User Log Format

StorHouse Release 5.6

Publication Number
007-6321-001
November 19, 2013

Publication Number: 007-6321-001

Contents

Welcome

This manual is a reference document for the StorHouse® user log. The *user log* is a StorHouse file that contains statistical and administrative information about StorHouse operations. StorHouse writes information to the user log when a specific event occurs or after a specified amount of time elapses. Logged information helps system administrators monitor StorHouse activity. In addition, reporting applications can read data extracted from the user log and generate reports based on that information.

Purpose of this Document

This document explains the structure and format of the user log and describes each of its components. It also describes how the user log works.

Intended Audience

This manual is intended for the StorHouse system administrator and for host system programmers who write user log data extraction applications. The StorHouse system administrator is responsible for configuring StorHouse for user log generation, gathering performance statistics, and tuning the StorHouse system to optimize operations. System programmers must know the types of statistics they need and the format in which the statistics are presented to write programs that extract the information and display it in a meaningful way.

This manual assumes that these persons already understand basic StorHouse concepts, functionality, and terminology.

Contents of Document

This document is organized as follows:

- Chapter 1, “Introduction,” describes the user log, concept of operation, log file format, and applicable StorHouse system parameters.
- Chapter 2, “User Log Content and Format,” describes user log characteristics and organization.
- Appendix A, “Data Record Types,” describes the data record types that may be logged for this release.
- Appendix B, “Sample Data Record,” contains an example of a user log data record.
- Appendix C, “User Log Message Protocol,” contains an example of a user log data record.

Related Documentation

You may want to refer to the material in the following documents:

- *Callable Interface Programmer's Guide*, publication number 900013 for IBM™ MVS™ hosts and the *Generic Callable Interface Programmer's Guide*, publication number 900046 for all other hosts, are references for programmers who write applications that invoke the StorHouse Callable Interface. These guides explain the functions of the Callable Interface and contain sample programs.
- *Command Language Reference Manual*, publication number 900005, is a general reference for Command Language, the standard command interface between StorHouse and all host computers. It contains descriptions of commands and related concepts.
- *StorHouse Glossary*, publication number 900027, defines technical terminology used in all SGI StorHouse publications.
- *Messages and Codes Manual*, publication number 900011, describes the messages and return codes generated by StorHouse host software and the StorHouse software. It lists the messages by status code, gives the meaning of each message, and indicates any actions to take as a result of the messages.

Notational Conventions

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

Convention	Meaning
Angle brackets (< >)	Enclose optional entries
Italics	New terms and emphasized text
lower case Helvetica font	User entries
UPPER CASE	System responses and StorHouse terms
▼	Procedures

Common Terms

Many of the charts in this document use common terms to describe record/field organization. These terms are described as follows:

Term	Specifies
FIELD	Field name
TYPE	Type of data in the field: character (char), numeric (num) or timestamp (time)
MAX. LEN.	Maximum number of characters allowed in the field, not including the null terminator
VALUE	Description of the field contents

If a field is null, it only contains the null terminator. If a field is 0 (zero), it contains an ASCII 0 (zero) followed by the null terminator.

Chapter 1

Introduction

This chapter introduces the user log and explains:

- What information is logged
- How the user log works
- User log file naming conventions
- User log system parameters.

About the User Log

The *user log* is a user-accessible StorHouse log file that contains statistical and administrative information to help system administrators manage StorHouse. The user log contains the following types of information:

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

Refer to Appendix A for a detailed description of each data record type.

How the User Log Works

Once a StorHouse system is installed, it automatically writes data records to the user log. StorHouse first writes these records to a temporary file on magnetic disk. The standard size of a temporary log file is 256 KB. Temporary files on magnetic disk scratch space are named ALMIxx, where xx is the log file number. This number may range from 01 through the value of LOG_MAX.

System administrators can control the number of temporary log files that may exist through the StorHouse system parameter LOG_MAX. For more information on system parameters, see "User Log System Parameters" on page 1-3.

When a temporary log file fills up or the system administrator executes the StorHouse NEWLOG /USER command, StorHouse closes the current temporary log file, opens a new one, and writes the closed file to StorHouse user storage. Reporting applications access the copy of the log file that resides in user storage. For more information on NEWLOG or other StorHouse commands, see the *Command Language Reference Manual*.

If StorHouse is extremely busy, all temporary log files can be in the process of being written to user storage. In this event, StorHouse puts the user log into a suspended state and discards all incoming log records. As soon as one temporary log file is completely copied to user storage, that file becomes available to contain logged records. StorHouse then generates a data record that contains a count of the records that were discarded during the suspended state. The name of this data record is missing_records. Refer to page A-32 for more information on this data record.

The appearance of the missing_records data record type indicates that StorHouse does not have enough temporary log files on magnetic disk scratch space for the amount of system activity. In this case, the system administrator should increase the value of the LOG_MAX system parameter.

User Log File Name Format

StorHouse names user log files in the following format:

Uyyyyjjjhhmmss

where:

- yyyy = year
- jjj = Julian date (from 1 to 365 or 366)
- hh = hour (from 00 to 23)

- mm = minutes
- ss = seconds.

The year, date, hour, minutes, and seconds correspond to the time when the user log file was written to StorHouse, after its last record was logged.

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

For example, a log file named U1998023145654 contains all log records between the previous user log file and January 23, 1998, at 2:56:54 p.m.

User Log System Parameters

System parameters are named data fields that StorHouse uses to manage resources and provide default information for functions. Each parameter has a name and a value. System parameters allow system administrators to configure and tune StorHouse for maximum performance.

System administrators choose values for many system parameters during system installation. Each parameter has an assigned default. Administrators can give a system parameter a different value from the default.

Twenty-one StorHouse system parameters govern the user log. These system parameters can be divided into the following categories:

- *Identification* – specify the account, file set, file access group, and volume set to which the user log is assigned
- *Management* – specify the interval length for data records and the maximum number of user logs on magnetic disk
- *Record control* – allow system administrators to customize the types of records that are written to the user log.

The following table defines the user log system parameters.

Table 1-1: User Log System Parameters

Parameter	Description
Identification System Parameters	
LOG_ACCOUNT	Specifies the account that writes user logs to StorHouse.
LOG_FSET	Specifies the file set that contains the StorHouse user logs.
LOG_GROUP	Specifies the file access group for StorHouse user log files.
LOG_VSET	Specifies the volume set that contains the StorHouse user logs.
Management System Parameters	
LOG_INTERVAL	Specifies the time between automatic writes of interval log records.
LOG_MAX	Specifies the number of user logs to maintain.
LOG_SERVER	Enables or disables logging of user log records for use by the real-time user log server.
Record Control System Parameters	
LOG_COMMAND	Enables or disables logging of command execution information records.
LOG_COPY	Enables or disables logging of file copy records.
LOG_DEVICE	Enables or disables logging of device information records.
LOG_FILE	Enables or disables logging of file open and close information records.
LOG_HEART	Enables or disables logging of system heartbeat information records.
LOG_OPERATOR	Enables or disables logging of operator console message records.
LOG_POLL	Enables or disables logging of polled (interval) records.
LOG_SECURITY	Enables or disables logging of security violation attempt information records for StorHouse/SM and StorHouse/RM.
LOG_SIGN	Enables or disables logging of signon and signoff information records for StorHouse/SM, and connect and disconnect records for StorHouse/RM.
LOG_SQL_STMT	Enables or disables logging of a user's SQL statement.
LOG_SQL_ TRANSACTIONS	Enables or disables logging of transaction statistics records upon completion of a transaction.
LOG_SYSTAT	Enables or disables logging of system startup and shutdown information records.
LOG_VOLUME	Enables or disables logging of volume mount/dismount/movement information records.
LOG_XFER	Enables or disables logging of extent transfer records.

For more information about these or any other system parameters, refer to the *System Administrator's Guide* or Appendix A of the *Command Language Reference Manual.*

Although this is a reference manual, system administrators must know how to display and set StorHouse user log system parameters to configure and tune the system. The following sections describe these tasks and provide examples.

Displaying System Parameters

System administrators can display the current value of a StorHouse system parameter by using the SHOW SYSTEM command. They must have SYSTEM or SHOW privilege to use this command.

▼ To display a system parameter value:

1. Sign on to StorHouse.
2. At the StorHouse command prompt (?), enter the following command and press R :

   SHOW SYSTEM name

   where name is the name of the system parameter. Below are some examples:

   – To find out whether the system is enabled to record security violation attempts into the user log, type:

     ? SHOW SYSTEM LOG_SECURITY

     The system returns the following information:

     LOG_SECURITY = TRUE

   – To display the current values for all user log system parameters (* is the StorHouse wild card symbol), type:

     ? SHOW SYSTEM LOG*

Setting System Parameters

System administrators can set or change the value of a StorHouse system parameter by using the SET SYSTEM command. They must have SYSTEM privilege to use this command.

▼ To set or change a system parameter value:

1. Sign on to StorHouse.
2. At the StorHouse command prompt (?), enter the following command and press R :

   SET SYSTEM name <value> modifiers

   where name is the name of the system parameter, value specifies a value for the named system parameter, and modifiers are other command modifiers you use to assign values other than the defaults. Below are some examples:

   – To change the value of the system parameter LOG_FILE from FALSE to TRUE, type:

     ? SET SYSTEM LOG_FILE TRUE

   – To change the number of user logs to maintain to 5, type:

     ? SET SYSTEM LOG_MAX 5

Chapter 2

User Log Content and Format

This chapter describes the common characteristics, organization, and format of user log records.

User Log Characteristics

System administrators and system programmers need to know the format of the user log, what kind of data records user log files contain, and the valid field string types. This section presents this information under the following topics:

- Transportable format
- Variable length records
- Field string types.

Transportable Format

Each user log is written to a StorHouse file with sequential organization and ASCII transportable format. This organization and format allow any host connected to StorHouse to read the logs, regardless of the host operating system.

Note For IBM hosts, user log files are converted from ASCII to EBCDIC by the StorHouse host interface software.

Variable Length Records

User log files contain variable length records. The maximum length for a user log record is 1024 bytes. If user log records are read using the StorHouse Callable Interface, the Callable Interface returns the length of each record when the Interface reads it.

If a user log file is transferred to a host using the StorHouse Interactive Interface (using the GET command), the host operating system must provide a way to determine the length of each variable length record.

Field String Types

User log records are subdivided into variable length fields. Each field is terminated by a null character (a binary zero for ASCII or EBCDIC). If a field is null, it has no contents, except for the null terminator. If a field contains 0 (zero), it has an ASCII (or EBCDIC) 0 (zero) followed by a null terminator.

There are three types of field strings: character, numeric, and timestamp. These field string types are described in the following table.

Table 2-1: Field String Types

Type	Abbreviation	Contains
Character	char	Alphabetic (A-Z), numeric (0-9), and any printable characters.
Numeric	num	Only the following characters: • 0-9 • + (plus sign) • - (minus sign) • . (decimal point)
Timestamp	time	Both alphabetic and numeric characters and are always in the format **dd-mmm-yyyy hh:mm:ss**, where: • dd = day of month • mmm = month abbreviation using first three letters • yyyy = year • hh = hour (24-hour clock) • mm = minutes • ss = seconds

User Log Organization

The user log is organized into two sections: the data dictionary and data records. The *data dictionary* is a group of records that contain information about the user log or about the content and format of data records contained in the user log. *Data records* contain the actual logged data.

The data dictionary is composed of three record types:

- *Dictionary headers* – describe the software version, release, and build in which the dictionary was defined, as well as the number of data record types logged.
- *Data record descriptors* – describe the specific fields that appear in each data record type. See Appendix A, “Data Record Types,” for all the data record types in this release.
- *Record header descriptors* – describe the set of seven fields that are common to the beginning of all data record types for this release. See Appendix A, “Data Record Types,” for all the record header descriptor fields in this release.

See “Data Dictionary Section” on page 2-6 for more information on the data dictionary and its components. Figure 2-1 illustrates the two sections of a user log file and the records they contain. User log file records appear in the order shown in this figure.

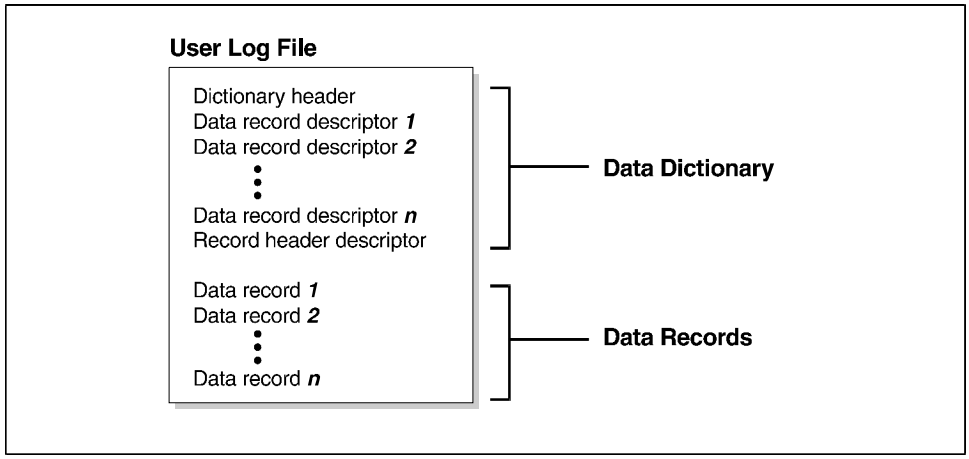

Figure 2-1: User Log Sections and Records

Conceptual User Log Drawings

Figure 2-2 and Figure 2-3 summarize the relationship between data record descriptors, record header descriptors, and data records in a user log file. You may find it helpful to refer back to these figures when reading the remainder of this document.

Figure 2-2 displays the contents of a data record descriptor and a record header descriptor. This figure emphasizes that the value of number_of_fields in the data record descriptor specifies the number of fields that follow in the data record.

Figure 2-3 illustrates how a data record descriptor and the record header descriptor define the contents of a data record.

The following two figures use the security_group data record descriptor. This record is always the sixth data record descriptor in the user log file.

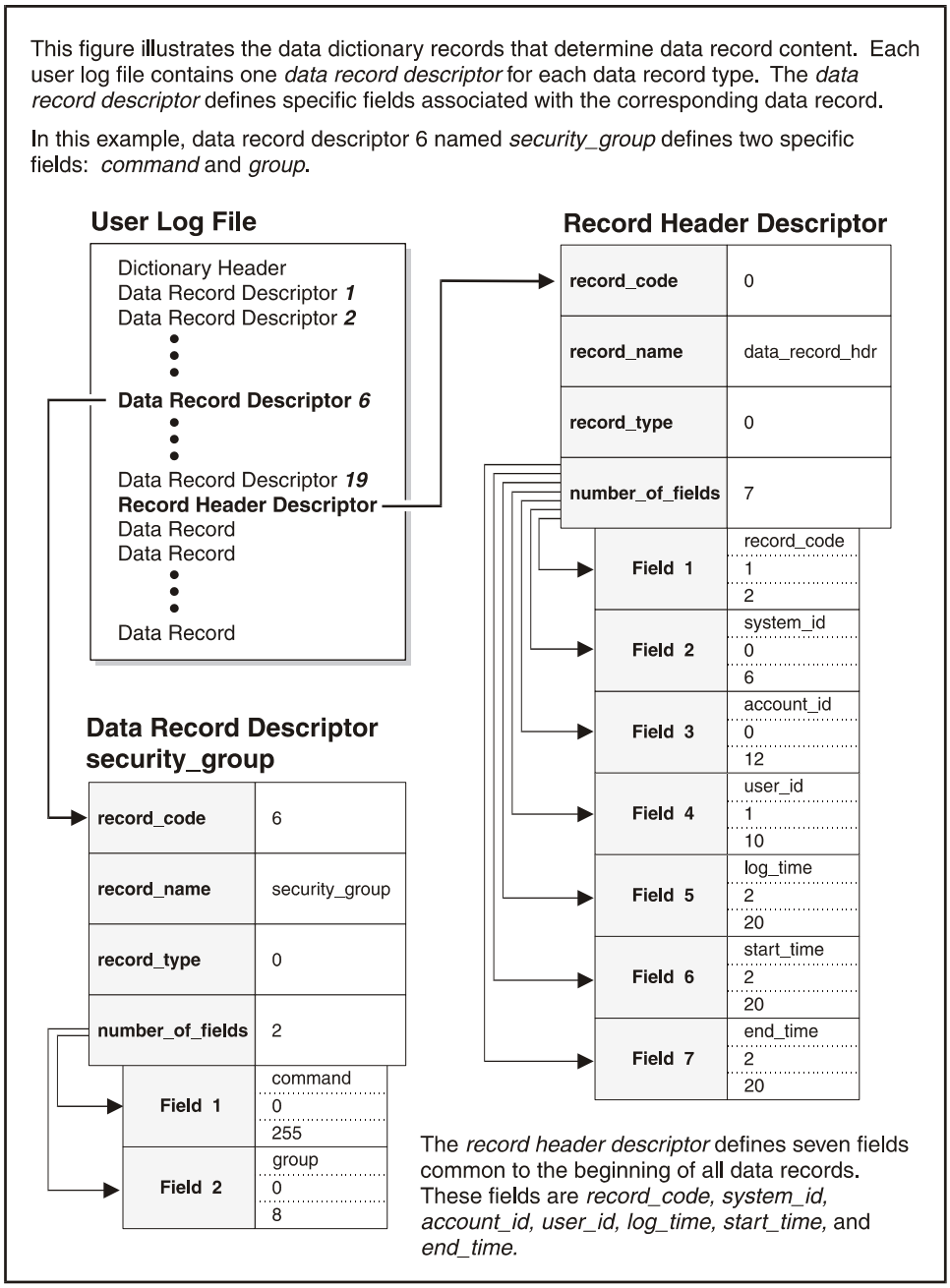

Figure 2-2: Content of Data Dictionary Records that Determine Data Record Fields

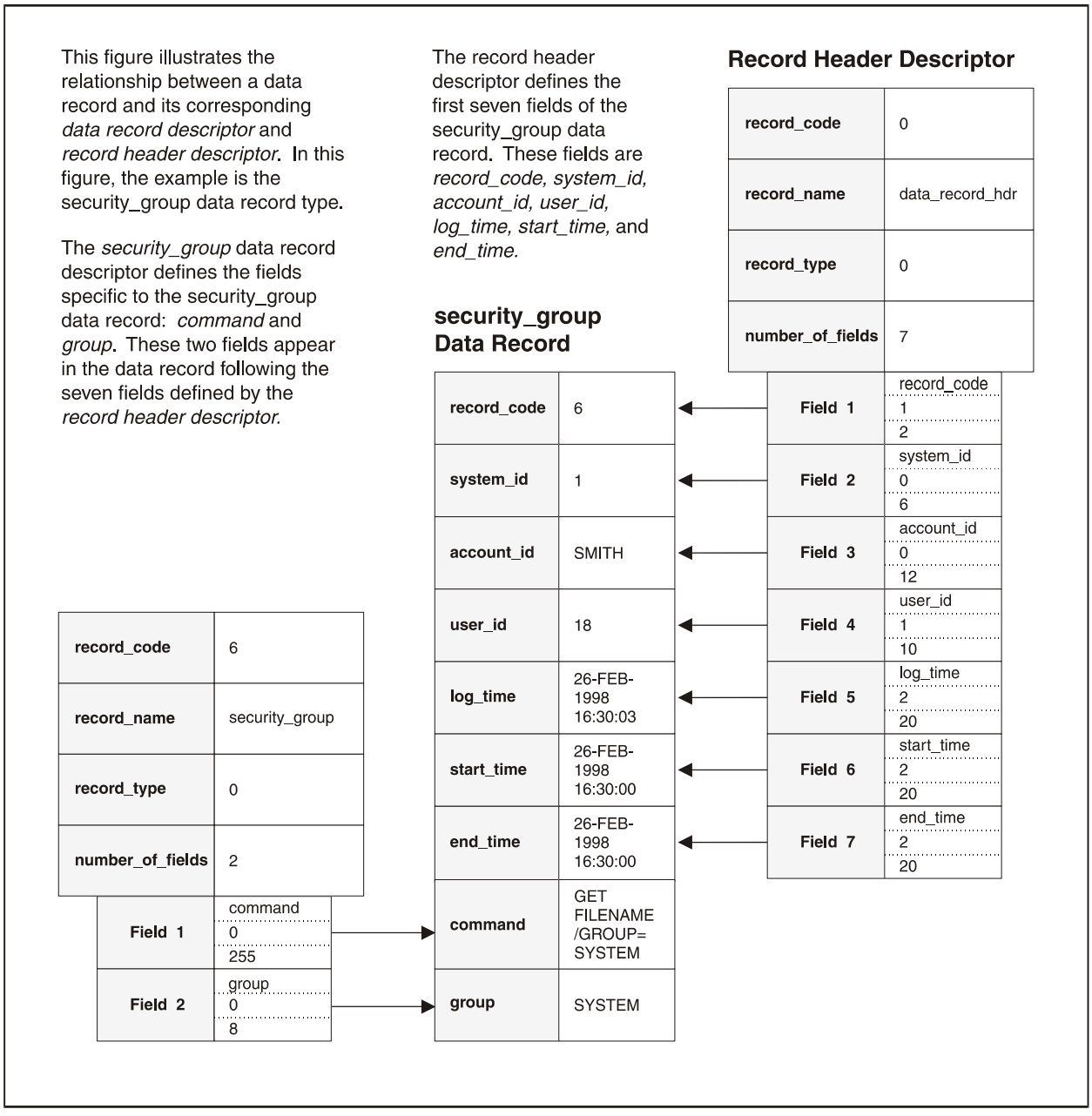

Figure 2-3: Relationship Between Data Record Fields and Data Dictionary Records

Data Dictionary Section

The *data dictionary* defines the format of records in the user log file. If two log files have different record formats, a single reporting (delogging) application can access both files by reading each file's data dictionary before processing the actual log records. The delogging program can extract desired information for input to a report generator, or translate each data record to a standard format, using data dictionary records to label each field.

The three types of data dictionary records are described in the following sections.

Dictionary Header

The *dictionary header* specifies the software version, release, and build in which the dictionary was defined, as well as the number of record types logged. It is always the first record in the data dictionary. Table 2-2 displays the content and format of the dictionary header. In Appendix A, "Data Record Types," fields in the dictionary header are shaded.

Table 2-2: Dictionary Header

Field	Type	Max. Len.	Value
release	char	10	StorHouse software version, release, and build in which log file and dictionary was defined. The format is **Vx.yz**, where: • x = version number • y = release number • z = build letter (not normally included).
numb_of_rec_types	num	3	Number of data record types that may appear in the user log. This is also the number of data record descriptors that follow this dictionary header record.

Data Record Descriptor

Data record descriptors describe the content and format of specific fields in each data record. They are located immediately following the dictionary header. There is one data record descriptor for each data record type in the user log. For example, if the numb_of_rec_types field in the dictionary header is 29, then there are 29 data record types and 29 data record descriptors.

Table 2-3 shows the content and format of a data record descriptor.

Table 2-3: Data Record Descriptor

Field	Type	Max. Len.	Value
record_code	num	2	Numeric code that identifies the data record type being defined. Values may range from 1 to 99, though not all values may be defined.
record_name	char	20	Name of data record. In a delogging report, this field could be used to print a title line.
record_type	num	1	Type of record: 0 = EVENT 1 = INTERVAL The system generates INTERVAL records at regular time intervals, set by the system parameter LOG_INTERVAL. The system generates EVENT records when a certain event occurs, such as when a user signs on to StorHouse, or when a file is opened.
number_of_fields	num	3	Number of data fields in the data record, not including the common fields contained in the record header descriptor. This field also specifies the number of field definitions in the record that follow this field.

The value of number_of_fields specifies the number of field format definitions that appear in this specific data record descriptor. For example, if number_of_fields is 3, then three field format definitions follow in the record. Each field format definition consists of the three components shown in Table 2-4.

Table 2-4: Field Format Definition Components

Field	Type	Max. Len.	Value
field_name	char	20	Printable name of field. This field may be used to print a title line in a delogging report.
field_type	num	1	Type of data contained in field. Values are: 0 = character 1 = numeric 2 = timestamp
field_max_len	num	3	Maximum number of characters in field. This number does *not* include the null terminator for the field. This number may be used by a delogging program to determine how many columns to allocate to a particular field in a report.

Each field format defines specific information that appears in the corresponding data records. In an actual data record, the fields appear in the same order as they are defined in the data record descriptor. The first data record descriptor field is located immediately after the data record's set of seven common fields, specified in the record header descriptor. Figure 2-4 illustrates this relationship.

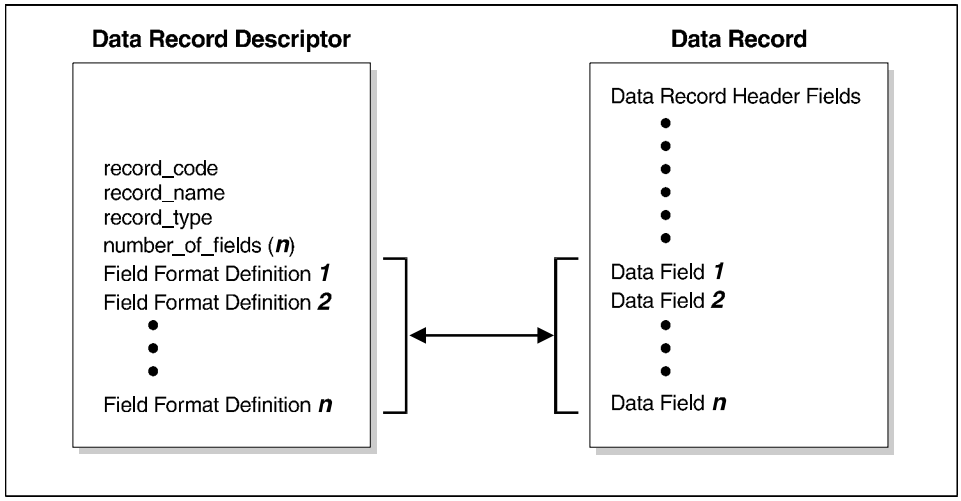

Figure 2-4: Relationship of Data Record Descriptor Fields to Data Record Fields

Record Header Descriptor

The *record header descriptor* defines the set of seven fields that are common to the beginning of all data records for this release. The record header descriptor is always the last record in the data dictionary, located just before the first data record. It appears once in the user log, has a record code of 0 (zero) and the name data_record_hdr.

Table 2-5 shows the format of the record header descriptor. Note that the format of the record header descriptor is similar to the data record descriptor. Both records include record_code, record_name, record_type, and number_of_fields. Additionally, the seven common fields described in the record header descriptor and the specific fields in a data record descriptor are defined in three parts: field_name, field_type, and field_max_len.

Table 2-5: Record Header Descriptor

Field	Type	Length	Value
record_code	num	1	0
record_name	char	15	data_record_hdr
record_type	num	1	0
number_of_fields	num	1	7
field_name field_type field_max_len	char num num	11 1 3	record_code 1 2
field_name field_type field_max_len	char num num	9 1 3	system_id 0 6
field_name field_type field_max_len	char num num	10 1 3	account_id 0 12
field_name field_type field_max_len	char num num	7 1 3	user_id 1 10
field_name field_type field_max_len	char num num	8 1 3	log_time 2 20
field_name field_type field_max_len	char num num	10 1 3	start_time 2 20
field_name field_type field_max_len	char num num	8 1 3	end_time 2 20

Data Records

Data records contain the actual data recorded in the user log. They are located immediately following the record header descriptor in the user log file. There is one data record for each logged entry. Data records consist of the seven common fields described in the record header descriptor, followed by the specific fields associated with the data record type. The specific fields for each data record are defined in the corresponding data record descriptor. This means that a data record's first data record descriptor field is always the eighth field in the data record. Table 2-6 shows the correlation between data record fields and data dictionary records.

Table 2-6: Correlation Between Data Record Fields and Data Dictionary Records

Data Record Fields	Data Dictionary Records
Common fields (7)	Record Header Descriptor
Specific Fields	Data Record Descriptor

Refer to Appendix B, "Sample Data Record," for an example of a user log record.

Appendix

A

Data Record Types

Appendix A defines the 30 data record types that may be logged by the current software release. Table A-1 lists all available record types in order by record code number.

Table A-1: Data Record Types

Code	Name	Description	Page
1	sys_startup	System startup	A-4
2	sys_shutdown	System shutdown	A-5
3	signon	User signon	A-6
4	signoff	User signoff	A-7
5	security_signon	Security violation attempt for signon	A-8
6	security_group	Security violation attempt for group	A-9
7	security_file	Security violation attempt for file	A-10
8	security_cmd	Security violation attempt for command	A-12
9	command_exec	Command execution	A-13
10	file_open	File open	A-15
11	file_close	File close	A-17
12	vol_dismount	Volume dismount	A-20
13	vol_movement	Volume movement	A-22
14	device_error	Device error	A-25
15	dev_state_chg	Device state change	A-27
16	library_info	Library information	A-28
17	drive_info	Drive information	A-30

Table A-1: Data Record Types

Code	Name	Description	Page
18	heartbeat	Heartbeat	A-31
19	missing_records	Missing records	A-32
20	vol_mount	Volume mount	A-33
21	op_message	Operator message	A-34
22	file_copy	File copy	A-35
23	extent_transfer	Extent transfer	A-36
24	statistics	Error and device usage statistics	A-37
25	connect	Application attempts to connect to a StorHouse database.	A-40
26	disconnect	Application disconnects from a Stor-House database.	A-41
27	security_connect	StorHouse/RM connect request is denied.	A-42
28	sql_statement	StorHouse/RM receives a user's SQL statement.	A-43
29	sql_transactions	SQL transaction is completed.	A-45
30	sql_file_close	StorHouse/RM file is permanently closed or a transaction is ended.	A-46

Data Record Type Descriptions

Each data record type description includes:

- The record name
- A description of the record
- The record code number
- The record type (EVENT or INTERVAL)
- The number of data fields specific to the data record type
- The record control parameter that designates whether the record is logged
- The management parameter that designates whether the record is logged, if applicable

- A description of each data record header field
- A description of each data field specific to the data record type.

Some data records contain StorHouse status codes and messages. The *Messages and Codes Manual* provides information about most of these codes. Some of the codes that may appear, especially device error status codes, are considered internal system codes and are not listed in the *Messages and Codes Manual*. Consult your customer service representative for information about these codes.

Note In the figures that follow for each data record type, fields that are defined by the data record header are shaded. Fields that are defined by the corresponding data record descriptor are unshaded.

sys_startup

Logged upon completion of system startup.

- Record Code: 1
- Record Type: 0 (EVENT)
- Number of Fields: 1
- Record Control Parameter: LOG_SYSTAT

Table A-2: sys_startup

Field	Type	Max. Len.	Value
record_code	num	2	1
system_id	char	6	System identifier
account_id	char	12	Null. No account is associated with startup.
user_id	num	10	0 (zero). No user is associated with startup.
log_time	time	20	Time record was logged.
start_time	time	20	Time startup began.
end_time	time	20	Time startup completed.
startup_type	num	2	0 or 1: 0 = System performed a normal startup. 1 = System performed crash recovery during startup.

sys_shutdown

Logged when the system shuts down. With some abnormal shutdowns, the system may be unable to write a sys_shutdown record to the user log.

- Record Code: 2
- Record Type: 0 (EVENT)
- Number of Fields: 1
- Record Control Parameter: LOG_SYSTAT

Table A-3: sys_shutdown

Field	Type	Max. Len.	Value
record_code	num	2	2
system_id	char	6	System identifier
account_id	char	12	Account id of user who initiated shutdown, or null for some abnormal shutdowns.
user_id	num	10	User id of user who initiated shutdown, or 0 (zero) for some abnormal shutdowns.
log_time	time	20	Time record was logged.
start_time	time	20	Time shutdown began.
end_time	time	20	Time record was sent to log.
shutdown_type	num	2	0 or 1: 0 = System shut down normally. 1 = System shut down abnormally or the operator requested an abnormal shutdown (for example, SHUTDOWN /NOW with user activity pending).

signon

Logged when a user attempts to sign on, regardless of whether the attempt is successful.

- Record Code: 3
- Record Type: 0 (EVENT)
- Number of Fields: 2
- Record Control Parameter: LOG_SIGN

Table A-4: signon

Field	Type	Max. Len.	Value
record_code	num	2	3
system_id	char	6	System identifier
account_id	char	12	Account id supplied in signon request.
user_id	num	10	User id assigned to user upon successful signon; otherwise, field is 0 (zero).
log_time	time	20	Time record was logged.
start_time	time	20	Time signon request was received.
end_time	time	20	Time signon processing was completed. (This value is always the same as start_time.)
status	num	5	StorHouse status code indicating response to signon request.
status_message	char	132	Text message corresponding to the status code above.

signoff

Logged when a user session terminates, whether it was due to a normal SIGNOFF or an abort.

- Record Code: 4
- Record Type: 0 (EVENT)
- Number of Fields: 3
- Record Control Parameter: LOG_SIGN

Table A-5: signoff

Field	Type	Max. Len.	Value
record_code	num	2	4
system_id	char	6	System identifier
account_id	char	12	Account id of terminated user session.
user_id	num	10	User id of terminated session.
log_time	time	20	Time record was logged.
start_time	time	20	Time signon request was received.
end_time	time	20	Time system completed signoff processing.
num_commands	num	10	Count of StorHouse Command Language commands entered by the user during session. This includes all valid, invalid, and null commands, as well as the SIGNOFF command, if entered.
num_sec_attempts	num	10	Number of commands that failed due to security violation attempts during this session. Each attempt will have generated an appropriate security violation attempt log record if logging of security violation attempts was enabled.
max_severity	num	5	StorHouse status code containing highest severity error code generated during user session.

security_signon

Logged when a signon request is denied due to a nonexistent account id or incorrect password. The violation may be due to an honest error on the part of a user or an attempt to access an account without authorization. Multiple violations (multiple records) with a variety of account and/or password combinations are more indicative of a true attempt to violate system security.

- Record Code: 5
- Record Type: 0 (EVENT)
- Number of Fields: 0
- Record Control Parameter: LOG_SECURITY

Table A-6: security_signon

Field	Type	Max. Len.	Value
record_code	num	2	5
system_id	char	6	System identifier
account_id	char	12	Account id given in signon request.
user_id	num	10	0 (zero)
log_time	time	20	Time record was logged.
start_time	time	20	Time invalid signon request was received.
end_time	time	20	Time invalid signon processing was completed. (This value is always the same as start_time.)
* * * **NO FIELDS** * * *			

security_group

Logged when a request to access a group is denied due to either a missing or incorrect group password, or an inappropriate privilege. The violation may be due to an honest error on the part of a user or an attempt to access a group without authorization. Multiple violations (multiple records) with a variety of group and/or password combinations are more indicative of a true attempt to violate system security.

- Record Code: 6
- Record Type: 0 (EVENT)
- Number of Fields: 2
- Record Control Parameter: LOG_SECURITY

Table A-7: security_group

Field	Type	Max. Len.	Value
record_code	num	2	6
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted access request.
user_id	num	10	User id of user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time request was received.
end_time	time	20	Time request processing was completed. (This value is always the same as start_time.)
command	char	255	Command that caused the violation, as entered by the user.
group	char	8	Name of the group (specified in the command) for which the security violation attempt was logged.

security_file

Logged when:

- StorHouse denies access to a file because of either a missing or incorrect file password, or an inappropriate privilege.
- There is an attempt to delete a retained file.

The violation may be due to an honest error on the part of a user or an attempt to access a file without authorization. Multiple violations (multiple records) with a variety of file and/or password combinations are more indicative of a true attempt to violate system security.

- Record Code: 7
- Record Type: 0 (EVENT)
- Number of Fields: 7
- Record Control Parameter: LOG_SECURITY

Table A-8: security_file

Field	Type	Max. Len.	Value
record_code	num	2	7
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted the access request.
user_id	num	10	User id of user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time request was received.
end_time	time	20	Time request processing was completed. (This value is always the same as start_time.)
command	char	255	Description of the activity that caused the violation. This may be a StorHouse command, Callable Interface function, or another action (for example, an internal StorHouse process).
file_sysid	num	5	System identifier specified in command on which the violation attempt was made. If no file identifier was specified or the file identifier had not been located before the violation attempt was recognized, this field contains a 0 (zero).

Table A-8: security_file (continued)

Field	Type	Max. Len.	Value
file_fno	num	10	File number specified in the command on which the violation attempt was made. If no file identifier was specified, or the file identifier had not been located before the violation attempt was recognized, this field contains a 0 (zero).
filename	char	56	Name of file for which required password was not specified.
group	char	8	Name of group in which the specified file is located.
status	num	5	StorHouse status code that identifies error conditions.
status_message	char	132	Text message corresponding to the status code in the status field.

security_cmd

Logged when a request to execute a command is denied due to lack of required privileges or there is a retention violation (for example, an attempt to delete a retained file). The violation may be due to an honest error on the part of a user, an attempt to perform unauthorized operations, or an indication that a user needs an additional privilege to perform required tasks.

- Record Code: 8
- Record Type: 0 (EVENT)
- Number of Fields: 3
- Record Control Parameter: LOG_SECURITY

Table A-9: security_cmd

Field	Type	Max. Len.	Value
record_code	num	2	8
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted command.
user_id	num	10	User id of user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time command was received.
end_time	time	20	Time command processing was completed. (This value is always the same as start_time.)
command	char	255	Command that caused the violation, as entered by the user.
status	num	5	StorHouse status code that identifies error condition.
status_message	char	132	Text message corresponding to status code above.

command_exec

Logged at the completion of each StorHouse command that parsed successfully and started execution, even if the command did not complete successfully.

- Record Code: 9
- Record Type: 0 (EVENT)
- Number of Fields: 5
- Record Control Parameter: LOG_COMMAND

Table A-10: command_exec

Field	Type	Max. Len.	Value
record_code	num	2	9
system_id	char	6	System identifier
account_id	char	12	Account id of user who sent command.
user_id	num	10	User id of user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time command was received.
end_time	time	20	Time command was completed and the response to the user was about to be returned.
command	char	255	The entire command, as entered by the user.
status	num	5	StorHouse status code returned as the overall response to the command.
status_message	char	132	Text message corresponding to status code above.
severity	num	5	Status code with highest severity encountered during execution of command. For example, if the command performed a wild card operation, and one item specified by the wild card was inaccessible and failed to be processed, the overall command status may be a successful completion, but the severity would contain the status code for the failed item.
command_id	num	4	Identifier of a StorHouse Command Language command. Table A-11 lists the identifiers for all commands.

Table A-11: StorHouse Command IDs

Command	ID	Command	ID	Command	ID
ARCHIVE	1401	MIGRATE /BY_VSET	903	SET SYSTEM	1001
BACKUP	901	MONITOR	130	SET USER	707
CATALOG DEVICE	1711	MOVE VOLUME	812	SET VOLUME	801
CATALOG VSET	1703	MOVE VSET	813	SET VSET	809
CHECKPOINT	1901	NEWLOG	703	SHOW ACCOUNT	709
CONSOLE	1801	PURGE	409	SHOW DEVICE	802
CREATE ACCOUNT	701	PUT	301	SHOW FILE	404
CREATE BACKUP	1403	RECOVER DEVICE	1707	SHOW FSET	810
CREATE FILE	1601	RECOVER VOLUME	1407	SHOW GROUP	710
CREATE FSET	806	RELOCATE	1405	SHOW LOCKS	407
CREATE GROUP	702	REMOVE ACCOUNT	704	SHOW PARTITION	814
CREATE PRIMARY	1404	REMOVE FILE	410	SHOW SCHEDULE	1202
CREATE VSET	807	REMOVE GROUP	705	SHOW SYSTEM	1002
DELETE	401	REMOVE SCHEDULE	1203	SHOW TIME	3
DOWN DEVICE	804	REMOVE VOLUME	1704	SHOW USER	711
ENABLE	408	REPLICATE	1411	SHOW VOLUME	803
ERASE VOLUME	1708	See *	1412	SHOW VSET	811
ERASE VSET	1709	RESERVE SYSTEM	6	SHUTDOWN	501
EXECUTE STH_LOAD	2101	RESTORE DIRECTORY	1903	SIGNOFF	2
EXPORT	1701	RETIRE VOLUME	1406	STAGE	1410
EXTRACT DIRECTORY	1902	RUN	8	UNCATALOG VOLUME	1705
GET	201	SCHEDULE	1201	UNCATALOG VSET	1706
HELP	601	SERVICE	1	UNDELETE	406
IMPORT	1702	SET ACCOUNT	706	UNLOCK	405
INITIALIZE DEVICE	1710	SET DEVICE	815	UP DEVICE	805
LOCK	402	SET FILE	403	VALIDATE VOLUME	1408
MESSAGE	5	SET FSET	808		
MIGRATE	902	SET GROUP	708		

*Code 1442 indicates a REPLICATION_SERVER event, which specifies that a file copy action occurred on a secondary StorHouse system due to a REPLICATE command on a primary StorHouse system.

file_open

Logged when a VRAM file is opened.

- Record Code: 10
- Record Type: 0 (EVENT)
- Number of Fields: 15
- Record Control Parameter: LOG_FILE

Table A-12: file_open

Field	Type	Max. Len.	Value
record_code	num	2	10
system_id	char	6	System identifier
account_id	char	12	Account id of user who opened the file.
user_id	num	10	User id of session associated with the open request.
log_time	time	20	Time record was logged.
start_time	time	20	Time open request was received.
end_time	time	20	Time open request was completed.
status	num	5	StorHouse status code returned in response to open request.
status_message	char	132	Text message corresponding to status code above.
net_link	char	9	For VRAM files, the network link name used to establish a data transfer link with the host software.
mode	num	1	Mode specified by user in open request: 1 = READ 2 = APPEND 3 = UPDATE
method	num	2	Access method specified by user in open request: 1 = SEQUENTIAL 2 = RECORD 3 = RECORD and SEQUENTIAL 4 = KEYED 5 = KEYED and SEQUENTIAL 6 = RECORD and KEYED 7 = SEQUENTIAL, RECORD, and KEYED

Table A-12: file_open (continued)

Field	Type	Max. Len.	Value
file_sysid	num	5	System identifier of the specified file.
file_fno	num	10	File number of the specified file.
filename	char	56	Name of the specified file.
group	char	8	Name of group in which the specified file is located.
version	num	11	Relative version number of the specified file.
revision	num	5	Absolute revision number of the specified file version.
max_revision	num	5	Highest existing revision number of the specified file version.
organization	num	5	Organization of specified file: 1 = KEYED 2 = KEYSEQUENTIAL 3 = RECORD 4 = SEQUENTIAL 5 = VRAM (RECORD, KEYED, or KEYSEQUENTIAL file created or last modified before Release 3.0)
volume_set	char	8	Name of volume set in which the specified file is located.
file_set	char	8	Name of file set in which the specified file is located.

file_close

Logged when a file is closed or at the completion of a sequential file GET or PUT command.

- Record Code: 11
- Record Type: 0 (EVENT)
- Number of Fields: 28
- Record Control Parameter: LOG_FILE

Table A-13: file_close

Field	Type	Max. Len.	Value
record_code	num	2	11
system_id	char	6	System identifier
account_id	char	12	Account id of user who accessed the file.
user_id	num	10	User id of user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time open request or PUT/GET command was received.
end_time	time	20	Time close was returned to host from StorHouse.
linked	time	20	Time data link was established during open processing.
close_received	time	20	Time close request was received by StorHouse.
status	num	5	StorHouse status code returned in response to close request or highest severity status code encountered for a PUT or GET command.
status_message	char	132	Text message corresponding to status code above.
net_link	char	9	For VRAM files, the network link name used to establish a data transfer link with the host software. For non-VRAM files, field is null.

Table A-13: file_close (continued)

Field	Type	Max. Len.	Value
mode	num	1	Mode specified by user in open request: 1 = READ for VRAM file 2 = APPEND for VRAM file 3 = UPDATE for VRAM file 4 = PUT for sequential file 5 = GET for sequential file
method	num	2	Access method specified by user in open request: 1 = SEQUENTIAL 2 = RECORD 3 = RECORD and SEQUENTIAL 4 = KEYED 5 = KEYED and SEQUENTIAL 6 = RECORD and KEYED 7 = SEQUENTIAL, RECORD, and KEYED
file_sysid	num	5	System identifier of the specified file.
file_fno	num	10	File number of the specified file.
filename	char	56	Name of the specified file.
group	char	8	Name of group in which the specified file is located.
version	num	11	Relative version number of the specified file.
revision	num	5	Absolute revision number of the specified file version.
max_revision	num	5	Highest existing revision number of the specified file version.
organization	num	5	Organization of specified file: 1 = KEYED 2 = KEYSEQUENTIAL 3 = RECORD 4 = SEQUENTIAL 5 = VRAM (RECORD, KEYED, or KEYSEQUENTIAL file created or last modified before Release 3.0)
volume_set	char	8	Name of volume set in which the specified file is located.
file_set	char	8	Name of file set in which the specified file is located.

Table A-13: file_close (continued)

Field	Type	Max. Len.	Value
num_records	num	10	If file is VRAM file, this field specifies number of records appended, read, or written since file was opened; otherwise, field is null.
transferred	num	20	If MODE=UPDATE or READ, this specifies number of bytes transferred to host since the file was opened. If MODE=PUT or GET, this specifies the number of bytes transferred; otherwise, field is null.
size_ext	num	20	If MODE=APPEND, specifies size (in bytes) of all extents created for file since it was opened; otherwise, field is null.
new_rev	num	5	If MODE=APPEND, UPDATE, or PUT, field specifies newest revision number; otherwise, field is null.
deletes	num	10	If MODE=UPDATE, field contains number of records deleted since file was opened; otherwise, field is null.
updates	num	10	If MODE=UPDATE, field contains number of records updated since file was opened; otherwise, field is null.
key_changes	num	10	If MODE=UPDATE, field contains number of records updated that caused key changes since file was opened; otherwise, field is null.
bytes	num	20	If MODE=UPDATE, field contains total size (in bytes) of records updated since file was opened; otherwise, field is null.
cache_hits	num	10	For VRAM files, if MODE=UPDATE or MODE=READ and METHOD is not SEQUENTIAL, then this field specifies the number of record reads satisfied from VRAM cache.
first_record	num	10	Specifies the time (in milliseconds) from the end of the file open to the end of the first I/O (in other words, when the record is read or written).
open_time	num	10	Specifies the time (in milliseconds) from the open request to the end of the file open.

vol_dismount

Logged each time a volume is dismounted from a drive.

- Record Code: 12
- Record Type: 0 (EVENT)
- Number of Fields: 13
- Record Control Parameter: LOG_VOLUME

Table A-14: vol_dismount

Field	Type	Max. Len.	Value
record_code	num	2	12
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time volume was originally requested. Additional requests may be received for this volume after the time of this original request. In this case, the volume may remain mounted until all requests are satisfied and the drive is needed for a different volume or a higher priority request causes the volume to be dismounted temporarily. In any case, start_time is always the time the volume was originally requested.
end_time	time	20	Time volume was dismounted, either temporarily or indefinitely.
vno	char	8	Volume number (coded in hexadecimal) used internally by the system to identify the volume.
volume_id	char	23	StorHouse volume identification code of dismounted volume. This is a full volume_id specification including media type, recording type, label, and side indicator. The side indicator is preceded by a colon.
volume_set	char	8	Name of volume set in which volume belongs.
extents	num	10	Number of extents written to volume while mounted.
write_bytes	num	20	Number of bytes written to volume while mounted.

Table A-14: vol_dismount (continued)

Field	Type	Max. Len.	Value
read_bytes	num	20	Number of bytes read from volume while mounted.
soft_errors	num	10	Number of soft (recoverable) errors encountered on volume while mounted. (Sometimes the hardware does not notify the software when it encounters a soft error, so not all soft errors are recorded here.)
hard_errors	num	10	Number of hard (unrecoverable) errors encountered on volume while mounted. (A hard error does not always cause a user request to be aborted. Sometimes the software can retry the operation on an alternate volume, area of the volume, or device and complete the user request successfully.)
request_ to_load	time	20	Time request was sent to the operator to load the volume from shelf storage. If the volume was already resident in a library or drive, this time is the same as the time shown in the “resident” field.
resident	time	20	Time volume was resident within a library or drive. If volume was retrieved from shelf or offline storage by an operator, this is the time the system processed the operator’s response to the volume load request. If volume was already resident in a library or drive when the original request was received, this the same as the start_time in the record header.
mounted	time	20	Time volume was mounted in a drive and ready for reading and writing.
src_dev_name	char	6	StorHouse device identification code for the device from which the volume was dismounted.
preempt	num	1	0=normal dismount 1=preemptive priority dismount

vol_movement

Logged after a volume is moved from one device location to another.

- Record Code: 13
- Record Type: 0 (EVENT)
- Number of Fields: 7
- Record Control Parameter: LOG_VOLUME

Table A-15: vol_movement

Field	Type	Max. Len.	Value
record_code	num	2	13
system_id	char	6	System identifier
account_id	char	12	Account id of user who requested volume movement.
user_id	num	10	Userid of the user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time volume move request was received.
end_time	time	20	Time volume movement was completed.
vno	char	8	Volume number (coded in hexadecimal) used internally by the system to identify the volume. If volume has not yet been assigned a volume number, field is null.
volume_id	char	23	StorHouse volume identification code of volume. This volume_id specification includes media type, recording type, label, and side indicator. The side indicator is preceded by a colon. If the volume has not yet been assigned a volume identifier, or the identifier has not yet been verified, field is null.
volume_set	char	8	Name of volume set (or free pool) in which volume belongs. If volume has not yet been assigned to a volume set or free pool, this field is null.

Table A-15: vol_movement (continued)

Field	Type	Max. Len.	Value
reason	char	16	Indicates why the volume was moved: CHANGE – Volume was moved in response to a change file information extent request (e.g., SET FILE/RETENTION). CLEAN – Volume was moved in response to a clean drive request. EXPORT – Volume was moved in response to an EXPORT command. IMPORT – Volume was moved in response to an IMPORT command or a request to enter a blank volume into system. INTERNAL – Volume was moved in response to internal system requirements rather than a user request. LABEL – Volume was moved in response to a write volume label request (for example, ERASE VOLUME command). MOVE – Volume was moved in response to a user's move request. OTHER – Volume was moved for an unknown reason. RECOVERY – Volume was moved in response to a volume label recovery request (for example, CATALOG VSET command). REMOVE – Volume was moved in response to a remove extent request (for example, REMOVE FILE command). TRANSFER – Volume was moved to satisfy a file extent read or write request (for example, PUT command or VRAM read).
src_dev_name	char	6	StorHouse device identification code for source location of volume.

Table A-15: vol_movement (continued)

Field	Type	Max. Len.	Value
dst_dev_name	char	6	StorHouse device identification code for destination location of volume.
request_to_load	time	20	Time request to move volume to or from shelf or offline storage was sent to operator.

device_error

Logged after a device has a hard error. A *hard error* is an error from which the device cannot automatically recover. If the device is a drive, the error may be due to a device error or a media error. A hard error does not always cause a user request to be aborted. Sometimes the software can retry the operation on an alternate volume, area of the volume, or device and complete the user request successfully.

- Record Code: 14
- Record Type: 0 (EVENT)
- Number of Fields: 7
- Record Control Parameter: LOG_DEVICE

Table A-16: device_error

Field	Type	Max. Len.	Value
record_code	num	2	14
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time error occurred.
end_time	time	20	Time processing of error completed. (This value is always the same as start_time.
src_dev_name	char	6	StorHouse device identification code for device that had the error. If the device is unknown or offline, the field is null.

Table A-16: device_error (continued)

Field	Type	Max. Len.	Value
dtype	char	8	Device type: DIRECT IBMLTO TCP SGLTO IBM AIT2 IBM2 AIT3 MAXTOR 0130 SEAGATE 0140 RAID 0150 SYMRAID1 0161 SYMRAID2 4310 SUNRAID1 OL152 FILESYST CENTERA OD152 OL172 OD172 OL600 SMOF551 OL502 OD301 M500 OD321 9710 LD6100 9730 LD8100 9738 HIT525 9740 DLT7000 L700 9490 L180 9840 ACSLS 9940 ADIC HPLTO QUALSTAR
status	num	5	StorHouse status code for device error.
status_message	char	132	Message text corresponding to the status code above.
vno	char	8	Volume number (coded in hexadecimal) used internally by the system to identify the volume. If error does not involve a volume, field is null.
volume_id	char	23	StorHouse volume identification code of volume. If error does not involve a volume, field is null. Volume_id specification includes media type, recording type, label, and (if applicable to error) the colon and side indicator.
error_message	char	256	Text or coded data (such as hexadecimal encoded sense bytes) that provides additional error information. Normally, this data is intended for service representatives.

dev_state_chg

Logged when a device changes state. For example, the system writes this record when the DOWN DEVICE command is executed to change a device from the up state to the down state.

- Record Code: 15
- Record Type: 0 (EVENT)
- Number of Fields: 2
- Record Control Parameter: LOG_DEVICE

Table A-17: dev_state_chg

Field	Type	Max. Len.	Value
record_code	num	2	15
system_id	char	6	System identifier
account_id	char	12	If state change is associated with a single user request, field contains user's account id; otherwise, field is null.
user_id	num	10	If state change is associated with a single user request, field contains user id of user's session; otherwise, field is 0 (zero).
log_time	time	20	Time record was logged.
start_time	time	20	Time state change request was received.
end_time	time	20	Time state change request was completed.
src_dev_name	char	6	StorHouse device identification code for device that changed states.
new_state	num	2	New state of device: 1 = Device online (UP) 2 = Device offline (DOWN) 3 = Device reserved

library_info

Logged at regular intervals to document library device use statistics.

- Record Code: 16
- Record Type: 1 (INTERVAL)
- Number of Fields: 17
- Record Control Parameter: LOG_POLL
- Management Parameter: LOG_INTERVAL

Table A-18: library_info

Field	Type	Max. Len.	Value
record_code	num	2	16
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time interval started.
end_time	time	20	Time interval ended and log record was generated.
src_dev_name	char	6	StorHouse device identification code for library device.
dtype	char	8	Device type: 0130 9710 0140 9730 0150 9738 0161 9740 4310 L700 OL152 L180 OL172 ACSLS OL600 ADIC OL502 QUALSTAR M500
soft_errors	num	10	Number of recoverable (soft) hardware errors during the interval.
hard_errors	num	10	Number of unrecoverable (hard) hardware errors during the interval.
accessor_req	num	10	Number of accessor requests processed during the interval. (Exchange station requests do not affect this value.)
exchange_req	num	10	Number of exchange station requests processed during the interval.

Table A-18: library_info

Field	Type	Max. Len.	Value
full_slots	num	10	Number of full slots in the library at the end of the interval. (0 if library is offline.)
empty_slots	num	10	Number of empty slots in the library at the end of the interval. (All if library is offline.)
freepool_vols	num	10	Number of data volumes in free pool VSETs at the end of the interval.
user_vols	num	10	Number of data volumes in user VSETs at the end of the interval.
cleaning_vols	num	10	Number of cleaning volumes at the end of the interval.
online_vols	num	10	Number of online volumes at the end of the interval. (0 if library is offline.)
shelf_vols	num	10	Number of shelf volumes at the end of the interval. (All if library is offline.)
busy_100	num	10	Number of seconds 100% of the drives were busy during the interval.
busy_75_99	num	10	Number of seconds greater than or equal to 75% but less than 100% of the drives were busy during the interval.
busy_50_74	num	10	Number of seconds greater than or equal to 50% but less than 75% of the drives were busy during the interval.
busy_0_49	num	10	Number of seconds less than 50% of the drives were busy during the interval.

drive_info

Logged at regular intervals to document drive use statistics.

- Record Code: 17
- Record Type: 1 (INTERVAL)
- Number of Fields: 12
- Record Control Parameter: LOG_POLL
- Management Parameter: LOG_INTERVAL

Table A-19: drive_info

Field	Type	Max. Len.	Value
record_code	num	2	17
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero).This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time interval started.
end_time	time	20	Time interval ended and log record was generated.
src_dev_name	char	6	StorHouse device identification code for the drive.
media_type	char	2	StorHouse media type that the drive uses.
recording_type	char	1	StorHouse recording type that the drive uses.
soft_errors	num	10	Number of recoverable (soft) hardware errors during interval.
hard_errors	num	10	Number of unrecoverable (hard) hardware errors during interval.
read_bytes	num	20	Number of bytes read by device during interval.
write_bytes	num	20	Number of bytes written by device during interval.
mounts	num	10	Number of mount requests issued to drive during interval.
mt_retries	num	10	Number of mount retries issued to drive during interval.
rd_retries	num	10	Number of read retries issued to drive during interval.

Table A-19: drive_info (continued)

Field	Type	Max. Len.	Value
wr_retries	num	10	Number of write retries issued to drive during interval
busy	num	10	Number of seconds the drive was busy during the interval.
offline	num	10	Number of seconds the drive was offline during the itnerval.

heartbeat

Logged at regular intervals to document counts of sessions started and commands entered during the interval.

- Record Code: 18
- Record Type: 1 (INTERVAL)
- Number of Fields: 2
- Record Control Parameter: LOG_HEART
- Management Parameter: LOG_INTERVAL

Table A-20: heartbeat

Field	Type	Max. Len.	Value
record_code	num	2	18
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time interval started.
end_time	time	20	Time interval ended and log record was generated.
num_sessions	num	10	Number of user sessions started during interval.
num_commands	num	10	Number of commands executed during interval.

missing_records

Logged after system is able to write to a user log file after one or more records have been discarded. This occurs when all temporary log files on magnetic scratch space are busy. This record specifies the number of discarded records.

- Record Code: 19
- Record Type: 0 (EVENT)
- Number of Fields: 1
- Record Control Parameter: None. Logging of this parameter is always enabled.

Table A-21: missing_records

Field	Type	Max. Len.	Value
record_code	num	2	19
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time first record was discarded.
end_time	time	20	Time system was able to write to log file again.
num_discarded	num	10	Number of discarded user log records.

vol_mount

Logged when a volume is mounted in a drive.

- Record Code: 20
- Record Type: 0 (EVENT)
- Number of Fields: 6
- Record Control Parameter: LOG_VOLUME

Table A-22: vol_mount

Field	Type	Max. Len.	Value
record_code	num	2	20
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time volume was originally requested.
end_time	time	20	Time volume was mounted.
vno	char	8	Volume number (coded in hexadecimal) used internally by the system to identify the volume.
volume_id	char	23	StorHouse volume identification code of the mounted volume. This is a full volume_id specification including media type, recording type, label, and side indicator. The side indicator is preceded by a colon.
volume_set	char	8	Name of volume set that contains the mounted volume.
dst_dev_name	char	6	StorHouse device identification code for the device where the volume was mounted.
file_sysid	num	5	System identifier of file that was the object of a file-oriented request.
file_fno	num	10	File number of file that was the object of a file-oriented request.

op_message

Logged when a message is sent to the operator console.

- Record Code: 21
- Record Type: 0 (EVENT)
- Number of Fields: 3
- Record Control Parameter: LOG_OPERATOR

Table A-23: op_message

Field	Type	Max. Len.	Value
record_code	num	2	21
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time request or reply was received.
end_time	time	20	Time close was returned to host from StorHouse.
request_num	num	10	Unique identification number for the request.
is_reply	num	1	Type of operator message: 0 = informational message or request 1 = reply (via CONSOLE /REPLY)
message_text	char	512	Text of request or reply.

file_copy

Logged when one or more extents of a file are written to their resident file sets. This record applies to the user commands ARCHIVE, CREATE BACKUP, CREATE PRIMARY, RELOCATE, RECOVER VOLUME, RETIRE VOLUME, and REPLICATE.

- Record Code: 22
- Record Type: 0 (EVENT)
- Number of Fields: 6
- Record Control Parameter: LOG_COPY

Table A-24: file_copy

Field	Type	Max. Len.	Value
record_code	num	2	22
system_id	char	6	System identifier
account_id	char	12	Account id of user who accessed the file.
user_id	num	10	User id of user's session.
log_time	time	20	Time record was logged.
start_time	time	20	Time file copy request command was initiated.
end_time	time	20	Time file was closed.
command	num	1	1 = ARCHIVE copied/extended file 2 = CREATE BACKUP copied/extended file 3 = CREATE PRIMARY copied file 4 = RELOCATE moved file 5 = RECOVER VOLUME moved file 6 = RETIRE VOLUME moved file 7 = REPLICATE copied file 8 = REPLICATION_SERVER (A file copy action occurred on a secondary StorHouse system due to a REPLICATE command on a primary StorHouse system.)
status	num	5	Copy status (decimal number)
file_sysid	num	5	System identifier of specified file
file_fno	num	10	File number of specified file
num_extents	num	10	Number of extents copied/moved
bytes	num	20	Total bytes copied/moved

extent_transfer

Logged whenever an extent is transferred from the performance buffer to its resident file set or from its resident file set to the performance buffer.

- Record Code: 23
- Record Type: 0 (EVENT)
- Number of Fields: 6
- Record Control Parameter: LOG_XFER

Table A-25: extent_xfer

Field	Type	Max. Len.	Value
record_code	num	2	23
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	Null. This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time transfer request was received.
end_time	time	20	Time transfer completed.
command	num	1	Command used to move or copy the extent: 1 = BACKUP
status	num	5	Transfer status code (decimal number)
file_sysid	num	5	System identifier of specified file (null if unknown)
file_fno	num	10	File number of specified file (null if unknown)
extent_id	num	10	Identifier of extent copied/moved
bytes	num	20	Total bytes copied/moved

statistics

Logged at mount, dismount, and when internal counters overflow for drives. Logged at intervals for libraries and auxiliary devices (such as grippers and exchange stations). Documents device error and usage statistics. These statistics are displayed for the interval specified in the LOG_INTERVAL system parameter.

- Record Code: 24
- Record Type: 0 (EVENT)
- Number of Fields: 27
- Record Control Parameter: LOG_DEVICE

Table A-26: statistics

Field	Type	Max. Len.	Value
record_code	num	2	24
system_id	char	6	System identifier
account_id	char	12	Null. This field is not applicable.
user_id	num	10	0 (zero). This field is not applicable.
log_time	time	20	Time record was logged.
start_time	time	20	Time statistics were generated.
end_time	time	20	Time statistics were received by StorHouse (end_time is always the same as start_time).
status	num	5	StorHouse status code for device error, if applicable; otherwise, zero.
src_dev_name	char	6	StorHouse device identification code for reporting device.
manufacturer	num	10	Internal StorHouse code for device manufacturer.
flagword	num	10	Internal use only
type	num	3	Device type: 1 = DRIVE 2 = LIBRARY 3 = ACCESSOR 4 = EXCHANGE STATION 5 = SHELF
read	num	10	Number of blocks read. This field is valid for drives only; otherwise, 0 (zero).
read_retry	num	10	Number of blocks reread due to errors. This field is valid for drives only; otherwise, 0 (zero).

Table A-26: statistics (continued)

Field	Type	Max. Len.	Value
read_error	num	10	Number of blocks read but not recovered. This field is valid for drives only; otherwise, 0 (zero).
write	num	10	Number of blocks written. This field is valid for drives only; otherwise, 0 (zero).
write_retry	num	10	Number of blocks rewritten due to errors. This field is valid for drives only; otherwise, 0 (zero).
write_error	num	10	Number of unwritable blocks. This field is valid for drives only; otherwise, 0 (zero).
seek	num	10	Number of seeks. This field is valid for drives only; otherwise, 0 (zero).
seek_retry	num	10	Number of seeks retried due to errors. This field is valid for drives only; otherwise, 0 (zero).
seek_error	num	10	Number of unrecoverable seeks. This field is valid for drives only; otherwise, 0 (zero).
erase	num	10	Number of blocks erased. This field is valid for drives only; otherwise, 0 (zero).
erase_retry	num	10	Number of blocks erased again due to errors. This field is valid for drives only; otherwise, 0 (zero).
erase_error	num	10	Number of blocks unable to be erased. This field is valid for drives only; otherwise, 0 (zero).
laser_on	num	10	Number of hours the laser has been turned on. This field is valid for drives only; otherwise, 0 (zero).
power_on	num	10	Number of hours the drive has been powered up. This field is valid for drives only; otherwise, 0 (zero).
vno	char	8	Volume number (coded in hexadecimal) used internally by the system to identify a volume. If the volume has not yet been assigned a volume number, the field is null. This field is valid for drives only; otherwise, null.

Table A-26: statistics (continued)

Field	Type	Max. Len.	Value
volume_id	char	23	StorHouse volume identification code of the volume mounted in the drive. This volume_id specification includes media type, recording type, label, and side. If the volume has not yet been assigned a volume identifier, or the identifier has not yet been verified, the field is null. This field is valid for drives only; otherwise, null.
scan	num	10	Number of bar codes scanned. This applies only to libraries with bar code readers; otherwise, 0 (zero).
scan_retry	num	10	Number of bar codes rescanned due to errors. This applies only to libraries with bar code readers; otherwise, 0 (zero).
scan_error	num	10	Number of unreadable bar codes. This applies only to libraries with bar code readers; otherwise, 0 (zero).
misc_op	num	10	Number of general device operations. This field is valid for libraries, accessors, exchange stations, and shelves only; otherwise, 0 (zero).
misc_op_retry	num	10	Number of general device operations retried due to errors. This field is valid for libraries, accessors, exchange stations, and shelves only; otherwise, 0 (zero).
misc_op_error	num	10	Number of failed general device operations. This field is valid for libraries, accessors, exchange stations, and shelves only; otherwise, 0 (zero).

connect

Logged when an application attempts to connect to a StorHouse database.

- Record Code: 25
- Record Type: 0 (EVENT)
- Number of Fields: 2
- Record Control Parameter: LOG_SIGN

Table A-27: connect

Field	Type	Max. Len.	Value
record_code	num	2	25
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted the access request.
user_id	num	10	User id of user's session (SQL engine ttno).
log_time	time	20	Time record was logged.
start_time	time	20	Time connect processing was received.
end_time	time	20	Time connect processing was completed. (This value is always the same as start_time.)
database_name	char	32	Name of the StorHouse database specified in the connect request.
sql_status	num	7	SQLCODE from connection processing and account validation.

disconnect

Logged when an application disconnects from a StorHouse database.

- Record Code: 26
- Record Type: 0 (EVENT)
- Number of Fields: 1
- Record Control Parameter: LOG_SIGN

Table A-28: disconnect

Field	Type	Max. Len.	Value
record_code	num	2	26
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted the access request.
user_id	num	10	User id of user's session (SQL engine ttno).
log_time	time	20	Time record was logged.
start_time	time	20	Time connection was established.
end_time	time	20	Time disconnect processing was completed.
database_name	char	32	Name of the StorHouse database to which the user was connected.

security_connect

Logged when a StorHouse/RM connect request is denied due to a nonexistent account id or incorrect password. The violation may be due to an honest error on the part of a user or an attempt to access an account without authorization. Multiple violations (multiple records) with a variety of account and/or password combinations are more indicative of a true attempt to violate system security.

- Record Code: 27
- Record Type: 0 (EVENT)
- Number of Fields: 1
- Record Control Parameter: LOG_SECURITY

Table A-29: security_connect

Field	Type	Max. Len.	Value
record_code	num	2	27
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted the access request.
user_id	num	10	User id of user's session (SQL engine ttno).
log_time	time	20	Time record was logged.
start_time	time	20	Time connect processing was received.
end_time	time	20	Time connect processing was completed. (This value is always the same as start_time.)
database_name	char	32	Name of the StorHouse database specified in the connect request.

sql_statement

Logged when StorHouse/RM receives a user's SQL statement.

SQL statements that are longer than 500 bytes are called *long SQL statements*. Each SQL statement record contains information for 500 bytes or less. If an SQL statement is 1025 bytes long, for example, there will be three statement records in the user log for that SQL statement. The statement_size in the first record contains the length of the complete statement. If the statement is longer than 500 bytes, additional segments exist.

Note If the value of the segment_number field is 1, use all fields. If the value of the segment_number field is greater than 1, use only the transaction_id, statement_id, and statement fields.

- Record Code: 28
- Record Type: 0 (EVENT)
- Number of Fields: 8
- Record Control Parameter: LOG_SQL_STMT

Table A-30: sql_statement

Field	Type	Max. Len.	Value
record_code	num	2	28
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted the access request.
user_id	num	10	User id of user's session (SQL engine ttno).
log_time	time	20	Time record was logged.
start_time	time	20	Time parsing was received.
end_time	time	20	Time parsing was completed.
database_name	char	32	Name of the StorHouse database to which the user was connected.
sql_status	num	7	SQLCODE from statement parsing.
transaction_id	char	24	Identifier of the transaction being logged. Together, this field and the statement_id field uniquely identify the SQL statement being logged.
statement_id	char	24	Identifier of the SQL statement being logged. Together, this field and the transaction_id field uniquely identify the SQL statement being logged.

Table A-30: sql_statement (continued)

Field	Type	Max. Len.	Value
segment_number	num	3	Unique identifier for the relative segment to which this part of the SQL statement belongs. (1 for first or only segment, 2 and so on for any additional segments.)
statement_type	char	5	SQL statement type identifier. Values can be: • QUERY = Query • DML = Data Manipulation Language • DDL = Data Definition Language
statement_size	num	5	Size of the entire SQL statement, as entered.
statement	char	500	Entire SQL statement, as entered by the user, or a unit of a long SQL statement. Each unit is no longer than 500 bytes.

sql_transactions

Logged when an SQL transaction is completed.

- Record Code: 29
- Record Type: 0 (EVENT)
- Number of Fields: 8
- Record Control Parameter: LOG_SQL_TRANS

Table A-31: sql_transactions

Field	Type	Max. Len.	Value
record_code	num	2	29
system_id	char	6	System identifier
account_id	char	12	Account id of user who submitted the access request.
user_id	num	10	User id of user's session (SQL engine ttno).
log_time	time	20	Time record was logged.
start_time	time	20	Time transaction was received.
end_time	time	20	Time transaction was completed (COMMIT /ROLLBACK received).
database_name	char	32	Name of the StorHouse database to which the user was connected.
transaction_id	char	24	Unique identifier for the transaction being logged.
num_statement	num	10	Specifies the number of SQL statements received in the transaction.
num_row_fetched	num	20	Specifies the number of rows fetched.
cpu_time	num	10	Specifies the engine CPU time in milliseconds.
wait_time	num	10	Specifies the time waiting for next user request in milliseconds.
transaction_result	char	9	Specifies whether the transaction was committed or rolled back. The values are COMMIT or ROLLBACK.
bytes_fetched	num	20	Specifies the total number of bytes transferred to the client during the transaction.

sql_file_close

Logged when a StorHouse/RM file is permanently closed or at the end of a transaction. These records are written for queries (read) and data loads (write). Records may also be written if a data load fails (stops at checkpoint), which would be a CHKPT type record. A data load that completes and is confirmed will always generate a WRITE record.

- Record Code: 30
- Record Type: 0 (EVENT)
- Number of Fields: 32
- Record Control Parameter: LOG_FILE

Table A-32: sql_file_close

Field	Type	Max. Len.	Value
record_code	num	2	30
system_id	char	6	System identifier
account_id	char	12	Account ID of user who accessed the file.
user_id	num	10	User ID of user's session (ttno).
log_time	time	20	Time record was logged.
start_time	time	20	Time that the first open for this file in the current transaction completes. For a loader restart, applies only to that restart.
end_time	time	20	Time file was permanently closed. For a loader restart, applies only to that restart.
database_name	char	32	Name of the StorHouse database to which the user was connected.
transaction_id	char	24	For reads, unique identifier for the transaction from which the file was accessed. For writes and checkpoints, contains the segment ID. If applicable, a ":" followed by a sequence ID is appended.
status	num	5	StorHouse status code returned in response to close request, highest severity status code from any I/O request, or loader status if no close or I/O failure.

Table A-32: sql_file_close (continued)

Field	Type	Max. Len.	Value
status_msg	char	132	Text message corresponding to the value of the status field.
mode	num	1	Mode specified by user in open request: 1 = READ (query) 2 = WRITE (data loader) 6 = CHKPT (data loader stopped at checkpoint)
method	num	2	Access method specified by the data loader in open request: 1 = SEQUENTIAL 2 = RECORD 8 = RECORD to SEQUENTIAL 9 = RECORD direct with fetch-ahead
file_sysid	num	5	System identifier of the specified file.
file_fno	num	10	File number of the specified file.
filename	char	56	Name of the specified file.
group	char	8	Name of group in which the specified file is located.
organization	num	5	Organization of specified file: 6 = RM DATA 7 = RM HASH INDEX 8 = RM VALUE INDEX 9 = Reserved for future use
volume_set	char	8	Name of volume set in which the specified file is located.
file_set	char	8	Name of file set in which the specified file is located.
trans_pages	num	10	The number of pages RM wrote to or read from StorHouse. For checkpoints, only pages checkpointed to point of failure.
trans_rows	num	20	For reads, null. For writes, the number of rows RM wrote to StorHouse. For checkpoints, only rows checkpointed to point of failure.

Table A-32: sql_file_close (continued)

Field	Type	Max. Len.	Value
trans_bytes	num	20	For reads and writes, the number of bytes RM wrote to or read from StorHouse. For checkpoints, only bytes checkpointed to point of failure.
tbl_name	char	32	Name of the table associated with this file (even if this is an index file).
idx_name	char	32	Name of the index if this file is associated with an index (null if a table file).
owner	char	32	Owner of the associated table.
obj_id	num	10	ID for the table/index.
reopens	num	10	Number of file re-opens.
tclosed_int	num	10	Length of time that file was in “temporary close” state, in milliseconds.
segopen_wait	num	10	Elapsed time spent waiting for segment file opens, in milliseconds (includes DF access time for reads). For a loader restart, applies only to that restart.
extopen_wait	num	10	Elapsed time spent waiting for extent opens, including re-opens, in milliseconds (for reads, includes the I/O time to read the first record). For a loader restart, applies only to that restart.
reopen_wait	num	10	Elapsed time spent waiting for extent re-opens, in milliseconds. For a loader restart, applies only to that restart.
segclose_wait	num	10	Elapsed time spent waiting for segment file closes, in milliseconds (includes DF write time for writes). For a loader restart, applies only to that restart.
extclose_wait	num	10	Elapsed time spent waiting for extent closes, including temporary closes, in milliseconds.

Table A-32: sql_file_close (continued)

Field	Type	Max. Len.	Value
io_wait	num	10	Elapsed time spent waiting for file I/O operations, in milliseconds. For a loader restart, applies only to that restart.
extents	num	10	Number of extents processed, including DF extents. For checkpoints, extents written so far.
fetched_pages	num	10	For reads, number of pages actually processed by the RM engine. For writes and checkpoints, null.
fetched_rows	num	20	For reads, number of rows actually processed by the RM engine. For writes and checkpoints, null.
fetched_bytes	num	20	For reads, number of bytes actually processed by the RM engine. For writes and checkpoints, null.

Appendix

B

Sample Data Record

This appendix presents a sample data record for a security violation attempt that generates a security_group data record. The scenario is explained first, followed by descriptions of the scenario's data dictionary and data records.

Scenario

Consider the following scenario:

- At 4:30:00 p.m. on February 26, 1998, a user with the account identification code of SMITH and a user identification code of 18, executes the following command:

  GET FILENAME /GROUP=SYSTEM

  This command requires access privilege to the group SYSTEM, which the SMITH account does not have.

- The LOG_SECURITY parameter is set to TRUE, which enables logging of security violation attempt records.

- The temporary user log file on magnetic disk scratch space is available for record logging.

Data Dictionary Records

The information contained in the security_group data record is defined by two data dictionary records:

- The *record header descriptor* defines the seven fields common to the beginning of all data records for this release. The content of the record header descriptor is shown in Table B-1.
- The *data record descriptor* corresponding to the security_group data record defines the specific fields associated with this type of data record. The content of a security_group data record descriptor is shown in Table B-2.

Table B-1: Record Header Description

Field	Type	Length	Value
record_code	num	1	0
record_name	char	15	data_record_hdr
record_type	num	1	0
number_of_fields	num	1	7
field_name field_type field_max_len	char num num	11 1 3	record_code 1 2
field_name field_type field_max_len	char num num	9 1 3	system_id 0 6
field_name field_type field_max_len	char num num	10 1 3	account_id 0 12
field_name field_type field_max_len	char num num	7 1 3	user_id 1 10
field_name field_type field_max_len	char num num	8 1 3	log_time 2 20
field_name field_type field_max_len	char num num	10 1 3	start_time 2 20
field_name field_type field_max_len	char num num	8 1 3	end_time 2 20

Table B-2: security_group Data Record Descriptor

Field	Type	Length	Value
record_code	num	2	6
record_name	char	20	security_group
record_type	num	1	0
number_of_fields	num	3	2
field_name	char	20	command
field_type	num	1	0
field_max_len	num	3	255
field_name	char	20	group
field_type	num	1	0
field_max_len	num	3	8

Table B-2 reveals the following information:

- The record_code is a numeric field with a maximum length of two characters. This field identifies the data record type. In this case, the data record type corresponds with record code 6.

- The record_name is a character field with a maximum length of 20 characters. This field identifies the data record name. In this case, the record name is security_group.

- The record_type is a numeric field with a maximum length of one character. This field identifies the record type as one that is logged after an event (0) or interval (1). In this case, the value is 0, indicating that the record is logged after a certain event, namely a group security violation attempt.

- The number_of_fields is a numeric field with a maximum length of three characters. This field identifies the number of specific fields associated with this record type. This value does not include the seven fields common to the beginning of all record types. In this case, there are two specific fields associated with record code number 6.

- For each field specified in number_of_fields, there are three field format definition fields: field_name, field_type, and field_max_len.

    - Field_name is a character field with a maximum length of 20 characters. This field identifies the field name. In this case, the two field names are command and group.

- Field_type is a numeric field with the maximum length of one character. This field specifies the field type of 0 (character string), 1 (numeric), or 2 (timestamp). In this case, both the command and group fields are field type 0, indicating that they are character strings.

- The field_max_len is a numeric field with a maximum length of three characters. This field specifies the maximum number of characters in a field. In this case, the command field has a maximum length of 255 characters, and the group field has a maximum length of eight characters.

Data Record

Using the user log file's record header descriptor and appropriate data record descriptor, the scenario generates a data record as shown in Table B-3.

Table B-3: Scenario Data Record

Field	Type	Max. Len.	Value
record_code	num	2	6
system_id	char	6	1
account_id	char	12	SMITH
user_id	num	10	18
log_time	time	20	26-FEB-1998 16:30:03
start_time	time	20	26-FEB-1998 16:30:00
end_time	time	20	26-FEB-1998 16:30:00
command	char	255	GET FILENAME /GROUP=SYSTEM
group	char	8	SYSTEM

A description of each data record field follows:

- record_code is a numeric field with a maximum length of two characters. In this case, the security violation corresponds to data record descriptor number 6, the data record descriptor for a security_group data record.

- system_id is a character field with a maximum length of six characters. This field identifies the StorHouse system that logged the error. In this case, the StorHouse system identifier is 1.

- account_id is a character field with a maximum length of 12 characters. This field identifies the account that caused the violation. In this case, the account identification code is SMITH.

- user_id is a numeric field with a maximum length of 10 characters. This field uniquely identifies the user session. In this case, the user identification is 18.

- log_time is a timestamp field with a maximum length of 20 characters. This field specifies the time when the record was logged. In this case, the record was logged on February 26, 1998, at 4:30:03 p.m.

- start_time is a timestamp field with a maximum length of 20 characters. This field specifies the time when the invalid command was entered. In this case, the invalid command was entered on February 26, 1998, at 4:30:00 p.m, as specified in the scenario.

- end_time is a timestamp field with a maximum length of 20 characters. This field specifies the time when command execution completed. In this case, the invalid command completed processing on February 26, 1998, at 4:30:00 p.m.

  Note: The values for start_time and end_time are the same. This is always the case for commands that have short durations.

- command is a character field with a maximum length of 255 characters. This field displays the command that caused the group security violation. In this case, the command is GET FILENAME /GROUP=SYSTEM.

- group is a character field with a maximum length of eight characters. This field displays the group for which the security violation attempt was logged. If no group is specified in the executed command, the account's default group is used. In this case, the group is SYSTEM.

Appendix

C

User Log Message Protocol

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

This appendix:

- Explains how the server and its clients use the message protocol
- Defines the nine message types
- Describes each field in the message layout
- Lists server error messages
- Provides a sample client program.

Understanding the Message Protocol

The user log server is a StorHouse process that communicates with client applications through TCP/IP sockets. It requires a network connection and can talk with clients on any platform capable of communicating across the network.

Both client and server must strictly adhere to the user log message protocol defined in this appendix. Clients use this protocol to request a user log record or a new copy of the user log data dictionary. Clients may also check the server for pending data, terminate the connection, or tell the server to skip past all buffered data.

The server uses the message protocol to send the client a new user log data dictionary, or a complete or partial user log record. The server can also report an error, acknowledge a message, or ask the client to terminate the connection.

User Log Server Port Number

Clients use a specific StorHouse port to connect to the user log server. The default port number for the user log server is always one greater than the default StorHouse host port number. The SM_LINKNAME parameter in the StorHouse host configuration file contains the installed value of the StorHouse host port number. Currently, the default StorHouse host port number is 1200. The default port number for the user log server is 1201.

The user log server port number is configurable to avoid possible conflicts with other port assignments your site may already have in place. SGI recommends that you write clients to work with any port number.

Client Design Considerations

SGI recommends that client applications be designed to read messages as fast as the user log server can write them. The exact read rate depends on the user log activity (for example, the average number of writes to the user log each day) at your site.

Clients should remain active. While connected, they should continually check the server for pending data. Idle or slow clients should terminate rather than maintain their connections so that the server can reclaim disk space used for record buffering.

Client/server operations have no effect on StorHouse writes to the standard user log file. These writes continue as usual.

Enabling the User Log Server

StorHouse collects user log records for the user log server in special message buffers. The StorHouse system parameter LOG_SERVER controls whether records are written to these message buffers. Always set LOG_SERVER to TRUE to enable real-time collection of user log information. The value of LOG_SERVER has no effect on whether records are written to the standard user log.

Message Definitions

This section describes the message record layout and defines the nine message types that are passed between the user log server and its clients. It also lists server error messages sent to the client.

Message Record Layout

Table C-1 defines the message record layout. Each message has a maximum length of 8198 bytes, which includes a 5-byte header, a 1-byte message type, and up to 8192 bytes of data. The header consists of an asterisk (*) in the first byte and a 4-byte length field in network neutral format. The shaded fields in the following table make up the header component of each record.

Table C-1: Message Record Layout

Bytes 0 - 4 make up the header common to all messages.

Byte Position	Field Description
0	Must contain an asterisk (*).
1 - 4	Contains the record length, excluding the 5-byte header, in network neutral format (host-to-network-length/network-to-host-length).
5	Contains the message type. Valid values are the ASCII values 1 - 9. Refer to Table C-1 for a description of each message type.
6 - end of record (maximum data size is 8192 bytes)	Contains data in one of the following formats: • a complete or partial user log record • a user log data dictionary • an error message • the characters Y or N in the first byte. Message types 1, 2, 3, 4, and 9 contain no data.

Message Types

The user log message protocol supports the following nine message types:

- REQUEST
- PING
- TERMINATECLIENT
- DICTIONARY
- RECORDCOMPLETE
- RECORDPARTIAL
- ACKNOWLEDGE
- ERROR
- SKIP

Table C-2 defines each message type in detail.

Table C-2: Message Types

Type	Value	Message Description	Data Portion Contains	Possible Responses
1	REQUEST	Client sends the server a request for the next record or action.	Nothing.	RECORDCOMPLETE RECORDPARTIAL ERROR TERMINATECLIENT DICTIONARY
2	PING	Client checks the server for pending data.	Nothing.	ACKNOWLEDGE
3	TERMINATECLIENT	Server tells the client to close the connection. Client notifies the server that it is intentionally terminating the connection.	Nothing.	None. None.
4	DICTIONARY	Server asks the client to request a user log dictionary. Client asks the server to send a user log dictionary.	Nothing.	Client must respond with DICTIONARY. Server can respond with: RECORDCOMPLETE RECORDPARTIAL ERROR
5	RECORDCOMPLETE	Server tells the client that data in the message completes the client's read request.	A complete user log record or the last portion of a user log record.	None.
6	RECORDPARTIAL	Server tells the client that data in the message is incomplete and that the client should read again until it receives a RECORDCOMPLETE message.	A portion of a user log record.	None.
7	ACKNOWLEDGE	Server provides feedback to the client in the first byte of the data portion of the message. Feedback indicates that a message was received and processed.	A Y or N in the first byte. • Y - data is available without blocking. • N - no data is available.	None.

Table C-2: Message Types

Type	Value	Message Description	Data Portion Contains	Possible Responses
8	ERROR	Server tells client that an error occurred. The data portion of the message contains the specific error text. Refer to Table C-3 for a list of messages.	A text message that describes the error.	None. Clients should process ERROR messages just like they process TERMINATECLIENT messages.
9	SKIP	Client instructs the server to advance past any buffered data so that all future reads go against the most current user log information.	Nothing.	ACKNOWLEDGE ERROR TERMINATECLIENT

Server Error Message Text

Table C-3 defines the text portion of ERROR messages sent by the server to the client.

Table C-3: Error Message Text

Message	Description
Invalid Message Header	The first byte of the message does not contain an asterisk (*) or the message length is invalid.
Buffer File Open Failed	The server could not open the buffer that contains the user log records.
Error Reading Record Size	The server determined that the length in the data buffer is invalid.
Can’t Open Dictionary	The server is unable to open the data dictionary and send it to the client.
Error - Dictionary Is Invalid	An error occurred while reading the data dictionary.
Client Protocol Error - (%c) Not Expected	The client application does not follow the standard message protocol. The character denoted by “%c” is not valid.
No buffer files present	The StorHouse is not logging user log records for the user log server.

Sample Client Program

The following sample program shows how a client formats message records, sends then to the user log server, and reads server replies. It is not meant to represent the programming style of a production client.

This sample program:

- Defines the nine message types and required constants and functions
- Creates a socket
- Obtains the host data structure
- Connects to the server at the specified port (1201)
- Sends a REQUEST record to the server
- Reads the server's response
- Prints one field from the user log record sent from the server.

The .h files define all functions, structures, and variables that are not explicitly defined in this program.

```
/* CLIENT */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>


#define PROGRAM "Client1"

/* public messages sent by client */
#define  REQ   '1'/* REQUEST FOR DATA */
#define  PING  '2'/* PING */

#define  SKIP  '9'/* FAST FWD TO END OF CURRENT */
/* READER LOG */

/* handled by both */
#define  TERM  '3'/* TERMINATE CLIENT */
#define  DICT  '4'/* DICTIONARY*/

/* sent by server */
#define  ACK   '7'/* ACKNOWLEDGE */
#define  ERR   '8'/* ERROR */
#define  RECC  '5'/* COMPLETE RECORD RETURNED */
#define  RECP  '6'/* PARTIAL RECORD RETURNED */

#define HOSTNAME  "smdev4"
#if 1
#define PORT 1201
#else
#define PORT atoi(argv[1])
#endif
```

The client uses the default user log server port 1201.

```
#define BUFFERSIZE  4097/* 1-byte message type plus */
/* 4096-bytes of data */


int WriteMessage( int sock, char *buf, int size );
int ReadMessage( int sock, char buf[] );

/*This function writes messages to the server.*/
int WriteMessage( int sock, char *buf, int size )
{
int length;
static char star = '*';
long lsize;

/* output message header */
lsize = htonl(size);/* make network neutral format */
length = write(sock, &star, 1);
if (length != 1) {
    fprintf(stderr, "Error writing to server\n");
    return 0;/* not successful */
}
length = write(sock, &lsize, sizeof(lsize));
if (length != sizeof(lsize)) {
    fprintf(stderr, "Error writing to server\n");
    return 0;/* not successful */
}

/* outout message */
length = write(sock, buf, size);
if (length != size) {
    fprintf(stderr, "Error writing to server\n");
    return 0;/* not successful */
}

return 1;/* successful */
}

/* This function reads messages from the server.*/
int ReadMessage( int sock, char buf[] )
{
int rval, cnt = 0;
long lsize;
char star;
memset(buf, 0, BUFFERSIZE);
rval = read(sock, &star, 1);
if (rval < 1)/* not sucessful */
return rval;

rval = read(sock, &lsize, sizeof(lsize));
if (rval < 1)/* not sucessful */
return rval;

/* validate the header */
if ((star != '*') || (lsize > BUFFERSIZE)) {
```

The client is responsible for converting the message length to a network neutral format.

```
fprintf(stderr, "Invalid Message Header\n");
return -1;
}

rval = read(sock, buf, lsize);
/* A production client should validate that it's */
/* actually read lsize number of bytes */
if (rval < 1)/* not sucessful */
return rval;

return 1;/* Successful */
}

main( int argc, char *argv[] )
{
int sock, rval;
struct sockaddr_in server;
struct hostent *hp;
char buf[BUFFERSIZE];
int term = 0;/* indicate the program should exit */

/* create socket */
sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock == -1) {
perror("Socket open failed");
exit(1);
}

/* get host data structure */
server.sin_family = AF_INET;
hp = gethostbyname(HOSTNAME);
if (hp == (struct hostent*)0) {
fprintf(stderr, "Unknown host %s\n", HOSTNAME);
exit(2);
}

memcpy((char*)&server.sin_addr, (char*)hp->h_addr,
hp->h_length);

/* connect to the server at the port specified */

server.sin_port = htons(PORT);
if (connect(sock, (struct
   sockaddr*)&server,sizeof(server)) == -1) {
perror("Socket connect failed");
exit(1);
}

do {
 buf[0] = REQ;/* request some data */

    /* use the connection - send the server a message*/
    if (!WriteMessage(sock, buf, 1)) {
    perror("Socket write failed");
```

```
    exit(1);
    }
    memset(buf, 0, BUFFERSIZE);
 /* Read message sent by server */
    rval = ReadMessage(sock, buf);
    if (rval == -1) {
    perror("Socket read failed");
    exit(1);
    } else if (rval == 0) {
    fprintf(stderr,
"%s: Server closed the connection - DISORDERLY\n",
PROGRAM);
    break;
    }
    switch(buf[0]) {
    case ACK:
    fprintf(stderr,
 "%s:ACK received - Data Present='%c'\n",
 PROGRAM, buf[1]);
    break;
    case TERM:
    fprintf(stderr, "%s:Server sent TERM\n",
 PROGRAM);
    term = 1;
    break;
    case RECC:
     fprintf(stderr, "%s:%s\n", PROGRAM,&buf[1]);
    break;
    case ERR:
    fprintf(stderr, "%s:Server Error '%s'\n",
 PROGRAM, &buf[1]);
    term = 1;
    break;
    default:
    fprintf(stderr, "%s: Protocol Error (%c) not
 expected", PROGRAM, buf[0]);
    term = 1;
    }
} while(!term);

/* close the connection */
close(sock);

/* terminate with success */
exit(0);
}
```

Insert user code here to process the retrieved user log record.

Index

A

C

D

E

F

H

I

J

L

M

N

O

P

R

S

T

U

V