



Release Notes for StorHouse/RM 3.4

StorHouse/RM Release 3.4

Publication Number
900126 Rev. O

July 2, 2008

The FileTek logo consists of the word "FileTek" in white, bold, sans-serif font, set against a solid teal square background.



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Copyright © 1996-2008 FileTek, Inc. As an Unpublished Licensed Work.
Publication Number: 900126 Rev. O

NOTICE: U.S. GOVERNMENT USERS

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 Unpublished licensed work property of FileTek, Inc. Unauthorized use, duplication or distribution prohibited.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

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

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

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

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

Contents

Welcome	vii
Intended audience	vii
Contents of document	vii
Related documentation	viii
 Chapter 1: Changes and enhancements	 1-1
System requirements	1-2
New soft drop feature	1-2
New SQL_DROP_HOLD system parameter	1-3
New SYSDROP_PEND system table	1-3
New CREATE TABLE clauses	1-4
Format	1-4
Examples	1-6
New PURGE TABLE statement	1-6
Format	1-7
Examples	1-7
Metadata backup enhancements	1-7
New option to reset journaling	1-8
Enhanced metadata backup process	1-8
Metadata restore enhancements	1-8
New metadata restore command format	1-8
Metadata restore file version support	1-9
Metadata backup file verification	1-9

Capability to restore to an alternate database	1-10
Enhanced metadata restore process	1-10
Automatic database shutdown during a restore	1-10
Disabled metadata restore feature in StorHouse/Admin	1-11
Redo journaling enhancements	1-11
New journal audit utility	1-11
Format	1-12
Examples	1-12
New checkpointing option for journal replay	1-13
StorHouse/ODBC installation enhancements	1-13
New segment delete options	1-14
New BIGINT data type	1-14
Data loader enhancements	1-15
New LOAD INDEX FOR TABLE Clause	1-16
Format	1-16
Examples	1-16
Support for date-time format masks	1-17
Generating time and date-time values	1-18
New real-time load status	1-18
Secure FTP (FTPS) support added	1-20
Support for additional FTP server commands	1-20
Compression	1-21
SQL enhancements	1-21
New RENAME statement	1-21
New COUNT_BIG aggregate function	1-22
New TO_VARCHAR scalar function	1-22
New RIGHT scalar function	1-23
New SUBSTR_UDB scalar function	1-23
Character to binary assignment allowed	1-23
TIMESTAMP support in SQL functions	1-24
LIKE predicate supports the BINARY data type	1-24
New SYSTIMESTAMP special register	1-25
Increased size of DEFAULT value string	1-25
Additional rules for delimited names	1-25

New SQL codes	1-25
Miscellaneous	1-26
New Table Audit Utility	1-26
Format	1-27
Examples	1-28
Example 1	1-28
Example 2	1-28
Example 3	1-29

Chapter 2: Special considerations 2-1

SQL code -301031	2-1
SQL code -30033	2-1
DESCRIBE BIND restrictions	2-2
Design advisory for join operations	2-2
ISQL product status	2-2
DDL processing in general	2-3
Host variables as BINARY, VARBINARY, and VARCHAR data types	2-3
Immediate restart after a load failure	2-3
LOB restrictions	2-4
ESQL	2-4
SQL	2-4
Use of SYS in table names	2-4
Use of control characters as delimiters	2-4



Contents

Welcome

The *Release Notes for StorHouse/RM 3.4* identifies changes, enhancements, and special considerations for StorHouse/RM release 3.4.

Intended audience

This document is intended for StorHouse/RM users who are familiar with the StorHouse/RM software and for new users who want a summary of changes.

Contents of document

This publication contains the following chapters:

- Chapter 1, “Changes and enhancements,” summarizes updates and new features in StorHouse/RM release 3.4.
- Chapter 2, “Special considerations,” describes issues that may, in certain environments or fields of use, require careful review during assessment of an application’s use of StorHouse/RM at this time.

Related documentation

Refer to the following documents for information about StorHouse/RM.

- The *StorHouse SQL Reference Manual*, publication number 900111, describes the SQL statements, predicates, and functions supported by StorHouse®.
- The *StorHouse/RM SQL and Utility Quick Reference*, publication number 900122, provides a summary of the material in the *StorHouse SQL Reference Manual* as well as utility formats described in the *StorHouse Database Administration Guide*.
- The *StorHouse Database Administration Guide*, publication number 900108, describes StorHouse database concepts and explains how to create user tables and indexes, manage accounts and privileges, set up user tablespaces, and perform other StorHouse database administration tasks.
- The *StorHouse ESQL Manual*, publication number 900121, explains how to use StorHouse SQL in application programs.
- The *FileTek MVS Data Loader Utility Manual*, publication number 900109, describes how to load data into StorHouse user tables from an MVS environment.
- The *FileTek FTP Data Loader Manual*, publication number 900115, explains how to load data into StorHouse user tables from UNIX®, VAX, or other hosts using your standard File Transfer Protocol (FTP) client software.
- The *FileTek FTP Data Unloader Manual*, publication number 900137, explains how to unload data from StorHouse databases using FTP. It describes the UNLOAD control statement you prepare to format result data, the SELECT statement you prepare to select the data to unload, and the subset of FTP commands you use to transfer control information and to receive result data.



- The *StorHouse/RM Metadata Conversion Manual*, publication number 900142, explains how to convert metadata (when necessary) from one StorHouse/RM release to another.
- The *StorHouse/RM Glossary*, publication number 900112, defines the terminology used in the StorHouse/RM User Document Set.



Welcome

Related documentation

Changes and enhancements

This chapter describes the changes and enhancements to StorHouse/RM for release 3.4. Significant new features are:

- Soft drop
- Metadata backup and restore enhancements
- Auditing utility for journaling
- Checkpoint options for the journal replay utility
- BIGINT data type
- Data loader enhancements
 - LOAD INDEX FOR TABLE clause
 - Date-time format masks
 - SYSTIME and SYSTIMESTAMP keywords
 - Ability to obtain a real-time load status
 - Secure FTP (FTPS)
 - Additional FTP server commands
 - MODE Z compression
- SQL enhancements
 - RENAME statement
 - COUNT_BIG aggregate function
 - TO_VARCHAR scalar function
 - RIGHT scalar function
 - SUBSTR_UDB scalar function
 - Character to binary assignment
 - TIMESTAMP support in SQL functions
 - Increased size of default value string
 - Additional rules for delimited names
 - New SQL codes

- Miscellaneous SQL enhancements
- Table audit utility

System requirements

StorHouse/RM release 3.4 requires the following:

- UNIX® operating system environment/platform:
 - Sun™ Solaris™ 8, 9, or 10 on Sparc systems
 - HP-UX™ 11.x on PA-RISC systems
 - IBM® AIX™ 5.2 or 5.3 on pSeries servers
- StorHouse/SM release 5.6

New soft drop feature

To support Electronic Data Replication/Recovery (EDR), FileTek developed a *soft drop* feature that provides a way to undo an errant table drop. In StorHouse/RM 3.4, dropping a user table no longer deletes metadata and invalidates segment files. The DROP TABLE statement now functions as a pending drop. A dropped user table may be undropped or later purged.

The following tools and resources are used to implement the soft drop feature:

- New system parameter – SQL_DROP_HOLD
- New system table – SYSDROP_PEND
- New CREATE TABLE statement clauses – FROM DROPPED, BEFORE, and INDEX NAMES
- New SQL statement – PURGE TABLE

- Utility extension – shseg_delete utility can now purge tables

New SQL_DROP_HOLD system parameter

Before a DBA or table owner may purge a dropped user table, a minimum number of days must elapse between the drop and purge operation. This value is specified by the new SQL_DROP_HOLD system parameter. An error occurs if an attempt to purge a dropped user table occurs before that time has expired. Specific information about the SQL_DROP_HOLD system parameter is as follows:

- Expanded Name: SQL length of time to defer actual table drop actions
- Type: Dynamic parameter (affects all engines started after the change)
- Range: 0 to 6500 days
- Default: 0 days, a purge may occur immediately after a delete operation
- User Access: SET, SHOW

New SYSDROP_PEND system table

StorHouse/RM 3.4 stores information about dropped user tables and associated indexes in the new SYSDROP_PEND system table. StorHouse/RM deletes rows from this table when a DBA or table owner successfully purges the user table.

Column name	Data type	Description
OLD_OWNER	VARCHAR(32) NOT NULL	Account ID of the owner of the dropped table or index.
OLD_NAME	VARCHAR(32) NOT NULL	Name of the dropped table or index.
NEW_OWNER	VARCHAR(32) NOT NULL	Same account ID as the OLD_OWNER.
NEW_NAME	VARCHAR(32) NOT NULL	Internal, unique name for the dropped table or index.

Column name	Data type	Description
TBL_OR_IDX	CHAR(1) NOT NULL	Type of component. Values are: T User table I Index
TBLID	INTEGER NOT NULL	ID of the user table.
DROP_TIME	TIMESTAMP NOT NULL	Date and time the user table was dropped.
USERNAME	VARCHAR(32) NOT NULL	Account ID of the user who dropped the table.

New CREATE TABLE clauses

An account with DBA privilege can now undrop a user table. The DBA may rename the user table and/or indexes or keep the original name(s). The CREATE TABLE statement consists of three new clauses for undropping a user table:

- FROM DROPPED (required) identifies the dropped user table
- BEFORE (optional) specifies a timestamp to select a specific table instance when multiple user tables with the same owner and table name exist (from creating and dropping, creating and dropping the same table)
- INDEX NAMES (optional) renames the indexes

Format

The CREATE TABLE format for undropping a user table is as follows.

```
CREATE TABLE [[owner.]table_name] FROM DROPPED [owner.]table_name
[BEFORE timestamp] [INDEX NAMES index [, index]...]
[TABLE SPACE tablespace_name]
```

Argument	Description
CREATE TABLE	
owner.	(optional) Account ID of the owner of the new table. If you omit the owner, StorHouse/RM uses the owner of the dropped table.
table_name	(optional) Name of the new user table. This name, in combination with the owner, must be unique from other tables, views, and synonyms in the current database. If you omit the table name, StorHouse/RM uses the name of the dropped table.
FROM DROPPED	
owner.	(optional) Account ID of the owner of the dropped table.
table_name	(required) Name of the dropped user table. This is the original table name before the table was dropped, not the temporary internal name assigned after the drop.
BEFORE timestamp	(optional) Date and time (timestamp literal) used to select a specific table instance to undrop when multiple user tables with the same owner and table name are in a database. This occurs when a user repeatedly creates a table, drops it, creates it, and drops it without purging the table. For example, if you dropped a user table on January 1, 2005, dropped it on the same day, created it again on January 2, 2005, and dropped it on the same day, then the SYSDROP_PEND system table would contain two table instances. If you wanted to undrop the first table instance, then you could specify BEFORE '1/2/2006'. If you omit the BEFORE clause, StorHouse/RM undrops the most recent table instance, in this example, the table dropped on January 2, 2006.

Argument	Description
INDEX NAMES index	(optional) Name(s) of the indexes for the new table. If you omit the index names and did not rename the table, StorHouse/RM uses the original index names. However, if you renamed the table and the table name was part of the index name, then StorHouse/RM also renames the table-name portion of the index name.
TABLE SPACE tablespace_name	(optional) Name of the user tablespace to contain the new user table. TABLE SPACE can be one or two words. If you omit this clause, StorHouse/RM uses the tablespace of the dropped table.

Examples

- The following CREATE TABLE example undrops a user table called LOCATION, keeping the original table and index names.

```
CREATE TABLE FROM DROPPED LOCATION
```

- The following CREATE TABLE example undrops the LOCATION table dropped before 01/30/2006 and renames the new table and indexes.

```
CREATE TABLE NEWLOCATION FROM DROPPED LOCATION
BEFORE '01/30/2006'
INDEX NAMES NEWLOCATION_IDX1, NEWLOCATION_IDX2
```

New PURGE TABLE statement

The new SQL statement PURGE TABLE deletes all metadata and associated indexes for a dropped user table and invalidates the segment files. Only a DBA or table owner may use this statement. StorHouse/RM checks the value of the SQL_DROP_HOLD system parameter to determine whether the purge operation may proceed.

After the purge, an authorized user then may run the segment delete utility to remove the segment files from StorHouse and the StorHouse REMOVE command to remove StorHouse directory entries to recover the space used by the segment files (if reusable).

Format

PURGE TABLE [owner.]table_name [BEFORE droptime]

Examples

- The following PURGE TABLE statement purges a user table called CUSTOMER owned by account ID USER 1. All instances of the user table are purged (no BEFORE clause).

PURGE TABLE USER1.CUSTOMER

- For the user table LOCATION, the following PURGE TABLE statement purges all instances that were dropped before 01/30/2006.

PURGE TABLE LOCATION BEFORE '01/30/2006'

Metadata backup enhancements

The metadata backup utility, sthdb_backup, backs up metadata for a StorHouse database. A new command option is available for resetting the journaling environment. And a temporary file is no longer used when writing metadata backup files.

New option to reset journaling

The new `-r` option enables you to reinitialize the journaling environment. This option creates a new journal info file and starts a new journal chain. Resetting the journaling environment may be useful in cases where a partial, or checkpointed, replay was performed. That is, a replay may have completed up to a certain checkpoint, but not all of the journal files may have been replayed.

For example, the following command reinitializes the journaling environment for the CALLS database.

```
sthdb_backup -r CALLS
```

Enhanced metadata backup process

Previously, the metadata backup utility wrote metadata to a temporary file and then to StorHouse. In order to accommodate metadata larger than the temporary file's size limit, the utility now writes metadata directly to StorHouse.

Metadata restore enhancements

The metadata restore utility, `sthdb_restore`, restores a StorHouse database with a metadata backup file. In this release, new command options and parameters are available.

New metadata restore command format

The new metadata restore command format is as follows:

```
sthdb_restore [options] database_name[:n] [-a alternate_database_name]...
```

The changes are as follows:

- A new `-r` option enables you to verify a metadata backup file without restoring the database.
- The `[:n]` parameter enables you to use an earlier version of a metadata backup file. The default is the most recent version.
- The `[-a alternate_database_name]` parameter enables you to restore the data to a different database.

The following sections describe the command enhancements.

Metadata restore file version support

Previously, the metadata restore utility used the most recent metadata backup file to restore a database. Now you can specify a version number, with the new `[:n]` argument after the database name, to restore an earlier backup of a database. The default is the most current version (0). The next most current version is -1.

For example, the following command restores the CALLS database with the version -1 metadata backup file.

```
sthdb_restore CALLS:-1
```

Metadata backup file verification

Before restoring a database, you can request a verification of the contents of a metadata backup file by using the new `-r` option. The utility performs a read verification only; it does not restore the database.

For example, the following command performs a read verification of the most recent metadata backup file.

```
sthdb_restore -r Calls
```

And the following example performs a read verification of the -1 file version.

```
sthdb_restore -r Calls:-1
```

Capability to restore to an alternate database

The new [-a alternate_database_name] argument specifies a name of a different database in which to restore the metadata. If the alternate database does not exist, the utility automatically creates a database directory. If the alternate database exists, the utility prompts (if running in interactive mode) to delete the existing content (for instance, .LOG, .IDX, .TBL, .CGF and .arc files.) in the database directory.

For example, the following command restores the metadata for the CALLS database to an alternate database called ALTCALLS.

```
sthdb_restore CALLS -a ALTCALLS
```

Enhanced metadata restore process

Previously, the metadata restore utility extracted the contents of a metadata backup file on StorHouse to a temporary file. In order to accommodate metadata larger than the temporary file's size limit, the utility now reads metadata directly from StorHouse.

Automatic database shutdown during a restore

Previously, a DBA had to bring down a database with the sthdb_down utility before running the metadata restore utility. In StorHouse/RM 3.4, the metadata

restore utility verifies whether a database is down and if not automatically brings it down.

Previously, the database directory had to exist prior to running RESTORE. In StorHouse/RM 3.4, restore will create the directory if it does not already exist.

Disabled metadata restore feature in StorHouse/Admin

Previously, a DBA could run the metadata restore utility with the StorHouse/Admin component of StorHouse/Control Center. Due to the complexity of restoring a database, that feature has been disabled. An authorized user can run the metadata restore utility only at the StorHouse operating system (UNIX) prompt with the operator account and password. Refer to the *StorHouse Database Administration Guide* for more information about running the metadata restore utility.

Redo journaling enhancements

StorHouse/RM release 3.4 includes a new journal audit utility and checkpointing options for the journal replay utility.

New journal audit utility

The journal audit utility, sthjou_audit, reads through a chain of journal files and performs CRC checks on each journal record. Optionally, the journal audit utility verifies each statement (for example, table INSERTs) in a journal file and prints a detailed accounting of the statements and the progress of the audit. By default, the utility checks journal files back to the most recent metadata backup. Use the [:n] parameter to specify a metadata backup file version number to audit journal files further back in time.

Format

sthjou_audit [options] database_name[:n]

Argument	Description
[options]	(optional) Command options.
-f -F	Option to verify each statement in a journal file and to print the number and type of statements encountered as well as the number of journal files processed. This option also provides more information about the progress of the audit as it proceeds through the journal chain.
-l n -L n	Option to limit the number of journal files audited. The n must be greater than or equal to 1. A negative value or 0 is invalid and generates an error. For example, -l 15 audits the first 15 journal files in a journal chain.
-h -H	Option to display online help for the utility.
database_name	(required) Name of a valid, accessible StorHouse database.
[:n]	(optional) File version number to audit journal files from the current one through a later metadata backup. Version 0 is the default and the most recent metadata backup file version. The next most recent version is -1. For example, Calls:-1 audits all journal files from the current one, through the most recent (0 version) backup, back to the -1 backup.

Examples

- The following example performs a standard journal audit through the most recent backup for the CALLS database.

```
$sthjou_audit CALLS
```

- The following example performs a detailed journal audit through the -l metadata backup file version for the CALLS database. The audit is limited to the first 25 journal files in the chain.

```
$sthjou_audit -f -l 25 CALLS:-1
```

New checkpointing option for journal replay

The journal replay utility, `sthjou_replay`, can restore up to the last committed transaction or to a *checkpoint*—a point (or journal file) in the journal chain where the replay stops. With replay checkpointing, you perform a partial replay, restoring only those transactions up to a checkpoint.

You can specify the checkpoint two ways:

- By using the new -l or -L option to list the journal files and then select the cutoff journal file from the list.
- By using the new -d or -D option to specify a *delta time* in days. The format is:

```
-d N[.nnnnn] or -D N[.nnnnn]
```

where N (required) is days and .nnnnn (optional) is a fraction of a day. For example, -d 1.5 replays all of the transactions except for the last day and a half.

StorHouse/ODBC installation enhancements

The StorHouse/ODBC drivers for UNIX platforms are now installed from packages rather than from tar files. Refer to the *StorHouse/ODBC Driver Reference Manual* for more information about installing the packages.

New segment delete options

The `sthseg_delete` utility, used to mass-remove invalidated segments and their associated objects from a StorHouse database, has two new options.

- `-p` or `-P` – Option to purge all user tables first. This option submits the `PURGE TABLE` SQL statement used to delete metadata and invalidate segments. Only invalidated segments may be deleted.
- `-b days` or `-B days` – Option to specify a minimum number of days the table must be dropped before its segments may be deleted. The default value is the value of `SQL_DROP_HOLD` system parameter.

Also, an additional verbose level (`-v` option) is now available. The new range is 0 (default, minimal messages) through 3 (most messages), for example, `-v 1` or `-v 3`.

For example, the following command purges all user tables in the `TEST` database, deletes segments meeting the `SQL_DROP_HOLD` requirement, and uses the default verbose level (0).

```
run sthseg_delete -p -d 0 TEST
```

The following command purges all user tables, instantly deletes segments, and requests detailed messages.

```
run sthseg_delete -p -b 0 -d 0 -v 3 TEST
```

New BIGINT data type

The new big integer (BIGINT) data type is a 64-bit integer useful for applications that handle numbers beyond 2^{31} . StorHouse/RM supports BIGINT as follows.

- You can define a user table column as BIGINT.

- You can create range indexes for BIGINT columns.
- StorHouse/RM automatically creates a new SYSRANGES_BIGINT system table when you create a database or when you create a range index (in an existing database) for a BIGINT column.
- A new aggregate function COUNT_BIG counts the number of rows in a table or the number of non-NULL values in a column or group of values, returning a BIGINT result.
- For existing SQL functions, you can use BIGINT wherever INTEGER is used (AVG, SUM, MAX, MIN, TO_CHAR, CONVERT, SUBSTR, SUBSTR_UDB, RIGHT, ABS, CHR, TO_VARCHAR, RPAD, INSTR, CLOB, and OVERLAY).
- In ESQL programs, you can define a BIGINT type in the SQLDA as TPE_DT_BIGINT. The type of the return area is int64_t.
- With a FileTek data loader and unloader, you can load and unload BIGINT data.
- In ODBC applications, BIGINT fields can be defined using the SQL_BIGINT type. The underlying data type is int64_t (__int64 for Windows environments). The defined value ODBCINT64 can be used as the underlying data type in portable code.

Data loader enhancements

This section describes the following enhancements to the FileTek data loaders:

- LOAD INDEX FOR TABLE clause
- Date-time format masks
- SYSTIME and SYSTIMESTAMP keywords
- Ability to obtain a real-time load status
- Secure FTP (FTPS)

- Additional FTP server commands
- MODE Z compression

New LOAD INDEX FOR TABLE Clause

The LOAD INDEX statement now supports a FOR TABLE clause to enable loading deferred indexes owned by other users.

Format

```
LOAD INDEX[ES] {index_name,... | [index_name,... ]  
FOR TABLE [owner.] tablename} [subspace_clause] [SEGMENTS  
segment_list]
```

where subspace_clause is:

```
SUBSPACE ROTATE |  
[ VALUE | HASH ] SUBSPACE number
```

Examples

To load all deferred indexes for table jde54:

```
LOAD INDEXES FOR TABLE jde54
```

To load the deferred index date for table employ owned by ssc:

```
LOAD INDEX date FOR TABLE ssc.employ
```

Support for date-time format masks

A FileTek data loader field specification now supports a format string mask for a character format date-time input field. This feature enables the loading of a TIME EXTERNAL, DATE EXTERNAL, and TIMESTAMP EXTERNAL data field that is not in one of the standard formats. The following table identifies the StorHouse standard formats of date and time input and result data.

Standard formats for loading and unloading date and time data

Loader data type	Input or result data format
DATE EXTERNAL	YYYY-M[M]-D[D] M[M]/D[D]/YYYY M[M]-D[D]-YYYY D[D].M[M].YYYY
TIME EXTERNAL	H[H]:M[M] H[H].M[M] H[H]:M[M] AM PM H[H]:M[M]:S[S] H[H]:M[M]:S[S].CCC H[H].M[M].S[S].CCC
TIMESTAMP EXTERNAL	Any date external and time external combination. The date and time components may be separated by a space or a dash. Example: YYYY-M[M]-D[D]-H[H].M[M].S[S].[CCC] '1995-10-25-09.45.234'

When specifying a format mask for input data, note the following.

- The available masks are listed in Chapter 2 of the *StorHouse SQL Reference Manual*.
- The format string masks must be quoted and must follow the DATE EXTERNAL, TIME EXTERNAL, or TIMESTAMP EXTERNAL data type in the field definition.

- MCS, MLS fields will accept a value that is empty (contains no digits).
- An “*” can be used as a wildcard to indicate that an optional non-alphanumeric character may be present (for example, if an MCS/MLS field is empty).
- Leading and trailing blanks are accepted in a date-time field value.
- The default field length is equal to the mask length. However, this default length may not be large enough for the data field. Therefore, when including a format mask string, specify a field length.

Below is an example format string mask in a field specification of an INTO TABLE clause.

```
TS_TEST    POSITION(35) TIMESTAMP EXTERNAL(16) 'YYYYMMDDHH24MISS'
```

Generating time and date-time values

In addition to the currently supported SYSDATE keyword, a FileTek data loader data specification now supports the SYSTIME and SYSTIMESTAMP keywords. These keywords generate date (SYSDATE), time (SYSTIME), and date-time (SYSTIMESTAMP) values to be loaded into a StorHouse table.

New real-time load status

Previously, you could obtain the status (successful or failed) of a completed FTP data load. With StorHouse/RM 3.4, you can also obtain the current state of a running data load. You use the status keyword on the FTP put command to obtain the load status.

The following example illustrates two status checks for an in-progress load. The first check (at the bottom of the listing) indicates the data loader is loading data into segment ID Calls:seg0001. The second check (at the top of the listing)

indicates the data loader is parsing the control statements for loading segment
Calls:seg0002.

```
ftp> put - status,dbn=Calls
200 PORT command successful.
500- %L-I-XLDINFO, \Load op: DB=Calls started Fri Aug 11 14:40:25 2006\
500- %L-I-XLDINFO, \Load <Calls:seg0002> started 11-AUG-2006:14:40:13\
500- %L-I-XLDINFO, \Load is still in progress. Chkp state = reading ctrl
stmts\
500- %L-I-XLDINFO, \0 stmts have so far completed (successfully)\
500- %L-I-XLDINFO, \Current stmt/op = unknown, phase = Parse control
stmts\
500- %L-I-XLDINFO, \*Waiting for input (ftp/VRAM) (8 sec)...\
500- %L-I-XLDINFO, \Total wait times (so far):\
500- %L-I-XLDINFO, \ SM I/O (data+file ops) = 0.0+0.0s\
500- %L-I-XLDINFO, \ Read input (ftp/VRAM) = 0.0s\
500- %L-I-XLDINFO, \-----\
500- %L-W-XLDEXISTS, An LD process already exists for this /LOADIDENT.
500 ***** Load operation/request finished ***** sqlcode=<-301811>
ftp>
ftp>
ftp> put - status,dbn=Calls
200 PORT command successful.
500- %L-I-XLDINFO, \Load op: DB=Calls started Fri Aug 11 14:27:47 2006\
500- %L-I-XLDINFO, \Load <Calls:seg0001> started 11-AUG-2006:14:27:12\
500- %L-I-XLDINFO, \Load is still in progress. Chkp state = loading data\
500- %L-I-XLDINFO, \0 stmts have so far completed (successfully)\
500- %L-I-XLDINFO, \Current stmt/op = LOAD DATA, phase = Xfer data\
500- %L-I-XLDINFO, \*Waiting for input (ftp/VRAM) (18 sec)...\
500- %L-I-XLDINFO, \Total wait times (so far):\
500- %L-I-XLDINFO, \ SM I/O (data+file ops) = 0.0+0.0s\
500- %L-I-XLDINFO, \ Read input (ftp/VRAM) = 11.4s\
500- %L-I-XLDINFO, \ SQL operations = 0.4s\
500- %L-I-XLDINFO, \-----\
500- %L-W-XLDEXISTS, An LD process already exists for this /LOADIDENT.
500 ***** Load operation/request finished ***** sqlcode=<-301811>
ftp>
```

Secure FTP (FTPS) support added

StorHouse/RM supports FTPS (i.e., explicit FTPS as per RFC 4217). FTPS can be used to encrypt all communication between the client and server machines, including the login. The server-level commands are AUTH SSL, AUTH TLS, PBSZ, and PROT.

FTPS is usually selected via a GUI menu. To use the data loader FTPS functionality, you must set up a server certificate. The certificate can be self-signed, using a toolset like OpenSSL, or it can be established by a certificate authority such as Verisign, depending on your requirements and your security environment.

Support for additional FTP server commands

The data loader now supports the following FTP server commands:

- SITE NEED150 – sets “150” message semantics (sends the intermediate “150...” reply for a transfer, even if StorHouse does not require the information (for example, for a confirm)). This may be needed for compatibility with some FTP clients (to prevent a hang condition).
- SITE IDLE – sets the idle timeout (instructs the system to timeout if the client is idle for the specified number of seconds)
- SITE PINGINT – sets the ping interval in seconds
- SITE PINGBUF – sets the size (in bytes) of the ping buffer
- FEAT – inquires what extensions are supported
- OPTS – sets related parameter values
- LIST – returns the remote filename and the size of the last load

- NLST – returns the remote filename of the last load
- SIZE – provides the size of the last transfer (for example, to verify the load)

Compression

StorHouse/RM supports MODE Z compression, which is usually selected using a GUI checkbox. FileTek recommends using MODE Z compression for all FTPS transfers to reduce the CPU load for encryption.

SQL enhancements

StorHouse/RM release 3.4 includes the following SQL enhancements.

New RENAME statement

The RENAME statement changes the name of a user table, view, or synonym. You must have DBA or RESOURCE privilege to use the RENAME statement. A DBA may change the table name for another owner. The old name becomes obsolete.

Note: You cannot change the owner of the table, synonym or view. The owner name is optional. If omitted, the default is the current owner.

The format is:

```
RENAME old_object_name TO new_object_name
```

For example:

```
RENAME my.table TO my.table2
```

New COUNT_BIG aggregate function

The COUNT_BIG function is similar to the COUNT function. Both functions count the number of rows in a table or the number of non-NULL values in a column or group of values. The difference between the functions is the result data type:

- INTEGER result for COUNT
- BIGINT result for COUNT_BIG

For example, the following SQL statement counts the number of rows in the CALL_DETAIL table:

```
SELECT COUNT_BIG(*)  
FROM CALL_DETAIL;
```

New TO_VARCHAR scalar function

The scalar function TO_VARCHAR converts a specified expression to variable-length character form and retrieves the results. The second argument supplies a maximum length for the result. The result type is always VARCHAR. If the expression is binary, no data manipulation occurs. The bytes are copied directly to the result and truncated if greater than the maximum length. If the first argument evaluates to NULL, the result is NULL.

For example, the following SELECT statement selects names from the CUSTOMER table and returns a 32-character VARCHAR result.

```
SELECT TO_VARCHAR (NAME, 32) FROM CUSTOMER
```


New RIGHT scalar function

The scalar function RIGHT returns a substring of the specified number of bytes from the tail (right end) of the specified string. If the input string is shorter than the specified number of bytes, the result is padded on the right.

For example, the following SELECT statement, using the scalar function RIGHT, returns the right-most 10 bytes from the ACCTNUM field in the CUSTOMER table.

```
SELECT ACCTNUM (RIGHT,10)
FROM CUSTOMER
```

New SUBSTR_UDB scalar function

The scalar function SUBSTR_UDB returns a substring of an argument based on a starting position in a specified expression and a length. The result data type is the same as the expression. If the argument evaluates to NULL, the result is blanks. For example, a SUBSTR_UDB of a zero-length VARCHAR string results in n blanks (where n is the length) versus NULL. The result is padded to n characters, if needed.

The following SELECT statement, using the scalar function SUBSTR_UDB, selects names from the CUSTOMER table and displays the corresponding account number portion of the customer ID as one 10-character substring.

```
SELECT NAME SUBSTR (CUSTID, 1, 10)
FROM CUSTOMER
```

Character to binary assignment allowed

StorHouse/RM 3.4 allows assign and comparison operations between character type fields (CHAR, VARCHAR) and binary type fields (BINARY, VARBINARY). These are bitwise operations; that is, no conversions of any type are performed.

StorHouse/RM 3.4 allows these comparison and assignment operations only to accommodate the DB2 practice of using CHAR with CCSID 65535 for binary data.

TIMESTAMP support in SQL functions

StorHouse SQL functions that currently support a parameter of only data type DATE or TIME now also accept TIMESTAMP. These functions are:

- ADD_MONTHS
- DAYOFMONTH
- DAYOFWEEK
- DAYOFYEAR
- LAST_DAY
- MONTH
- MONTHS_BETWEEN
- NEXT_DAY
- QUARTER
- WEEK
- YEAR

Additionally, the format parameter in the TO_CHAR function also supports TIMESTAMP.

LIKE predicate supports the BINARY data type

The LIKE predicate supports the BINARY data type. StorHouse/RM 3.4 allows matching of BINARY data types. Hex (x'dddd') should be used for constants.

New SYSTIMESTAMP special register

StorHouse/RM 3.4 supports the SYSTIMESTAMP special register. This special register contains the current data and time.

Increased size of DEFAULT value string

The DEFAULT clause in a CREATE TABLE statement enables you to specify a default value to be loaded into a column if no value is supplied. The maximum length of the string has been increased from 32 to 247 bytes.

Additional rules for delimited names

The following rules now apply to SQL delimited names.

- An empty string ("") is allowed for SQL delimited names.
- Trailing spaces in SQL delimited names are silently truncated.
- For the OBJECT_TYPE clause in CREATE/ALTER TABLE SPACE subspace specifications, an empty string is considered to be a space.
- Delimited names in 8-bit ASCII are not supported. (Note that this limitation does not apply to character literals.)

New SQL codes

The following SQL codes are new.

-11112 Commit disallowed - transaction rolled back

-20239 Invalid [port] specification in connect string

- 85027 Journal error, error writing to journal
- 85028 Journal error, error truncating journal file
- 85029 Journal error, error rolling back cycle
- 85101 Warning: Partial checkpointed replay completion success

Miscellaneous

Additional SQL enhancements are:

- New host type TPE_DT_CSTR – similar to TPE_DT_CHAR except that TPE_DT_CSTR always adds a null terminator. If there is no room for the null terminator, an error occurs.
- Row reclaim – system tables created in release 3.4 re-use space previously occupied by deleted rows.
- PRIMARY and FOREIGN KEY specifications are no longer accepted in SQL statements. Previously, they were accepted and ignored.

New Table Audit Utility

StorHouse/RM release 3.4 includes the sthtbl_audit utility, which validates the structure of a StorHouse/RM system table. The utility verifies every header structure in the physical file, and if the -s option is specified, every record in the file is read and checked for valid structure. If the table was created by release 3.4, the utility also verifies the deleted row chain. The standard output from the utility indicates whether the table has a 3.4 structure. If the table was created by release 3.4, the utility writes the message, “Table file supports record reuse,” to stdout.

Format

sthtbl_audit [-options] dbname tblid

Argument	Description
-options	(optional) Criteria to control how the utility processes information. The option characters are not case sensitive, for instance, you can specify -d or -D. The options do not have defaults. If you omit an option, the utility ignores the option functionality.
-d -D all deleted valid tid[:tid]	Dump selected records. If you specify -d, you must specify either all, deleted, valid, or tid. The tid argument specifies a tuple ID (row ID) or a range of row IDs.
-f -F	Repair any problems in the system table.
-n -N	Do not obtain a lock on the system table.
-p -P rows	Emit progress indications at the specified interval (and total).
-s -S	Perform a full table scan and associated checks and stats.
dbname	(required) Name of the StorHouse database. The database name is case sensitive. When running the sthtbl_audit utility with the StorHouse Command Language RUN or SCHEDULE command, enclose a database name with lowercase characters in double quotes.
tblid	(required) Table ID. Prefix the table ID of a temporary table with "x" (for example, x3). Temporary tables are scannable only once the record count has been written, which does not occur until the first record is fetched.

The utility sends normal output (for example, table errors) to stdout and wraps the output so as not to exceed 77 characters per line. Errors in setup/execution are sent to stderr.

The return status is like diff:

- 0 = Table contains no detectable errors
- 1 = Errors found in table (unless fixed)
- >1 = A serious error occurred

Examples

Here are some examples.

Example 1. This example runs the sthtbl_audit utility for table ID 2 in the TESTBUILD34 database.

```
sth.8> $STHROOT/bin/sthtbl_audit -p 10000 -s TESTBUILD34 2
sthtbl_audit: Obtaining READ lock on table...
sthtbl_audit: (DB=TESTBUILD34, tblid=2) started Wed Sep  5 18:25:48
2007
sthtbl_audit: Table file supports record reuse.
sthtbl_audit: 2582 total records processed Wed Sep  5 18:25:50 2007
sthtbl_audit: 0 errors found
sthtbl_audit: Table file statistics:
sthtbl_audit:   1% of table file records are deleted
sthtbl_audit:   Average row length = 62 (max = 397)
```

Example 2. This example does not use any options, so the utility only performs a header scan.

```
aixsth.31> $STHROOT/bin/sthtbl_audit jde_aixsth 3
sthtbl_audit: Obtaining READ lock on table...
sthtbl_audit: (DB=jde_aixsth, tblid=3) started Fri Sep 14 14:11:37 2007
sthtbl_audit: Table file supports record reuse.
```

Example 3. This example scans all the rows in the specified table and provides information for tuple IDs 3-10.

```
aixsth.103> $STHROOT/bin/sthtbl_audit -d 3:10 -s jde_aixsth 3
sthtbl_audit: Obtaining READ lock on table...
sthtbl_audit: (DB=jde_aixsth, tblid=3) started Fri Sep 14 14:41:13 2007
sthtbl_audit: Table file supports record reuse.
Row #3 = { 2 0 SYSCOLIDX SYSADM SYSCOLUMNS SYSADM OWNER U A N B 0 0 }
Row #4 = { 2 0 SYSCOLIDX SYSADM SYSCOLUMNS SYSADM TBL U A N B 1 0 }
Row #5 = { 2 0 SYSCOLIDX SYSADM SYSCOLUMNS SYSADM COL U A N B 2 0 }
Row #6 = { 3 0 SYSIDXIDX SYSADM SYSINDEXES SYSADM TBLOWNER U A N B 0 0 }
Row #7 = { 3 0 SYSIDXIDX SYSADM SYSINDEXES SYSADM TBL U A N B 1 0 }
Row #8 = { 3 0 SYSIDXIDX SYSADM SYSINDEXES SYSADM IDXNAME U A N B 2 0 }
Row #9 = { 3 0 SYSIDXIDX SYSADM SYSINDEXES SYSADM COLNAME U A N B 3 0 }
Row #10 = { 4 0 SYSTBLSPACESIDX SYSADM SYSTBLSPACES SYSADM ID U A N B 0
0 }
sthtbl_audit: 124 total records processed Fri Sep 14 14:41:13 2007
sthtbl_audit: 0 errors found
sthtbl_audit: Table file statistics:
sthtbl_audit: 4% of table file records are deleted
sthtbl_audit: Average row length = 89 (max = 200)
```

1

Changes and enhancements

New Table Audit Utility

Special considerations

This chapter highlights known issues that may, in certain environments or fields of use, require careful review during assessment of an application's use of StorHouse/RM at this time. If applicable to your environment, please discuss these issues with your FileTek systems engineer to explore possible design alternatives.

SQL code -301031

For some StorHouse/RM processing errors, SQL code -301031 (transaction aborted) may be mistakenly returned instead of the correct error code. Usually, this is caused by an out-of-space condition in a volume set or a miscellaneous hardware failure. If you receive this code, you may need to call FileTek Customer Support for help determining necessary corrective action.

SQL code -30033

SLQ code -30033 is a generic return code that applies to any unrecoverable error caused by a relational engine's exit. The code may appear in conjunction with several error situations. Whether these errors are program-induced or user-induced, the code always indicates that a serious error has occurred and requires assistance from FileTek Customer Support.

DESCRIBE BIND restrictions

DESCRIBE BIND VARIABLES does not correctly process an SQL statement that contains both scalar functions and host variable markers. For example, the following SQL statement, which contains the scalar function `T O_HEX` and host variable markers, does not work properly with DESCRIBE BIND VARIABLES:

```
SELECT * FROM table WHERE ( TO_HEX (bin_column) LIKE :var )
```

The following SQL statement works correctly with DESCRIBE BIND VARIABLES because it contains no scalar function:

```
SELECT * FROM table WHERE bin_column > :var
```

Refer to the *StorHouse SQL Reference Manual* for complete documentation about DESCRIBE BIND VARIABLES.

Design advisory for join operations

Queries that use extensive join operations may not be good candidates for StorHouse/RM execution, especially at the high data volumes that you generally expect in large database environments. When you require such queries, consult your FileTek systems engineer for performance analysis and modeling assistance.

ISQL product status

The ISQL tool allows interactive processing of SQL statements. Use this tool only as a general-purpose development tool. You should not incorporate it into production software or operations procedures. If you do use it, you must use the new version included in the release 3.4 client (host.sol2) tar file. This version allows longer connect strings (up to 100 characters).

DDL processing in general

In StorHouse/RM, DDL statement execution is atomic and permanent. The software performs an implicit COMMIT before and after every DDL statement. Transaction logic (user-specific bundling of statements) may lead to undesirable table-level locks that restrict the entire database from use. To protect against this, StorHouse/RM adds to user-specified transaction (BEGIN-END groups) logic an implicit COMMIT before and after every DDL statement.

Host variables as BINARY, VARBINARY, and VARCHAR data types

In an ESQL program, you cannot declare host variables as BINARY, VARBINARY, and VARCHAR data types in a Declare Section. To define these types of variables, set up an SQLDA and use the DESCRIBE statement as documented in the *StorHouse ESQL Manual*.

Immediate restart after a load failure

If a data load fails and you restart it immediately, the restart may fail if the load cleanup hasn't completed. You receive the following errors when this happens:

```
500- %L-I-XLDINFO, \sqlcode=<-60021> Loader: Unrecoverable segment file\
```

```
500- %L-I-XLDINFO, \Unrecoverable segment, table=RESL_TBL1\
```

You can avoid this situation by waiting a few seconds before restarting a load. If you receive these errors, try the restart again after the brief delay.

LOB restrictions

Restrictions on accessing LOBs via ESQL and using LOBs in SQL are as follows.

ESQL. You must use a single-row fetch with the BLOB_FILE and CLOB_FILE data types. You cannot use array or pointer-fetch with these data types. Additionally, StorHouse/RM does not support file reference variables as input, that is, to transfer a LOB value from a client file to StorHouse. StorHouse/RM supports output file reference variables to transfer a LOB value from StorHouse to a client file.

SQL. LOB data types are not allowed with the following SELECT statement clauses: DISTINCT, ORDER BY or GROUP BY. Additionally, LOB data types are not allowed with the following functions: MIN, MAX, and COUNT(DISTINCT lob_expr).

Use of SYS in table names

Table names may not start with SYS, for instance, SYS_STARTUP or SYSSERVICE. The SYS prefix is reserved for system tables.

Use of control characters as delimiters

For data loading, use of control characters as delimiters, especially whitespace characters, is discouraged in input data. A terminator declared explicitly that is also a whitespace character may result in a "terminator not found" error, particularly when the prior field is enclosed. If you are using one of these characters as a delimiter, specify WHITESPACE rather than the character.

The whitespace characters are SP (space), CR, FF, LF, VT, and HT (tab). In any ASCII code page these have hex values 20, 0D, 0C, 0A, 0B, and 09. In any EBCDIC code page the hex values are 40, 0D, 0C, 25, 0B, and 05.