sgi

SGI StorHouse/RFS Configuration File Reference Manual

StorHouse®

SGI StorHouse/RFS Release 5.2.x

Publication Number
007-6323-001

March 2014

Publication Number: 007-6323-001

Contents

Welcome

StorHouse/Relational File System (RFS) is a comprehensive, easy-to-deploy file system interface that enables applications mounting NFS or sharing through CIFS to archive and retrieve medical images, scientific and biomedical research data, documents, e-mail, voice mail, video, and other digitized data in file format to/from a StorHouse virtual storage solution. No API is required. StorHouse simply appears on a network as one or more unified file shares. Users and applications can access the file shares through a traditional drive-letter mapping or a server-oriented file path.

For retrievals, StorHouse/RFS also supports StorHouse/FTP, an optimized FTP server that provides high-speed, sequential retrieval of very large (multiple gigabyte) StorHouse files using any standard FTP client. The system also uses a deterministic algorithm to select record mode retrieval if files to be retrieved are smaller than a user-determined threshold. FTP retrieval capability is in addition to the native NFS and CIFS support that StorHouse/RFS Linux platforms already provide.

Purpose of this guide

This document defines the configuration parameters that are used to control StorHouse/RFS operation. It also describes how to maintain the StorHouse/RFS configuration file using StorHouse/CCi.

Audience

The audience of this manual consists of StorHouse/RFS administrators, the people responsible for configuring StorHouse/RFS and defining how applications data is stored on StorHouse. This manual assumes that these administrators are familiar with StorHouse/RFS concepts/operation, StorHouse concepts/operation, and the StorHouse/CCi web-based administration interface.

What's inside

This manual consists of two chapters and two appendices:

- Chapter 1, "The StorHouse/RFS configuration file," defines the parameters in the StorHouse/RFS configuration file including format; definition; default, minimum, and maximum values; and a parameter example.
- Chapter 2, "Maintaining the StorHouse/RFS configuration file," explains how to maintain the StorHouse/RFS configuration file by updating and deploying a StorHouse/RFS profile through StorHouse/CCi.
- Appendix A, "Additional information," lists deprecated parameters and sections, parameters reserved for Customer Support, and the parameters reserved for StorHouse/CCi. It also describes the consequences of using a text editor to update the StorHouse/RFS configuration file and procedures for rereading the configuration file and restarting the StorHouse/RFS service.
- Appendix B, "Understanding the text version of the StorHouse/RFS configuration file," explains the layout and syntax of the StorHouse/RFS configuration file and provides a configuration file example.

C H A P T E R 1

The StorHouse/RFS configuration file

This chapter defines the parameters in a StorHouse/RFS configuration file.

About the StorHouse/RFS configuration file

The StorHouse/RFS configuration file is a text file that resides on the StorHouse/RFS server. It is comprised of sections and definitions, which contains parameters that control StorHouse/RFS operation.

Table 1-1 lists the sections in the StorHouse/RFS configuration file.

Table 1-1: Sections in the StorHouse/RFS Configuration File

Section Name	Description
[RFS]	Specifies general StorHouse/RFS operating parameters and defaults.
[RFS_PIT]*	Contains restricted use StorHouse/RFS parameters related to point in time (PIT).
[RFSPROFILE]*	Contains restricted use StorHouse/CCi parameters required to manage profiles.
[AUDITLOG]	Specifies information required to create audit log records.
[EXCLUSIONS]	Identifies file masks to exclude from searches and collections.
[COLLECTORS]	Specifies collector names and identifies corresponding collection definitions.
[CACHE]*	Contains restricted use StorHouse/RFS parameters related to caching.
[FTPD]	Specifies configuration parameters for StorHouse/FTP.
[STATS]	Specifies parameters used to generate and store statistics.
[UNIX]*	Contains restricted use StorHouse/RFS parameters to tweak Unix functionality.

Sections annotated with an asterisk (*) have restricted use only. Refer to Appendix A for more information.

Table 1-2 lists the definitions in the StorHouse/RFS configuration file.

Table 1-2: Definitions in the StorHouse/RFS configuration file

Definition Name	Description
System	Defines where collections and statistics are stored on StorHouse.
Storage	Specifies where file locator data and collection metadata are stored on StorHouse and identifies the corresponding system definition.
Collection	Defines specifications for each collection set and identifies the corresponding storage definition.
Collector	Specifies directories where each collector looks for data and the security requirements for files in those directories. It also identifies the corresponding collection definition.

Refer to Appendix A for more detailed information about configuration file layout and syntax rules. The remainder of this chapter defines each configuration file parameter by section and definition.

RFS section

The RFS section contains system-wide values and defaults. The parameters in the RFS section are:

■ CacheDir	■ LogFile
■ ChangedTimeOlderWarning	■ MaxActiveIOsFiles
■ DeleteExcluded	■ MaxCacheSpace
■ ExcludedTimeToLive	■ MaxSMFiles
■ FileAuditPath	■ MaxSMWriters
■ FileCleanupTimeout	■ SafetyPath
■ Flush	■ SpaceAvailable

The RFS section also contains the restricted-use parameters CleanupTimeout,

Debug, DebugLogLongOpSecs, ReturnJukeBox, and Simulate and the deprecated parameters AllowUnixEquivPerms, DefaultDomain, LocalPath, and Version. Appendix A describes these parameters.

Note: DefaultDomain and LocalPath are deprecated in RFS Release 5 but still valid and used in RFS Release 4. LocalPath is hardcoded in Release 5 as /rfs/file/localpath.

CacheDir

Format	CacheDir=<fully qualified path>
Example	CacheDir=/rfs/cache
Default	None
Required	Yes
Dynamic	No
Description	Fully qualified path where StorHouse/RFS caches data that it reads from StorHouse. This directory can be on the StorHouse/RFS server or on a device accessible to the server. The root account owns the cache directory. The ID under which StorHouse/RFS runs requires access to the cache directory.

ChangedTimeOlderWarning

Format	ChangedTimeOlderWarning=<YES|NO>
Example	ChangedTimeOlderWarning=YES
Default	YES
Required	No
Dynamic	Yes
Description	Controls whether RFS logs a warning message when a client (RFS user) modifies a file's "changed time" (date-time) to be earlier than the current setting. Changing the date-time could change the order of the file versions in RFS. It's a good idea to set this value to NO when using backup tools that preserve times. A value of YES enables message logging. A value of NO suppresses message logging.

DeleteExcluded

Format	DeleteExcluded=<YES | NO>
Example	DeleteExcluded=No
Default	No
Required	No
Dynamic	Yes
Description	If an EXCLUSIONS section has been defined in the rfs.cfg file, this parameter controls whether RFS will delete excluded files. DeleteExcluded appears in both the RFS section and the collector definition. The value in the collector definition overrides the one in the RFS section.

ExcludedTimeToLive

Format	ExcludedTimeToLive=<number of minutes>
Example	ExcludedTimeToLive=60
Default	60
Minimum	60
Maximum	None
Required	No
Dynamic	Yes
Description	If the DeleteExcluded parameter is true, this parameter defines the number of minutes an excluded file will remain in RFS before being deleted.

FileAuditPath

SGI recommends using the FileAuditPath parameter in the AUDITLOG section of the StorHouse/RFS configuration file rather than this parameter. While this one still is operational, StorHouse/RFS now groups all audit parameters in the AuditLog section of the configuration file for clarification.

Format	FileAuditPath =<fully qualified path>
Example	FileAuditPath=/rfs/fileactivitylog
Default	None
Required	No
Dynamic	Yes
Description	The directory where StorHouse/RFS writes the file activity log. If you omit this parameter, StorHouse/RFS does not create a file activity log.

FileCleanupTimeout

Format	FileCleanupTimeout=<numeric in units of minutes>
Example	FileCleanupTimeout=1440
Default	1440
Minimum	480
Maximum	1440
Required	No
Dynamic	Yes
Description	Interval that StorHouse/RFS reviews file instances in memory. StorHouse/RFS removes stale file instances based on this interval. A file becomes stale once it has not been touched (for example, opened) since the last file instance cleanup. Once a file is removed from cache, it remains available in the StorHouse/RFS file system. The system automatically adds the file back to the cache on access.

Flush

Format	Flush=<WRITE | CLOSE | NO | SYSTEM>
Example	Flush=NO
Default	NO
Required	No
Dynamic	No
Description	Option that determines when StorHouse/RFS sends the flush buffers command to the operating system for files written to the staging area. This command causes StorHouse/RFS to complete writing files from memory buffers to the staging area.

- WRITE flushes the buffers after each write request. This setting is the safest (improves the chances of recovering a file in the event of a failure) but the slowest.
- CLOSE flushes the buffers when a file is closed. This setting is faster but provides a window where data in the system cache could be lost in the unlikely event that the StorHouse/RFS server crashes.
- NO flushes the buffers by the operating system as appropriate for current system operating conditions. This setting provides the fastest performance but the longest window of vulnerability.
- SYSTEM is a synonym of NO.

Note the following:

- StorHouse/RFS automatically flushes an .ldr file (the file containing file locator data for a local collection) within two minutes of collecting.
- As part of the collection process, the software does not specifically flush data files according to a time schedule.
- StorHouse/RFS removes a file from the staging area only after the file is collected and safely flushed from the buffer.

LogFile

Format	LogFile=<fully qualified file name>
Example	LogFile=/rfs/logs/rfs.log
Default	None
Required	Yes
Dynamic	No
Description	Fully qualified path and file name of the StorHouse/RFS log file. StorHouse/RFS creates log entries when accumulating files to local collections, writing StorHouse collections, and performing local searches and StorHouse searches. SGI recommends that you do not change the default LogFile value unless that file system is insufficient to handle the size of log files. If you need to change this parameter, please contact SGI Customer Support before making any modifications.

MaxActiveIOsFile

Format	MaxActiveIOsFile=< max active IOs per file>
Example	MaxActiveIOsFile=9
Default	9
Minimum	1
Maximum	MaxCommandThreads - 1
Required	No
Dynamic	Yes
Description	The maximum number of read/write operations to process in parallel per user file. This number should be reduced if a large number of files are typically being ingested or retrieved at the same time. Reducing this value will help prevent reads and writes for other files from being delayed (in IOP Queue) due to running out of command threads.

MaxCacheSpace

Format	MaxCacheSpace=<numeric in units of MB>
Example	MaxCacheSpace=1000
Default	None
Minimum	1000
Maximum	64000
Required	Yes
Dynamic	No
Description	Maximum amount of storage (in MB) to use for caching files retrieved from StorHouse. Cache uses a least recently used/most recently used scheme to manage data. When space is needed to cache a record read from StorHouse, StorHouse/RFS replaces the least recently used record with the new one. ■ If you specify 0 or a value under 1000 MB, StorHouse/RFS reserves 1000 MB. ■ SGI recommends that you set MaxCacheSpace to 25% of system memory or less.

MaxSMFiles

Format	MaxSMFiles=<numeric>
Example	MaxSMFiles=64
Default	64
Minimum	32
Maximum	128 (not currently enforced)
Required	No
Dynamic	Yes
Description	Maximum number of StorHouse connections that can be opened at the same time for reading or writing files. MaxSMFiles is defined on a system basis. One StorHouse connection is used for reading/writing any number of user files in a StorHouse collection–that is, one connection per StorHouse collection. When the number of connections is exceeded and a StorHouse connection is needed, StorHouse/RFS closes the least recently used connection. In a multiple StorHouse/RFS server environment, the sum of all MaxSMFiles parameters in all RFS servers accessing the StorHouse should not exceed the StorHouse VRAM_NUM_KU system parameter for that StorHouse.

MaxSMWriters

Format	MaxSMWriters=<numeric>
Example	MaxSMWriters=4
Default	4
Minimum	0
Maximum	32 (limited to the sum of all MaxSMWriters if the system does not contain tape drives)
Required	No
Dynamic	Yes
Description	Maximum number of StorHouse connections that may be used for writing collections to StorHouse. MaxSMWriters is defined on a system basis. This parameter helps prevent a busy system from using all connections for writing data when some are needed for retrieving data. ■ The value of MaxSMWriters cannot exceed the MaxSMFiles value. ■ Setting MaxSMWriters to 0 prevents writing data to StorHouse but not to the StorHouse/RFS server. When you are ready to resume ingest to StorHouse, you must reset MaxSMWriters to an acceptable number. ■ RFS currently does not enforce the maximum. In practice, the overall MaxSMWriters should not exceed the number of tape drives configured (in the libraries where RFS VSETs reside). If the system does not contain tape drives, MaxSMWriters should not exceed 32. ■ In a multiple StorHouse/RFS server environment, the sum of all MaxSMWriters parameters in all RFS servers accessing the StorHouse should not exceed the StorHouse VRAM_NUM_KW system parameter for that StorHouse.

SafetyPath

Format	SafetyPath=<fully qualified path>
Example:	SafetyPath=/var/sgi/rfs
Default:	/var/sgi/rfs
Required	No
Dynamic	No
Description	Fully qualified path to contain a secondary copy of metadata for user file changes (such as renames, deletes, security updates) that are not associated with the current local collection. StorHouse/RFS uses these safety copies in the event it detects a corrupt load (.ldr) file and has to rebuild it. StorHouse/RFS removes the entries from the safety path directory after successfully loading the entries into a StorHouse table. StorHouse/RFS fully manages the content of this directory. Always assign SafetyPath to a different file system from CollectionDir in case the CollectionDir file system runs out of space.

SpaceAvailable

Format	SpaceAvailable=<numeric in units of MB>
Example	SpaceAvailable=0
Default	0
Minimum	None
Maximum	None
Required	No
Dynamic	Yes
Description	Amount of file space that the operating system sees as available for ingesting more data into StorHouse/RFS. If you omit this parameter, StorHouse/RFS will return 100% of the amount of space configured in the smallest collector that is writable by the caller. Note that in any case, StorHouse/RFS will always return space used as 0.

RFS_PIT section

The RFS_PIT section is for StorHouse/CCi use only. It contains the restricted parameters OwnedBy and PitOs. Refer to Appendix A for a description of these parameters.

RFSPROFILE section

The RFSPROFILES section is for StorHouse/CCi use only. It contains the restricted parameters CreateTime, Name, UpdatedTime, and Version and the deprecated parameter ProfileID. Refer to Appendix A for a description of these parameters.

AUDITLOG section

The audit log section contains parameters that control StorHouse/RFS audit logging.

The parameters in the AUDITLOG section are:

- FileAuditPath
- MaxLoadInterval
- Storage

FileAuditPath

Format	FileAuditPath=<fully qualified path>
Example	FileAuditPath=
Default	Blank to indicate no audit log recording
Required	Only if auditing is desired
Dynamic	No
Description	Fully qualified path where StorHouse/RFS will log the file audit/hash records. Using file audit can result in a mild to a more significant performance impact depending on the quantity and types of auditing and hashing selected. Specifying FileAuditPath in the AUDITLOG section overrides one specified in the RFS section.

MaxLoadInterval

Format	MaxLoadInterval=<numeric in units of minutes>
Example	MaxLoadInterval=1440
Default	1440
Minimum	10
Maximum	100000
Required	No
Dynamic	Yes
Description	Amount of time in minutes between bulk loads of the file audit log to the audit log database. SGI recommends using the default value of 1440 (1 day).

Storage

Format	Storage=<storage definition name>
Example	Storage=
Default	Left blank signifying no audit database is selected
Required	No, unless you want to load the audit log records to the audit log database
Dynamic	No
Description	Name of the storage definition that will contain the audit log database. Multiple collections and StorHouse/RFS systems can use the same storage definition.

Cache Section

The Cache section is for Customer Support use only. It contains the deprecated parameters Mode and DirWait and the restricted parameters MAX_RFS_FILE_LIST, and TGT_RFS_FILE_LIST. Refer to Appendix A for information about these parameters.

Exclusions section

The EXCLUSIONS section lists file masks to be excluded from searches and collections. There are no parameters in this section, only values such as AUX*, MIDI*, WAVE*, *.DRV, *.INI, *.DLL, *.EXE, *.INF, THUMBS*, and ~*.TMP and -$*.DOC.

Note: Certain Windows applications, such as Media Player, will have extensive delays starting if you omit the following file masks from the exclusions list: MIXER*, AUX*, MIDI*, WAVE*, *.DRV, *.INI, *.DLL, *.EXE, *.INF, THUMBS*, and ~*.TMP.

Here's how the EXCLUSTIONS section works. While ingesting each file, StorHouse/RFS checks the file Against each mask in the exclusions list to determine whether the file being ingested should be excluded. If a match is found, the file is not migrated into a collection but remains in the staging area. It is the user's

responsibility to maintain housekeeping in the staging directory for excluded files. RFS will never remove excluded files.

If a user request is made for a file matching the excluded mask, RFS checks the staging directory to see if the file still resides on it. If the file is not in the staging area, StorHouse/RFS reports FILE NOT FOUND. If the file is found, it is returned to the user as normal. Using an exclusions list speeds processing for those applications that request support files (such as DLLs) when reading files. Exclusions can speed up access because the files are kept locally and do not have to be recalled from storage or tape on StorHouse/SM. If you do not want to exclude any files from searches or collections, do not provide any file masks in the EXCLUSIONS section. If you do, simply add the file masks.

FTPD section

The FTPD section provides configuration information for StorHouse/FTP. This section is required only if you are using StorHouse/FTP to retrieve files. StorHouse/FTP is also referred to as “Rapid Recall.”

The parameters in the FTPD section are:

- CertsDir
- EnableFTP
- FtpDebug
- MaxFTPSMReaders
- MinSizeSeqMode
- SecureLoginsOnly
- SecureXFRsOnly

CertsDir

Format	CertsDir=<fully qualified path>
Example	CertsDir=/usr/local/ssl/certs
Default	/usr/local/ssl/certs
Required	No
Dynamic	Yes
Description	Specifies the UNIX-style pathway to secure certificate and key files on the StorHouse/RFS Linux server. StorHouse/RFS uses these certificates and key files to support secure FTP (FTPS) sessions between the StorHouse/RFS server and remote clients. If you omit this parameter or specify it without a value, StorHouse uses the default.

EnableFTP

Format	EnableFTP=<YES | NO>
Example	EnableFTP=NO
Default	None
Required	Yes
Dynamic	Yes
Description	Controls the availability of the StorHouse/FTP feature. YES indicates the StorHouse/FTP feature is turned on. NO indicates the feature is turned off.

FtpDebug

Format	FtpDebug=<0 | 1>
Example	FtpDebug=0
Default	0
Required	No
Dynamic	Yes
Description	Controls the level of FTP session information logging reported to the StorHouse/RFS log file. (The LogFile parameter in the RFS section of the StorHouse/RFS configuration file defines the log location.) You can also control FTP debugging in a session-only manner by passing the command `site debug <value>` in an established StorHouse/RFS FTP session. In this command, use a value of 1 to enable logging and a value of 0 to disable it.

MaxFTPSMReaders

Format	MaxFTPSMReaders=<numeric>
Example	MaxFTPSMReaders=5
Default	5
Minimum	1
Maximum	64
Required	No
Dynamic	Yes
Description	Controls the maximum number of allowable concurrent FTP retrievals from StorHouse. If you omit this parameter or specify it without a value, StorHouse uses the default value.

MinSizeSeqMode

Format	MinSizeSeqMode=<numeric in units of MB>
Example	MinSizeSeqMode=2000
Default	2000
Minimum	0
Maximum	2147483647
Required	No
Dynamic	Yes
Description	Controls file access method optimization. Small files are retrieved using record-level I/O, which fetches exactly the blocks containing the file from the storage device. Large files are retrieved using sequential I/O, which is faster and more efficient for large amounts of data but may read more blocks than absolutely required. Files are considered large if their size equals or exceeds the value in this parameter. If you omit this parameter or specify it without a value, StorHouse uses the default value. If zero is specified, all files are retrieved using the sequential method.

SecureLoginsOnly

Format	SecureLoginsOnly=<YES | NO>
Example	SecureLoginsOnly=NO
Default	NO
Required	No
Dynamic	Yes
Description	Controls whether StorHouse/RFS allows unencrypted logins (for example, clear text passwords on an unsecured connection). If you omit this parameter or specify it without a value, StorHouse/RFS uses the default.

SecureXfersOnly

Format	SecureXfersOnly=<YES | NO>
Example	SecureXfersOnly=NO
Default	NO
Required	No
Dynamic	Yes
Description	Controls whether StorHouse/RFS allows file retrievals without a secure connection. If you omit this parameter or specify it without a value, StorHouse/RFS uses the default.

STATS section

The STATS section is an optional section required only to create and store StorHouse/RFS statistics in a local statistics file, in a StorHouse database, or both. If you do not want to collect statistics, you can omit the STATS section.

Note the following:

- To maintain StorHouse/RFS statistics in a file but not a database, the only required STATS parameter is FileType.
- To maintain StorHouse/RFS statistics in a database and a file, you must specify the FileType, Database, TableName, DBUserId, and DBPassword parameters.
- To store statistics in a database on a mirror StorHouse system, you must also specify the MirrorName parameter.
- If the StatsInterval parameter is missing or set to 0 no statistics are written.

The parameters in the STATS section are:

- Database
- DBPassword
- DBUserID
- FileType
- MirrorName
- StatsInterval
- SystemName
- TableName

The STATS section also contains the restricted parameter PasswordType, which is discussed in Appendix A.

Database

Format	Database=<database name on StorHouse>
Example	Database=STATSDATABASE
Default	None
Required	No
Dynamic	No
Description	Name of the StorHouse database that will store StorHouse/RFS statistics. The tblgen utility creates this database for StorHouse/RFS systems. The database name is case sensitive.

DBUserId

Format	DBUserId=<StorHouse account>
Example	DBUserId=sjones
Default	None
Required	No
Dynamic	No
Description	StorHouse account ID used to log in to the StorHouse/RM system to load the statistics tables. This account ID must be the owner of the statistics tables or have the following minimum privileges: SQLEXECUTE StorHouse privilege and INSERT database component privilege on the statistics tables.

DBPassword

Format	DBPassword=<StorHouse password>
Example	DBPassword=*)&^*!
Default	None
Required	No
Dynamic	No
Description	StorHouse/RM account password (maximum 32 characters) associated with the DBUserId that is used to load the statistics tables. StorHouse/CCi encrypts the password when you create/update the password parameter in a StorHouse/RFS profile. Refer to Appendix A for more information about password encryption.

FileType

Format	FileType=<HTML | XML | TXT>
Example	FileType=HTML TXT
Default	None
Required	Yes
Dynamic	No
Description	Option to store statistics in a local statistics file in HTML, XML, text, or any combination of formats. Use a space to separate multiple formats, as shown in the example. If omitted, no local statistics file is created.

MirrorName

Format	MirrorName=<system definition name>
Example	MirrorName=ComplianceSet2
Default	None
Required	No
Dynamic	No
Description	Name of the system definition that identifies the secondary StorHouse system for storing statistics. StorHouse/RFS writes statistics to this StorHouse system after writing the data to the primary StorHouse system.

StatsInterval

Format	StatsInterval=<numeric in units of minutes>
Example	StatsInterval=0
Default	0
Minimum	0
Maximum	No practical limit
Required	No
Dynamic	Yes
Description	Interval used by StorHouse/RFS to capture and write statistics. If omitted or set to 0, no statistics are written.

SystemName

Format	SystemName=<system definition name>
Example	SystemName=ComplianceSet
Default	None
Required	No
Dynamic	No
Description	Name of the system definition that identifies the primary StorHouse system for storing statistics.

TableName

Format	TableName=<base_table_name>
Example	TableName=STATS
Default	None
Required	No
Dynamic	No
Description	Base name of the StorHouse tables used to store statistics. StorHouse/RFS adds a suffix (RFS, TABLES, and SYSTEMS) to the table name. For example, if the owner (DBUserId) is SYSADM and the base table name is STATS, then the complete table names are: ■ SYSADM.STATS_RFS ■ SYSADM.STATS_TABLES ■ SYSADM.STATS_SYSTEMS You use the tblgen utility to create the statistics tables. ■ The TableName in the StorHouse/RFS configuration file must match the table name provided when running the tblgen utility. ■ You must include the owner name as part of the table name specification only if the owner differs from the logon user (DBUserId).

UNIX section

The UNIX section is for Customer Support use only. It contains the restricted parameter SchedPolicy and the deprecated parameters TCP_Receive and TCP_Send, which are discussed in Appendix A.

COLLECTORS section

The COLLECTORS section lists each collector definition name and corresponding collection definition. The format of each line in this section is:

CollectorDefinitionName=CollectionDefinitionName

For example, the following COLLECTORS section identifies three collectors: DFFILES, BIGFILES, and RFS31. The same collection definition—DIRECT—is assigned to the first two collectors. The DIRECT31 collection definition is assigned to the RFS31 collector.

```
[COLLECTORS]
DFILES=DIRECT
BIGFILES=DIRECT
RFS31=DIRECT31
```

System definition

A system definition identifies the StorHouse system to use for storing StorHouse collections and statistics. For a StorHouse collection, each system definition is a unique destination defined by StorHouse system name, group name, VSET name, and FSET name. Any data destined for different VSETs and/or FSETs on a single StorHouse system or any data destined for different StorHouse systems requires a unique system definition in the StorHouse/RFS configuration file.

The parameters in a system definition are:

- DNSName
- FSET
- FSETSegments
- MailRecipient
- RetryInterval
- MaxSMFiles
- MaxSMWriters
- SMGroup
- SMPassword
- SMUserID

The System definition also contains the restricted parameters DBDriver, DBHost, DBPort, PasswordType and STHPort and the deprecated parameters Checkpoint and SystemID, which are discussed in Appendix A.

DNSName

Format	DNSName=<any string>
Example	DNSName=alpha2
Default	None
Required	Yes
Dynamic	No
Description	DNS name of the StorHouse system. This value can also be an IP address if the DBDriver is not specified.

FSET

Format	FSET=<any string>
Example	FSET= The example FSET is shown as blank, or empty, to indicate the default.
Default	Default FSET for the account specified for the SMUserId parameter
Required	No
Dynamic	No
Description	Name of the StorHouse file set that will contain StorHouse collections written to this StorHouse system. If you are using a set of FSETs, that is, you specify a value for the FSETSegments parameter, then you must specify a value for the FSET parameter.

FSETSegments

Format	FSETSegments=<numeric>
Example	FSETSegments=1
Default	1
Required	No
Dynamic	No
Minimum	0
Maximum	64
Description	Number of FSETs in the VSET used by all collectors that use this system definition. StorHouse/RFS rotates through all these FSETs when writing collections. You can use this parameter to spread data across more tape volumes, which allows more concurrency while writing tapes, and, in some use cases, may reduce conflicts from concurrent retrieval activity. To ensure distribution of data across tapes you must define these FSETs to StorHouse as "CONTIGUOUS." ■ If you omit values for the FSET name and FSETSegments parameters, StorHouse/RFS uses the default FSET for the account. ■ If you specify a value of 0 for FSETSegments, StorHouse/RFS assumes a value of 1, meaning only 1 FSET will be used. ■ Multiple FSETs can reduce contention for a tape volume and also increase contention for tape drives. For this reason, the value of FSETSegments should be a fraction (usually half) of the tape drives in the system, and not more than one less than the tape drives in the system or 1, whichever is higher. This parameter can affect performance and should be chosen carefully. SGI Customer Support can help you select the optimum value.

SMGroup

Format	SMGroup=<any string>
Example	SMGroup= The example SMGroup is shown as blank, or empty, to indicate the default.
Default	Default group for the account specified for the SMUserId parameter
Required	No
Dynamic	No
Description	Name of the StorHouse file access group used by StorHouse/RFS to access files on this StorHouse system.

MailRecipient

Format	MailRecipient=<email address>
Example	MailRecipient=mjones@cc.com
Default	None
Required	No
Dynamic	No
Description	E-mail address where StorHouse/RFS sends messages when an event occurs. Example events are: ■ StorHouse/RM goes down and comes up. ■ StorHouse/SM goes down and comes up. ■ A particular collection cannot be written to StorHouse/SM for the first time. ■ The collection that could not be written is successfully written. ■ The file locator data for a particular collection cannot be loaded into StorHouse/RM for the first time. ■ The file locator data that could not be loaded is successfully loaded. ■ The staging space is full. ■ The collection space is full. If you omit this value and/or the MailServer value in the RFS section, no email is sent.

MaxSMFiles

Format	MaxSMFiles=<numeric>
Example	MaxSMFiles=64
Default	64
Minimum	32
Maximum	128 (not currently enforced)
Required	No
Dynamic	Yes
Description	Maximum number of StorHouse connections that can be opened at the same time for reading or writing files. MaxSMFiles is defined on a system basis. One StorHouse connection is used for reading/writing any number of user files in a StorHouse collection–that is, one connection per StorHouse collection. When the number of connections is exceeded and a StorHouse connection is needed, StorHouse/RFS closes the least recently used connection. In a multiple StorHouse/RFS server environment, the sum of all MaxSMFiles parameters in all RFS servers accessing the StorHouse should not exceed the StorHouse VRAM_NUM_KU system parameter for that StorHouse.

MaxSMWriters

Format	MaxSMWriters=<numeric>
Example	MaxSMWriters=4
Default	4
Minimum	0
Maximum	32 (limited to the sum of all MaxSMWriters if the system does not contain tape drives)
Required	No
Dynamic	Yes
Description	Maximum number of StorHouse connections that may be used for writing collections to StorHouse. MaxSMWriters is defined on a system basis. This parameter helps prevent a busy system from using all connections for writing data when some are needed for retrieving data. ■ The value of MaxSMWriters cannot exceed the MaxSMFiles value. ■ Setting MaxSMWriters to 0 prevents writing data to StorHouse but not to the StorHouse/RFS server. When you are ready to resume ingest to StorHouse, you must reset MaxSMWriters to an acceptable number. ■ RFS currently does not enforce the maximum. In practice, the overall MaxSMWriters should not exceed the number of tape drives configured (in the libraries where RFS VSETs reside). If the system does not contain tape drives, MaxSMWriters should not exceed 32. ■ In a multiple StorHouse/RFS server environment, the sum of all MaxSMWriters parameters in all RFS servers accessing the StorHouse should not exceed the StorHouse VRAM_NUM_KW system parameter for that StorHouse.

SMPassword

Format	SMPassword=<any string>
Example	SMPassword=*)&^*!
Default	None
Required	Yes
Dynamic	No
Description	StorHouse account password (maximum 32 characters) associated with the SMUserId that is used to write StorHouse collections. StorHouse/CCi encrypts the password when you create/update the password parameter in a StorHouse/RFS profile. Refer to Appendix A for more information about password encryption.

RetryInterval

Format	RetryInterval=<numeric in units of minutes>
Example	RetryInterval=3
Default	3
Minimum	1
Maximum	10 (currently not enforced)
Required	No
Dynamic	Yes
Description	Number of minutes that StorHouse/RFS should wait before attempting to connect to a StorHouse/SM system or to a StorHouse/RM system that is unresponsive, or down. ■ StorHouse/RFS performs separate checks for StorHouse/SM and for StorHouse/RM. In other words, if StorHouse/SM is up but StorHouse/RM is down, then the retry interval applies to StorHouse/RM. If both systems are down, then the retry interval applies to both systems.

- StorHouse/RFS marks a system as down after five consecutive failed write or retrieval requests. After the RetryInterval interval has expired, StorHouse/RFS attempts to connect to the unresponsive system and if still down, marks the system as down and tries again at the next RetryInterval interval.

SMUserId

Format	SMUserId=<any string>
Example	SMUserId=RFSUSER
Default	None
Required	Yes
Dynamic	No
Description	StorHouse account ID used by StorHouse/RFS for logging in to this StorHouse system to write StorHouse collections. This account must have the following StorHouse privileges to write StorHouse collections: ATF, DELETE, GET, PUT, RECORD, SETGROUP, SHOW, and VTF.

VSET

Format	VSET=<any string>
Example	VSET= The example VSET is shown as blank, or empty, to indicate the default.
Default	Default VSET for the account specified for the SMUserId parameter
Required	No If you omit this parameter, make sure there is a valid default volume set for the StorHouse/SM account.
Dynamic	No
Description	Name of the StorHouse volume set that will contain StorHouse collections written to this StorHouse system.

VTF

Format	VTF=<DIRECT | NOW | NEXT>
Example	VTF=NEXT
Default	NEXT
Required	No
Dynamic	No
Description	Option that determines when StorHouse/RFS writes collections to their resident FSETs as well as whether to create performance copies of collections in the performance buffer. VTF stands for Vulnerability Time Factor. ■ DIRECT – Bypass the performance buffer and write the collection directly to the primary FSET. ■ NOW – Write the collection to the performance buffer and then copy the collection to the primary FSET immediately. ■ NEXT – Write the collection to the performance buffer and then copy the collection to the primary FSET at the next StorHouse write-back operation.

Storage definition

A storage definition contains information for storing file locator data and collection metadata. Multiple collections can use the same storage definition.

The parameters in a storage definition are:

- Database
- MaxSearchConnections
- MirrorName
- DBPassword
- DBUserID
- ReadOnly
- RefreshTimeout
- SearchConnectionTimeout
- SystemName
- TableName

The Storage definition also contains the restricted parameter PasswordType and the deprecated parameter TimeoutOverride, which are discussed in Appendix A.

Database

Format	Database=<database name on StorHouse>
Example	Database=RFSDATA
Default	None
Required	Yes
Dynamic	No
Description	Name of the StorHouse database that contains the set of StorHouse tables identified at the TableName parameter. The database name is case sensitive.

DBPassword

Format	DBPassword=<StorHouse password>
Example	DBPassword=*)&^*!
Default	None
Required	Yes
Dynamic	No
Description	StorHouse account password (maximum 32 characters) associated with the DBUserId that is used to load and query the StorHouse tables. StorHouse/CCi encrypts the password when you create/update the password parameter in a StorHouse/RFS profile. See Appendix A for more information about encrypted passwords.

DBUserId

Format	DBUserId=<StorHouse account>
Example	DBUserId=SYSADM
Default	None
Required	Yes
Dynamic	No
Description	StorHouse account ID used to log in to the StorHouse system to load file locator data and collection metadata and to query the StorHouse tables. This account must be the owner of the StorHouse table specified at the TableName parameter.

MaxSearchConnections

Format	MaxSearchConnections=<numeric>
Example	MaxSearchConnections=2
Default	2
Minimum	1
Maximum	8. Currently, RFS does not enforce the maximum.
Required	No
Dynamic	Yes
Description	Maximum number of concurrent ODBC connections that can be used at any time for accessing the StorHouse tables to load or query file locator data and collection metadata.

MirrorName

Format	MirrorName=<system definition name>
Example	MirrorName=mailboxset2
Default	None
Required	Yes, if using a secondary StorHouse system for StorHouse/RFS duplexing
Dynamic	No
Description	Name of the system definition that identifies the secondary StorHouse system. StorHouse/RFS writes data to this StorHouse system after writing the data to the primary StorHouse system.

ReadOnly

Format	ReadOnly=<YES | NO>
Example	ReadOnly=NO
Default	NO
Required	No
Dynamic	Yes
Description	Determines whether a StorHouse/RFS server treats this storage definition as “read-only.” If set to NO, the storage definition allows reads and writes. If set to YES, the storage definition allows reads but not writes. If RefreshTimeout is greater than zero, ReadOnly defaults to YES.

RefreshTimeout

Format	RefreshTimeout=<numeric in units of minutes>
Example	RefreshTimeout=0
Default	0
Minimum	0
Maximum	No maximum
Required	No
Dynamic	Yes
Description	The number of minutes StorHouse/RFS caches directory information before checking the database for updates. RefreshTimeout and ReadOnly are used to configure a StorHouse/RFS read-only instance for a particular storage definition (for example, StorHouse/RFS refresh). Define this parameter only on StorHouse/RFS read-only servers. A value of 0 indicates that refresh operations are disabled.

SearchConnectionTimeout

Format	SearchConnectionTimeout=<numeric in units of minutes>
Example	SearchConnectionTimeout=10
Default	10
Minimum	1
Maximum	1440 (Enforced by CCi)
Required	No
Dynamic	Yes
Description	Length of time an ODBC connection must be idle before it is released. This parameter is used only if MaxSearchConnections is specified. If you set SearchConnectionTimeout to 0, StorHouse/RFS uses the default value of 10.

SystemName

Format	SystemName=<system definition name>
Example	SystemName=mailboxset
Default	None
Required	Yes
Dynamic	No
Description	Name of the system definition that identifies the primary StorHouse system to contain file locator data and collection metadata.

TableName

Format	TableName=<base_table_name>
Example	TableName=MAILBOXTABLE
Default	None
Required	Yes
Dynamic	No
Description	Base table name of the StorHouse tables to contain file locator data and collection metadata. You use the tblgen utility to create the tables. ■ The TableName parameter in the StorHouse/RFS configuration file must match the base table name you provide when running the tblgen utility. ■ You must include the owner name as part of the table name specification only if the owner differs from the logon user (DBUserId).

Collection definition

A collection definition defines collection options. Each collection requires a separate collection definition. The parameters in the collection definition are:

- CollectionDir
- Compression
- FileAudit
- FileAuditHash
- FileVersionLImit
- MaxLoadInterval
- MaxWriteSize
- Retention
- Storage

The collection definition also contains the deprecated parameters Browse and CollectorID, which are described in Appendix A.

CollectionDir

Format	CollectionDir=<fully qualified path>
Example	CollectionDir=/rfs/collection/coll123
Default	None
Required	Yes
Dynamic	No
Description	Fully qualified path where StorHouse/RFS creates file locator data, or load (.ldr) files. This directory can be on the StorHouse/RFS server or on a device accessible to the server.

Compression

Format	Compression=<YES | NO>
Example	Compression=NO
Default	NO
Required	No
Dynamic	No
Description	Option to compress the StorHouse collections. Compression rates vary by the type of source data written to StorHouse/RFS. Compression costs CPU time but reduces storage costs and network bandwidth requirements. If you have available CPU time and want to reduce the amount of storage required on StorHouse, then specify COMPRESSION=YES. No compression occurs if the value is NO or if you omit the parameter from the StorHouse/RFS configuration file. Some file types such as video, sound, and archive files are already compressed. Attempts to compress them further can result in a larger file while also taking more CPU time.

FileAudit

Format	FileAudit=<file_audit_type>
Example	FileAudit=
Default	FileAudit= (A blank value indicates no audit log recording)
Required	No
Dynamic	Yes
Description	Determines the types of file and directory actions to be logged. For example, if you specify R, StorHouse/RFS will create an audit log record when a file is renamed. You may specify multiple file audit types with no delimeter between values. Valid values are: ■ A – Access ■ C – Create ■ D – Delete ■ M – Modify ■ R – Rename ■ S – Set

FileAuditHash

Format	FileAuditHash=<hash_type>
Example	FileAuditHash=
Default	FileAuditHash= (A blank value indicates no file hashing)
Required	No
Dynamic	Yes
Description	Determines the types of hashing to be performed on files written to StorHouse. You may specify multiple hash types as comma-separated values. Valid values are: ■ MD5 SHA256 ■ SHA SHA384 ■ SHA1 SHA512 ■ SHA224 RMD160

FileVersionLimit

Format	FileVersionLimit=<numeric>
Example	FileVersionLimit=0
Default	0
Minimum	0
Maximum	1000 (Currently not enforced)
Required	No
Dynamic	Yes
Description	Sets a limit for the maximum number of versions StorHouse/RFS will save for each file. By default, StorHouse/RFS creates a new version of a file every time the file is collected and keeps these versions until the file is deleted. FileVersionLimit puts a cap on the number of versions StorHouse/RFS can keep. Once the limit is reached, StorHouse/RFS deletes files starting with the oldest version until the number of version for a file is equal to the limit. StorHouse/RFS does not delete files that are still in their retention period. A value of 0 tells StorHouse to keep all versions of a file until the file is deleted.

MaxLoadInterval

Format	MaxLoadInterval=<numeric in units of minutes>
Example	MaxLoadInterval=1440
Default	1440
Minimum	30
Maximum	1440 (Currently not enforced)
Required	No
Dynamic	Yes
Description	Longest time to wait between loads (writing StorHouse collections and inserting file locator data into StorHouse tables), regardless of how much data has been collected. StorHouse/RFS uses this parameter in conjunction with MaxWriteSize to determine when it writes StorHouse collections. The interval starts when a collector collects the first file in the local collection. ■ If you specify a value less than 30 (for instance, 5 or 10), StorHouse/RFS sets the MaxLoadInterval to 30. ■ If you specify 0, StorHouse/RFS uses the default value (1440, or one day).

MaxWriteSize

Format	MaxWriteSize=<numeric in units of MB>
Example	MaxWriteSize=1800
Default	1800
Minimum	1
Maximum	1800
Required	No
Dynamic	Yes
Description	Largest size a local collection can get before StorHouse/RFS closes it and starts a new one. ■ StorHouse/RFS places any file larger than the MaxWriteSize into its own collection. ■ StorHouse/RFS places any file smaller than MaxWriteSize into a local collection along with the other files smaller than the MaxWriteSize value. StorHouse/RFS continues to add smaller files to the collection until the total collection size reaches the MaxWriteSize. ■ The maximum number of files allowed in a collection is 200,000. If a local collection has not exceeded the MaxWriteSize but contains more than 200,000 files, StorHouse/RFS closes the current collection and starts a new one.

Retention

Format	Retention=<-1 | FOREVER | number of days>
Example	Retention=0
Default	0
Minimum	0
Maximum	FOREVER
Required	No
Dynamic	No
Description	Number of days to retain a file before it may be deleted or overwritten. When a file is referenced (for example, there is an attempt to write, delete or rename the file), StorHouse/RFS determines the expiration date by adding the retention days to the file's last modified time provided by the operating system. If the last modified time 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=10, then the retention expires at noon on February 25. A user may delete, modify, or overwrite a file after the expiration. The retention number may be 0 through 65000 days (or 0 years and 0 days through 178 years). ■ If you specify 0 (the default), no retention period applies to files in the collection. Files may be deleted after they are collected and new file versions may be created. ■ If you specify a number greater than 0, no file versions are allowed. A file may be deleted, modified, or overwritten only after the retention period expires. ■ If you specify FOREVER, the file may not be deleted, modified, or overwritten. ■ A value of -1 is a synonym for FOREVER. You can change the retention 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—that is, change a retention period to 0. In this case, StorHouse/RFS may create new file versions and allow existing file versions to be deleted with the exception of the original file version.

Storage

Format	Storage=<storage definition name>
Example	Storage=STORAGE_RFS31
Default	None
Required	Yes
Dynamic	No
Description	Name of the storage definition that contains specifications for storing file locator data and collection metadata. Multiple collections can use the same storage definition.

Collector definition

A collector definition defines each StorHouse/RFS collector, including where the collector looks for files to collect. You assign each collector to a collection definition. You can assign multiple collectors to the same collection definition.

The parameters in the collector definition are:

- DeleteExcluded
- Export
- ExportClients
- MaxCollectionSpace
- MaxFileSizePct
- MaxStagePct
- MaxStagingSpace
- MaxThrottleDelay
- Permissions
- StagingDir
- UserDir
- WaitTime

The Collector definition also contains the deprecated parameter KeepSubdirectories, which is described in Appendix A.

DeleteExcluded

Format	DeleteExcluded=<YES | NO>
Example	DeleteExcluded=No
Default	No
Required	No
Dynamic	Yes
Description	If an EXCLUSIONS section has been defined in the rfs.cfg file, this parameter controls whether RFS will delete excluded files. DeleteExcluded appears in both the RFS section and the collector definition. The value in the collector definition overrides the one in the RFS section.

Export

Format	Export=<YES|NO|Other valid values>
Example	Export=NO
Default	NO
Required	No
Dynamic	No
Description	Defines whether a collector can be exported, and the export options. Valid Values are:

- Yes – RFS exports the collector with root_squash and rw options.
  - This is a change from previous RFS behavior where no_root_squash was the default.
  - If you omit YES and specify export options, RFS assumes YES.
- No, or blank – RFS does not export the collector.
- Comma separated list of options on the RFS server. The text line may include up to 4096 bytes.

Export options new in the current release include:

- root_squash - Converts the uid and gid of a root user on the client to the anonymous uid/gid. In other words, root_squash removes the superuser rights of root on the client from the server to keep the client from having superuser rights across server mounts.
- no_root_squash - Allows root on a client to access RFS as root (current RFS behavior).
- ro - Exports the collector as read-only.
- rw - Exports the collector as read-write (current RFS behavior).
- anonuid - Overrides the UID of nfsnobody on the RFS server when root squashing. Selecting this option enables you to assign file ownership to a specific user with a UID.
- anongid - Overrides the GID of nfsnobody on the RFS server when root squashing. Selecting this option enables you to assign the group portion of file security to the given GID.

ExportClients

Format	ExportClients=<blank separated list of client specification(s)> Each specification may be followed by a list of export options (only for that client) in parenthesis following the client name.
Example	ExportClients=*.sgi.com ExportClients=devclient4 linxsth(ro,no_root_squash) ExportClient=linxdev linxsth(no_root_squash) In the third example, RFS will export the collector to two clients: linxdev with the options "root_sqash,rw" and linxsth with the options "no_root_squash,rw".
Default	When Export for this collector is set to YES, the default is rw,root_squash. In previous releases, it was no_root_squash.
Required	No
Dynamic	No
Description	Defines the clients that can mount the collector. Format choices are : ■ Blank or missing – RFS exports the collector to all clients. ■ Single host (most common format) – Exports to a single host specified as an abbreviated name recognized by the resolver, the fully qualified domain name, or an IP address. ■ NIS Netgroups – Exports to @group. RFS considers only the host part of each netgroup member for membership checking and ignores empty host parts or those containing a single dash (-). ■ Wildcards – Exports to a machine name(s) containing the wildcard characters “*” and “?”. Using this option makes the exports file more compact (for instance, *.cs.foo.edu matches all hosts in the domain cs.foo.edu). Because these characters also match the dots in a domain name, the given pattern will match all hosts within any subdomain of cs.foo.edu. ■ IP networks – Exports to an IP address and netmask pair (address/netmask) where netmask can be specified in dotted-decimal format or as a contiguous mask length. For example, appending either “/255.255.252.0” or “/22” to the network base address results in identical subnetworks with10 bits of host.

- Wildcards generally do not work on IP addresses although they may work by accident when reverse DNS lookups fail.
- You can use the IP address format to export directories to all hosts on an IP (sub-) network simultaneously.

MaxCollectionSpace

Format	MaxCollectionSpace=<numeric in units of MB>
Example	MaxCollectionSpace=1000
Default	None
Minimum	1000
Maximum	None
Required	Yes
Dynamic	Yes
Description	Maximum amount of disk space, in gigabytes, that StorHouse/RFS can use to collect files for this collector. The sum of all MaxCollectionSpace parameters is the amount of rename space that can be used by all collectors regardless of the staging directory. All files are collected by renaming them from the staging area to a rename directory, which StorHouse/RFS creates automatically. The rename directory appears at the same level as the staging directory but includes the name of the collection that it is creating. If the rename space becomes full, StorHouse/RFS performs cleanup routines or stops renaming files to the rename directory until it writes the local collection to StorHouse.

MaxFileSizePct

Format	MaxFileSizePct=<percent>
Example	MaxFileSizePct=100
Default	100
Minimum	20 (Currently not enforced)
Maximum	100
Required	No
Dynamic	Yes
Description	Trigger that causes an out-of-space error message when a single file reaches the specified percent of staging space.

MaxStagePct

Format	MaxStagePct=<percent>
Example	MaxStagePct=100
Default	100
Minimum	20 (Currently not enforced)
Maximum	100
Required	No
Dynamic	Yes
Description	Trigger that causes an out-of-space error message when the specified percent of staging space is full.

MaxStagingSpace

Format	MaxStagingSpace=<numeric in units of MB>
Example	MaxStagingSpace=10000
Default	None
Minimum	10000
Maximum	None
Required	Yes
Dynamic	Yes
Description	Maximum amount of storage (in MB) to use for writing files to the staging area. When the maximum amount is reached, StorHouse/RFS returns an out of space condition appropriate for the operating system. Files may resume being written to the staging area only when space permits. Warning: The size of all staging areas must not exceed the size of the disk subsystem where they reside. ■ You set a MaxStagingSpace for each StagingDir and UserDir combination. For example, you set different MaxStagingSpace values for these two staging areas: /email/mailboxes and /email/journal. ■ Use a minimum value of 100000.

MaxThrottleDelay

Format	MaxThrottleDelay=<numeric in units of milliseconds>
Example	MaxThrottleDelay=1000
Default	1000
Minimum	None
Maximum	None
Required	No
Dynamic	Yes
Description	Delay that occurs each time StorHouse/RFS enacts throttling. For example, setting MaxThrottleDelay to 500 means that if StorHouse/RFS needs to throttle once, there will be a delay of 500 milliseconds before the software performs the next action. Throttling enables StorHouse/RFS to slow down when needed to allow writes to StorHouse to catch up.

Permissions

Format	Permissions=<R | W | X>
Example	Permissions=R
Default	None
Required	No
Dynamic	No
Description	Permissions to share files across StorHouse/RFS systems that have access to the same StorHouse table array. The values have the same meaning as the UNIX “Other” permissions and override any “Other” permissions set with standard UNIX commands. ■ R – Read a file and list files in a directory. ■ W – Modify a file and create and delete files in a directory. ■ X – Execute a file (for instance, when the file is a program) or access a directory (for instance, use cd). If you omit this parameter or specify it without a value, no other permissions are allowed, that is, only the file owner or group member may access files and directories collected by this collector. (Other permissions relate to the Windows “Everyone” group concept for file and directory access.) **Note**: The Permissions parameter is deprecated in RFS 5.2 and later releases.

StagingDir

Format	StagingDir=<fully qualified path>
Example	StagingDir=/rfs/collectors/staging
Default	None
Required	Yes
Dynamic	No
Description	Path where StorHouse/RFS stages files to be collected and where a collector checks for files to collect. The StagingDir can be the same for all collectors, or different for each collector, or any combination. This directory can be on the StorHouse/RFS server or on a device accessible to the server as long as the simultaneous read/write performance of the device is consistent with the performance expectations for the collector(s) using this staging area. The StagingDir must be located at least one level below the root level of a file system or drive. This is because StorHouse/RFS collects files by renaming them to the directory above the StagingDir. If StagingDir is on a different file system, the rename will result in copying the entire file. A user or application must have read and write permission to access the StagingDir to archive files. The full collection path is comprised of the StagingDir and the UserDir. For example, assume the following: ■ StagingDir is /email. ■ UserDir is /mailboxes. A collector then collects files in /email/mailboxes. StorHouse/RFS displays the UserDir, but not the StagingDir, in the virtual file system.

UserDir

Format	UserDir=<subdirectory under StagingDir>
Example	UserDir=/
Default	/
Required	No
Dynamic	No
Description	Subdirectory under the StagingDir where StorHouse/RFS stages files to be collected and where a collector checks for files to collect. Wildcard characters are not allowed in the UserDir, for instance, /mailboxes/* is invalid. ■ You can use the default (/) once per StorHouse/RFS configuration. It indicates to collect all files written to the StagingDir. ■ The UserDir value must be unique. ■ StorHouse/RFS creates a folder or directory for the UserDir in the virtual file system.

WaitTime

Format	WaitTime=<numeric in units of minutes>
Example	WaitTime=2
Default	None
Minimum	1
Maximum	None
Required	Yes
Dynamic	Yes
Description	Number of minutes a file must be idle in a staging area before it is eligible for collection. This parameter prevents files that are in the process of being written from being collected prematurely. The collector checks the last modified time of each file in the staging area, and when the file has aged, it renames the file to the rename directory. If this value is set to 0, StorHouse/RFS changes it to 10.

C H A P T E R 2

Maintaining the StorHouse/RFS configuration file

This chapter explains how to use StorHouse/CCi to maintain the StorHouse/RFS configuration file.

About StorHouse/RFS profiles

After StorHouse/RFS installation, SGI registers your StorHouse/RFS system in StorHouse/CCi and creates a StorHouse/RFS profile. A StorHouse/RFS profile is a tool for managing the properties, or operating parameters, of a StorHouse/RFS server. In other words, it is the graphical representation of the sections and definitions in the StorHouse/RFS configuration file.

After profile creation, SGI deploys, or assigns, the profile to a StorHouse/RFS server according to your site specifications. Once deployed, StorHouse/CCi converts the profile to configuration file text. The rfs.cfg file resides in the /rfs/files directory.

Refer to Appendix B, “Using the StorHouse/RFS configuration file in text format,” for a detailed description of the configuration file layout and syntax rules

Updating a StorHouse/RFS profile

At some point, you may need to update and redeploy a StorHouse/RFS profile to modify existing configuration file parameters.

While it is possible to use a text editor to update the StorHouse/RFS configuration file directly on your StorHouse/RFS server, SGI strongly recommends that you use StorHouse/CCi to update the corresponding StorHouse/RFS profile instead. In fact, never update the configuration file with a text editor unless SGI Customer Support specifically tells you to do so.

Refer to Appendix A, “Additional Information,” for a discussion of the problems that could occur if you use a text editor instead of StorHouse/CCi to update the StorHouse/RFS configuration file.

Use this procedure to update and deploy a StorHouse/RFS profile in StorHouse/CCi.

To update and deploy a StorHouse/RFS profile

1. Navigate to the StorHouse/CCi website.
2. On the StorHouse/CCi **Login** window, type your user ID and Password, and click **Enter**.

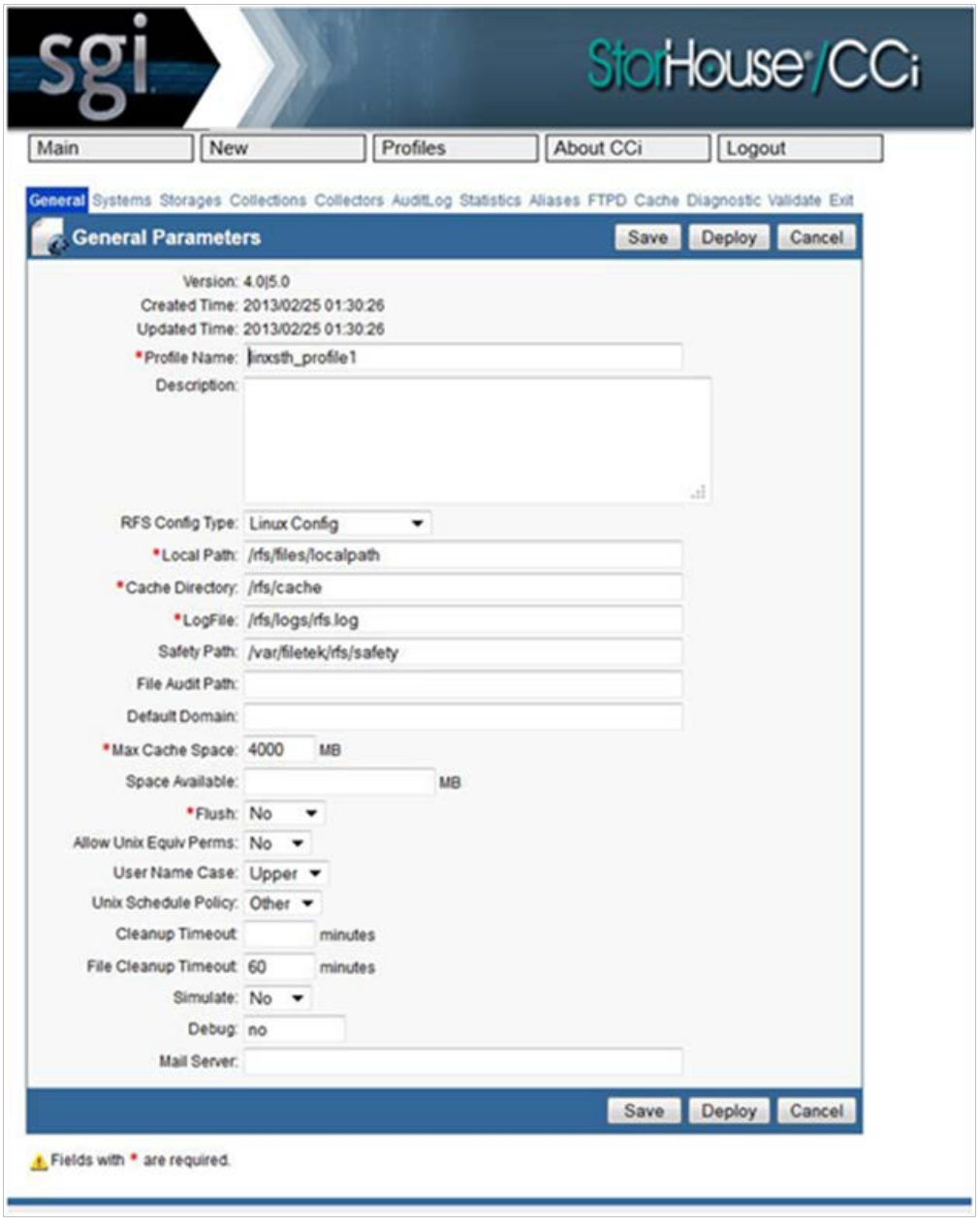
   The following example StorHouse/CCi Main window shows a site with two StorHouse/RFS server nodes (alpha3 and hprfs1) and one StorHouse storage server (hpsth1).

3. On the StorHouse/CCi **Main** page, click **Profiles** to display the list of available StorHouse/RFS profiles managed by this StorHouse/CCi system.

4. On the StorHouse/RFS **Profiles** page, click (edit) next to the StorHouse/RFS Profile you want to update and deploy. In this example, the profile to be updated and deployed is linxsth_profile1.

5. On the **RFS General Parameters** window, click the StorHouse/RFS section or definition name that contains the parameter(s) to be updated. (The StorHouse/CCi General tab indicates the RFS section of the configuration file).

6. Update the parameter(s), and click **Save**.


sgi
StorHouse/CCi
Main
New
Profiles
About CCi
Logout
General
Systems
Storages
Collections
Collectors
AuditLog
Statistics
Aliases
FTPD
Cache
Diagnostic
Validate
Exit
General Parameters
Save
Deploy
Cancel
Version: 4.0|5.0
Created Time: 2013/02/25 01:30:26
Updated Time: 2013/02/25 01:30:26
*Profile Name: linxsth_profile1
Description:
RFS Config Type: Linux Config
*Local Path: /rfs/files/localpath
*Cache Directory: /rfs/cache
*LogFile: /rfs/logs/rfs.log
Safety Path: /var/filetek/rfs/safety
File Audit Path:
Default Domain:
*Max Cache Space: 4000 MB
Space Available: MB
*Flush: No
Allow Unix Equiv Perms: No
User Name Case: Upper
Unix Schedule Policy: Other
Cleanup Timeout: minutes
File Cleanup Timeout: 60 minutes
Simulate: No
Debug: no
Mail Server:
Save
Deploy
Cancel
Fields with * are required.

7. Repeat steps 5 and 6 as necessary to update all parameters that require modification. Then proceed to step 8.

8. Click **Deploy** to display the list of StorHouse/RFS servers managed by StorHouse/CCi.

9. Select the checkbox next to the StorHouse/RFS system where you will deploy the updated profile (in this case, alpha3).

10. Click **Deploy** to deploy the updated StorHouse/RFS profile linxsth_profile1 to the StorHouse/RFS server alpha3, or click Cancel to cancel the update. StorHouse/CCi displays the following status window.

11. Click **Continue** or any other available option (Main, New, Profiles, About CCi, or Logout).

Note: After you update a profile, you must reread the StorHouse/RFS configuration file to implement changes to dynamic parameter and restart the StorHouse/RFS service to implement changes to static parameters. Refer to Appendix A for information about how to perform these tasks.

APPENDIX A

Additional information

This appendix contains the following additional information about StorHouse/RFS configuration file parameters and operating procedures:

- Deprecated parameters and sections
- Parameters for Customer Support use only
- Parameters for StorHouse/CCi use only
- Password encryption
- Potential problems associated with using a text editor to update the StorHouse/RFS configuration file

Deprecated parameters and sections

Table A-1 lists deprecated configuration file parameters and their associated sections or definitions.

Table A-1: Deprecated Parameters

Parameter	Configuration File Section or Definition	
Database	Alias section	
MirrorName	Alias section	
Password	Alias section	
PasswordType	Alias section	
SystemName	Alias section	
TableName	Alias section	
UserId	Alias section	
DirWait[1]	Cache section	
Mode[1]	Cache section	
Browse	Collection definition	
CollectorID	Collection definition	
KeepSubdirectories	Collector definition	
Permissions	Collector definition	Deprecated in RFS v5.2.
ProfileID	RFSPROFILE section	
AllowUnixEquivPerms	RFS Section	
DefaultDomain	RFS Section	Deprecated in RFS v5. Valid in RFS v4.
LocalPath	RFS section	Deprecated in RFS v5. Valid in RFS v4.
Version	RFS section	
TimeoutOverride	Storage definition	
Checkpoint	System definition	
SystemID	System definition	
TCP_Receive	UNIX section	
TCP_Send	UNIX section	

[1] Do not change the value of these parameters.

The ALIAS section is deprecated. This section was previously used for alias checking. It contains the following parameters: Database, MirrorName, Password, PasswordType, SystemName, TableName, and UserId.

Parameters for Customer Support use only

Table A-2 lists parameters for Customer Support use only.

Table A-2: parameters for Customer Support Use Only

Parameter	Configuration File Section or Definition	Definition
DirWait[1]	Cache section	When a file becomes eligible for cleanup in memory, the number of seconds the parent directory must idle before file tracking information in the parent directory can be removed.
MAX_RFS_FILE_LIST[1]	Cache section	Number of objects that must exist in memory before StorHouse/RFS takes aggressive actions to free memory.
TGT_RFS_FILE_LIST[1]	Cache section	Number of StorHouse/RFS file instances kept in memory.
CleanupTimeout[1]	RFS section	Time interval that StorHouse/RFS waits before closing StorHouse connections and purging cache files that are no longer in use.
Debug[1]	RFS section	Sets the debug level for StorHouse/RFS.
DebugLogLongOpSecs	RFS section	Threshold controlling the post-Storehouses/RFS version 5.1.0 enhancement to debug logging (IOP duration).

LookupBrowseThresholdPct	RFS section	Percent of MaxRFSFiles at which RFS performs an internal browse to service a file lookup. SGI recommends setting this parameter to a low value such as zero (0) when RFS contains a high number of very large directories (those with tens of thousands of files and/or subdirectories). The range of values is 0-100. Note: The RFS configuration file does not contain this parameter unless Customer Support instructs the customer to manually add it.
MaxCommandThreads	RFS section	The maximum number of threads RFS will use to process NFS/CIFS I/O requests. This number may need to be increased if frequently there are a high number of files being read or written concurrently or there are often a lot of queued I/Os due to recalls or tape reads. In either case, MaxActiveIOsFile should probably be decreased as well.
NumRPCThreads	RFS section	Number of threads used in SGI's RPC (remote procedure call) support of inter-process communications.
Simulate[1]	RFS section	Option to run StorHouse/RFS in simulation mode for development or evaluation use only.
ReturnJukeBox[1]	RFS section	Option to run StorHouse/RFS in a mode better suited for systems where all prospective clients and StorHouse/RFS servers are running a minimum of Linux version 6.3.
PasswordType[2]	STATS section	Used internally by StorHouse/RFS
DBDriver[1]	System definition	Name of the ODBC driver to be used for StorHouse connections.

DBHost[1]	System definition	Name or IP address of the host to use for the ODBC connection. ■ If omitted, RFS uses the name specified for the DNSName parameter. ■ RFS ignores DBHost if the DBDriver parameter and value are omitted.
DBPort[1]	System definition	Port used by RFS to connect to the StorHouse database through ODBC.
PasswordType[2]	System definition	Used internally by StorHouse/RFS
STHPort[2]	System definition	Used internally by StorHouse/RFS
PasswordType[2]	Storage definition	Used internally by StorHouse/RFS
SchedPolicy[1]	UNIX section	Scheduling policy used by UNIX StorHouse/RFS systems to control how threads are processed. Valid values are: ■ FIFO (first in first out) ■ RR (round robin) ■ OTHER, which is the default timesharing scheduling policy.

[1]Change this parameter only if instructed by Customer Support to do so.

[2]Never change this parameter. If you do so, your system may not work properly.

Parameters for StorHouse/CCi use only

The RFSPROFILES and the RFS_PIT sections contain parameters created and used solely by StorHouse/CCi. These parameters are not user-editable, so never change their values.

Table A-3 lists the parameters in the RFSPROFILES section.

Table A-3: Parameters in the RFSPROFILES Section

Parameter	Definition
CreateTime	Provides the correct create date/time string
Name	Provides the profile name
ProfileID	Provides the profile ID (deprecated)
UpdatedTime	Provides the correct update date/time string
Version	Provides the correct version number

Table A-4 lists the parameters in the RFS_PIT section.

Table A-4: Parameters in the RFS_PIT Section

Parameter	Definition
OwnedBy	Name of the StorHouse/CCi system that created the PIT system.
PitOs	Operating system of the StorHouse/RFS PIT machine.

Risks of updating the configuration file with a text editor

As previously stated in this manual, SGI *strongly recommends* that you *always* use StorHouse/CCi rather than a text editor to update the StorHouse/RFS configuration file. Failing to follow this recommendation could lead to synchronization problems between the StorHouse/RFS profile and configuration file. Furthermore, if you update a password in the configuration file with a text editor, you must subsequently run a separate procedure to encrypt that password as described in the following section.

Encrypting passwords for StorHouse/RFS platforms

You must run the `gen_cfg` program to encrypt the passwords in the StorHouse/RFS configuration file.

The `gen_cfg` program has two parameters: the input `rfs.cfg` (required) and the output `rfs.cfg` (optional). The default output parameter is

```
/rfs/files/rfs.cfg
```

To encrypt passwords for a StorHouse/RFS system

1. Log in to the StorHouse/RFS server as root.
2. Run the gen_cfg program.

   ```
   ./gen_cfg rfs.cfg
   ```

   or

   ```
   ./gen_cfg rfs.cfg /rfs/files/rfs.cfg
   ```

3. Respond to the prompts by entering and then re-entering the passwords. For example:

   ```
   enter the password for user SYSADM in collection definition
   DS:

   reenter the password:
   ```

Rereading the StorHouse/RFS configuration file

Whenever you update dynamic parameters in the StorHouse/RFS configuration file, you must request StorHouse/RFS to reread the file. The dynamic parameter changes take effect immediately after StorHouse/RFS performs the reread. The user ID you use requires read and write permission to the rfs.cfg file. StorHouse/RFS checks these permissions before rereading the file. StorHouse/RFS displays a Web page indicating successful processing. Otherwise, StorHouse/RFS returns a file not found condition, and the operating system displays the appropriate error message.

To reread the StorHouse/RFS configuration file

1. Log in to the StorHouse/RFS server platform as root.
2. Type the following (uppercase required):

```
ls /RFS/SM_LOCAL/RFSCONFIG.HTML
```

Restarting the StorHouse/RFS service

Whenever you update static parameters in the StorHouse/RFS configuration file, you must stop and then start the StorHouse/RFS service to implement the changes. Stopping StorHouse/RFS terminates any in-process local collections, StorHouse transfers, and StorHouse writes at the first convenient point. Starting StorHouse/RFS resumes accumulating files to the current local collection and starts StorHouse transfers and writes from the interruption point.

Note: To minimize the disruption caused by requiring StorHouse/RFS users to restart transmissions of large files, first check with users to determine an appropriate restart time.

To restart the StorHouse/RFS service

1. Log in to the StorHouse/RFS server platform as root.
2. Stop the StorHouse/RFS server by typing and entering:

   ```
   /opt/FLTKrfs/bin/lwsm stop lwreg
   ```

3. Start the StorHouse/RFS server by typing and entering:

   ```
   /opt/FLTKrfs/bin/lwsm autostart
   ```

APPENDIX B

Understanding the text version of the StorHouse/RFS configuration file

> ***NOTE:*** *If your site has a version of StorHouse/CCi prior to version 3.1 and plans to use features such as StorHouse/FTP or StorHouse point-in-time recovery, you may need to update the StorHouse/RFS configuration file with a text editor to add the FTPD and RFS_PIT sections and their associated parameters. Consult your SGI customer support representative before making these or any other updates to ensure you perform them correctly. Also review Appendix A before making any file updates.*

As previously stated in this manual, SGI recommends that you always use StorHouse/CCi to update your site's StorHouse/RFS profile and then deploy that profile to your StorHouse/RFS server. Nonetheless, there may be times when SGI Customer Support asks you to update the configuration file directly using a text editor.

For example, in the unlikely event that you are experiencing a problem with StorHouse/RFS operation, SGI may ask you to add certain debugging or other troubleshooting parameters to the StorHouse/RFS configuration file. These

parameters are normally reserved for Customer Support use only. To add these parameters, you must update the configuration file directly because the StorHouse/CCi interface does not display reserved parameters in the user interface.

It is helpful to familiarize yourself with the format of the configuration file so that you will be prepared to update it manually if required. This chapter describes the StorHouse/RFS configuration file format and provides a text example to help you achieve that goal.

About the StorHouse/RFS configuration file

Each StorHouse/RFS server has one StorHouse/RFS configuration file, rfs.cfg, which provides the operating parameters for StorHouse/RFS. The configuration file format consists of sections and definitions.

Sections

StorHouse/RFS determines the names of the sections in the StorHouse/RFS configuration file. These names are case sensitive (always uppercase) and must be delimited by square brackets. Table B-1 defines these sections.

Table B-1: Sections in the StorHouse/RFS configuration file

Section Name	Description
[RFS]	Specifies general StorHouse/RFS operating parameters and defaults.
[RFS_PIT]	Restricted use by StorHouse/CCi
[RFSPROFILE]	Restricted use by StorHouse/CCi
[COLLECTORS]	Specifies collector names and identifies corresponding collection definitions.
[CACHE]	Restricted use by SGI Customer Support
[AUDITLOG]	Specifies information required to create audit log records.
[EXCLUSIONS]	Identifies file masks to exclude from searches and collections.
[FTPD]	Specifies configuration parameters for StorHouse/FTP.
[STATS]	Specifies parameters used to generate and store statistics.
[UNIX]	Restricted use by SGI Customer Support

In StorHouse/RFS configuration files generated by StorHouse/CCi, the RFSPROFILE section appears first followed by the RFS section. Configuration files created by a text editor do not have an RFSPROFILES section. In those files, the RFS section appears first in the file. In either case, the RFS_PIT section must follow the RFS section. Other sections and definitions may appear in any order in the file.

Definitions

Definition names are user-specified and case insensitive (any combination of letters and numbers). Similar to section names, definition names must also be delimited by square brackets. SGI recommends a 12-character maximum length for definition names.

Table B-2 describes the types of definition names in the StorHouse/RFS configuration file.

Table B-2: Definitions in the StorHouse/RFS configuration file

Definition Name	Description
System	Defines where collections and statistics are stored on StorHouse.
Storage	Specifies where file locator data and collection metadata are stored on StorHouse and identifies the corresponding system definition.
Collection	Defines specifications for each collection set and identifies the corresponding storage definition.
Collector	Specifies directories where each collector looks for data and security requirements for files in those directories and identifies the corresponding collection definition.

Here's an example of how definition names work. As Table B-2 indicates, a system definition tells StorHouse/RFS where to store collections and statistics on a specific StorHouse system. If you have two StorHouse systems – one for production and one for test – you would require two system definitions in the StorHouse/RFS configuration file. You might use [SthProd] for the system definition name of your production StorHouse system and [SthTest] for the system definition name of your test system.

Parameters

Each section or definition in a StorHouse/RFS configuration file contains a series of parameters expressed as keyword-value pairs, in any order, according to the following rules:

- An equal sign separates each keyword and value.
- No spaces are allowed before or after the equal sign.
- Each keyword-value pair appears on a separate line in the file.

In the following example, FileCleanupTimeout is the keyword and 60 is the value:

FileCleanupTimeout=60

Required, optional, and commented out parameters

Parameters are required or optional. A required parameter must be specified with a value in the StorHouse/RFS configuration file. An optional parameter may be omitted from the file, or the keyword may be present without a value. In this case, the parameter assumes the default value. To comment out a keyword so that StorHouse/RFS ignores it, precede the keyword with a semicolon (for example, ;StatsInterval=60).

Dynamic and static parameters

Parameters are dynamic or static. When you change a dynamic parameter, the value takes effect immediately after you request a reread of the StorHouse/RFS configuration file or restart the StorHouse/RFS service. When you change a static parameter, the value takes effect only after you restart the StorHouse/RFS service. Refer to Appendix A, “Additional Information,” for instructions about how to reread the StorHouse/RFS configuration file and restart the StorHouse/RFS service.

Configuration file example

This section contains an example of a StorHouse/RFS configuration file.

```
[RFS]
LogFile=/rfs/logs/rfs.log
Version=4.0
LocalPath=/rfs/files/localpath
CacheDir=/rfs/cache
SafetyPath=/rfs/files/safety
MaxCacheSpace=1000
Flush=NO
FileCleanupTimeout=1440
CleanupTimeout=10

[STATS]
StatsInterval=10
SystemName=SYS1
Database=RFS
TableName=ARCHIVE_STATS
DBUserID=ADMINSID
PasswordType=1
DBPassword="rZ3KGf9"

[SYS1]
DNSName=yoursys
STHPort=1200
SMUserID=STORSID
PasswordType=1
SMPassword="Wb49Ij"
SMGroup=RFS
VSET=RFS
FSET=RFS
FSETSegments=1
MaxSMFiles=64
MaxSMWriters=4
RetryInterval=0
VTF=NEXT
```

```
[STOR1]
SystemName=SYS1
Database=RFS
TableName=ARCHIVE_1
DBUserID=DBASID
PasswordType=1
DBPassword="UcE3m^O"
MaxSearchConnections=4
SearchConnectionTimeout=10

[Collectors]
ROOT=COLL1

[ROOT]
StagingDir=/rfs/collectors/root
UserDir=/
MaxStagingSpace=100000
MaxCollectionSpaceMB=200000
WaitTime=60
Group=RFS

[COLL1]
Storage=STOR1
CollectionDir=/rfs/collections
MaxLoadInterval=1440
MaxWriteSize=1800
Retention=0
Compression=NO
```