse/RFS changes the .dat file extension to .da2 after writing the collection to the primary StorHouse system.

.dad file. A renamed .dat file. In a single StorHouse system configuration, StorHouse/RFS changes the .dat file extension to .dad after successfully writing the collection to StorHouse. In a StorHouse/RFS duplexing configuration, StorHouse/RFS changes the .dat file extension to .dad after successfully writing the collection to both StorHouse systems.

.dat file. A file, called .dat file, that StorHouse/RFS renames during various stages of the write process to indicate which step(s) of the write process have been completed. *See also* .dad file, .da1 file, and .da2 file.

file locator data. Information about each file in a collection. StorHouse/RFS obtains file locator data for each file collected, stores the data in the StorHouse collection and in a StorHouse table, and uses the data to locate files.

file version. A file that has been collected with the same file name and path as a previously archived file. StorHouse/RFS creates file versions only for files without retention requirements. StorHouse/RFS adds a negative numeric suffix (-1, -2, and so on) to a file version name in the virtual directory, for example, status.txt(-1).

isolation directory. A directory called RFS_ISOLATED containing user files that fail a CRC check.

.ld1 file. A renamed .ldr file in a StorHouse/RFS duplexing configuration. StorHouse/RFS changes the .ldr file extension to .ld1 after first loading file locator data into the secondary StorHouse system when the primary StorHouse system is unavailable.



.ld2 file. A renamed .ldr file in a StorHouse/RFS duplexing configuration. StorHouse/RFS changes the .ldr file extension to .ld2 after loading file locator data into the primary StorHouse system.

.ldd file. A renamed .ldr file. In a single StorHouse configuration, StorHouse/RFS changes the .ldr file extension to .ldd after loading the file locator data into StorHouse. In a StorHouse/RFS duplexing configuration, StorHouse/RFS changes the .ldr file extension to .ldd after loading file locator data into both StorHouse systems.

.ldr file. *See* load file.

.lds file. A secondary, or safety, copy of a load file in a safety directory. The .lds file contains modifications such as file deletes, renames, and security changes.

load file. A file, also called .ldr file, containing file locator data for a local collection. StorHouse/RFS renames the .ldr file during various stages of the write process to indicate which step(s) of the write process have been completed. *See also* .ldd file, .ld1 file, and .ld2 file.

local collection. A group of user files stored in one rename directory and related file locator data stored in a collection directory. StorHouse/RFS deletes a local collection when the user-defined maximum collection space is exceeded, but never before successfully writing the collection to StorHouse. *See also* collection. *Contrast* StorHouse collection.

local search. A search for a file in a staging area or a rename directory. StorHouse/RFS starts a StorHouse search if it cannot locate the file locally. *Contrast* StorHouse search.

local table. An in-memory table containing the file locator data that StorHouse/RFS checks during a local search. *Contrast* StorHouse table.



Glossary

mirror system. A secondary StorHouse system with the same user data, and file locator data as the primary StorHouse system. StorHouse/RFS accesses data on the mirror StorHouse system should the primary system become unavailable.

rename directory. A directory that contains renamed user files as a local collection. StorHouse/RFS creates rename directories automatically. The files in one rename directory and their associated file locator data in a collection directory compose a local collection. StorHouse/RFS removes local collection data from a rename directory when the maximum collection space value in the StorHouse/RFS configuration file is exceeded, but never before successfully writing the collection to StorHouse.

retriever. The StorHouse/RFS server component that performs local searches and StorHouse searches when a user or an application searches for files or requests to open a file.

rfsmaint utility. A StorHouse/RFS utility used for compacting and deleting StorHouse collections.

rfsrestore utility. A StorHouse/RFS utility used in conjunction with other StorHouse/RM utilities for recovering file locator tables in a StorHouse database.

safety directory. A directory that contains duplicate entries of user file changes (such as file deletes, renames, and security changes). You specify the directory path in the StorHouse/RFS configuration file. Should a .ldr file become corrupt, StorHouse/RFS automatically initiates a recovery process to re-create the entries from the .lds file in the safety directory. StorHouse/RFS deletes the data in the safety directory after successfully writing the collection to StorHouse.



security table. A table in a StorHouse database that contains security information for Windows systems that use extended security, such as security descriptors.

staging area. The location where StorHouse/RFS writes physical files for a user or application during the collection process. A staging area consists of a staging directory and a user directory. You assign a collector to a staging directory and user directory. A collector removes files from the staging area and adds them to a local collection. *See also* user directory and staging directory.

staging directory. The directory in a staging area where StorHouse/RFS writes physical files during the collection process. The staging directory must be at least one level below the root of the file system or drive. A staging directory may have one or more user directories. A user or application must have permission to access a staging directory. *See also* user directory and staging area.

storage definition. A profile that defines where file locator data is stored on StorHouse. The StorHouse/RFS configuration file contains storage definitions.

StorHouse. The FileTek storage solution that consists of a server, a hierarchy of storage devices, and storage software components—StorHouse/SM, StorHouse/RM, and StorHouse/Control Center.

StorHouse/Relational File System (RFS). The FileTek file system interface to one or more StorHouse systems. StorHouse/RFS enables the collection, storage, and retrieval of a virtually unlimited number of files.

StorHouse/RFS configuration file. The file that contains the operating parameters for a StorHouse/RFS server. You can maintain this file with the StorHouse/RFS configuration file utility (Windows environment) or a text editor (UNIX environment).



Glossary

StorHouse/RFS duplexing. A feature where StorHouse/RFS writes the same data to two StorHouse systems for disaster protection and data availability.

StorHouse/RFS file system driver. The interface between the StorHouse/RFS server and the user application or operating system. This driver provides the virtual file system that enables users or applications to read and write files through StorHouse/RFS.

StorHouse/RFS server. The set of programs—StorHouse/RFS file system driver, StorHouse/RFS collectors, and StorHouse/RFS retriever—that make up StorHouse/RFS. The StorHouse/RFS server runs on Windows 2000, Solaris 2.6 or higher, AIX, and HP-UX 11 or HP-UX 11i platforms.

StorHouse collection. A group of user files stored as a single file on StorHouse. *Contrast* local collection.

StorHouse database. A relational database that contains StorHouse tables used for storing file locator data, collection metadata, security entries, aliases, and statistics for StorHouse/RFS.

StorHouse index. A database component that contains indexing data for a StorHouse table.

StorHouse search. A search that queries a StorHouse table and accesses data in StorHouse collections. *Contrast* local search.

StorHouse table. The StorHouse database component containing file locator data and collection metadata for each collected file, as well as statistics, aliases, and security entries. You create StorHouse tables with the StorHouse/RFS utility called `tblgen`. (StorHouse/RFS automatically creates file locator tables after you create a collections table with the `tblgen` utility.) *Contrast* local table. *See also* table array.



system definition. A profile that defines where a StorHouse collection is stored. A system definition contains the StorHouse server name, DNS name, StorHouse file access group name, StorHouse storage resource names, as well as the account and encrypted password used to log in to StorHouse to write collections. You create system definitions with the StorHouse/RFS configuration file utility (for Windows) or by editing the StorHouse/RFS configuration file (for UNIX).

table array. A set of tables that StorHouse/RFS uses to store file locator information for one or more collection sets. After you run the `tblgen` utility to create a collections table, StorHouse/RFS creates an initial set of 32 tables for file locator data and expands the array as necessary up to 255 sets of 32 tables.

tblgen utility. A StorHouse/RFS utility that creates the following tables in a StorHouse database: aliases, statistics, collections, and security. For Windows installations, you also use this utility to set the number of directory levels for extended security.

user directory. A directory in a staging area. You assign a collector to a staging directory and to a specific user directory. StorHouse/RFS automatically creates a folder or directory in the virtual file system for each user directory. *See also* staging area and staging directory.

virtual file system. The shared drive or mount point on the platform running the StorHouse/RFS server. The virtual file system is the user or application interface to StorHouse/RFS. To an application, the virtual file system, which looks like any other available drive or disk on the network, is where users or applications write and open files.



Glossary