April 2021

SQLite Data Recovery Specification, Test Assertions, and Test Cases

Version 1.0

National Institute of
Standards and Technology
U.S. Department of Commerce

Disclaimer

Abstract

This specification defines requirements, test assertions, and test cases for basic methods of recovering and reporting evidence as contained within SQLite databases. This also includes their associated journal mode log files. The specification does not address more advanced methods of recovery of data as stored in these files. Today, millions of different applications reside on smartphones, computers, and IoT devices that all store data in this format.

This document defines SQLite forensic data recovery tool requirements. These requirements are used to derive test assertions, statements of conditions that are then checked after a test case is run. Each test assertion is covered by one or more test cases consisting of a test protocol and the expected test results. The test case protocol specifies detailed procedures for setting up the test, executing the test, and measuring the test results.

Thanks and appreciation to Sam Brothers for his collaboration with co-authoring the SQLite Data Recovery and assistance in designing SQLite datasets for testing purposes.

Comments and feedback are welcome. This document, and future revisions, are available for download at: https://www.cftt.nist.gov/SQLite_Forensic_Tools.htm.

TABLE OF CONTENTS

1 Introduction

There is a critical need in the law enforcement community to ensure the reliability of computer forensic tools. A capability is required to ensure that forensic tools consistently produce accurate, repeatable, and objective test results. The goal of the Computer Forensic Tool Testing (CFTT) project at the National Institute of Standards and Technology (NIST) is to establish a methodology for testing computer forensic tools by the development of functional specifications, test procedures, test criteria, and test sets. The results provide the information necessary for toolmakers to improve tools, for users to make informed choices about acquiring and using computer forensics tools, and for interested parties to understand the tools' capabilities. This approach for testing computer forensic tools is based on well-recognized international methodologies for conformance testing and quality testing. This project is further described at http://www.cftt.nist.gov/.

The Computer Forensics Tool Testing (CFTT) program is a joint project of the Department of Homeland Security (DHS) Science and Technology Directorate, and the National Institute of Standards and Technology.

2 Purpose

This specification defines requirements, test assertions, and test cases for SQLite Data Recovery (SDR) Forensics Tools capable of performing the following tasks:

1. Displaying recovered SQLite database information to the user,
2. identifying, categorizing, and reporting upon Write-Ahead Log (WAL) and Rollback Journal data, and
3. sequencing wal journal data.

The requirements are used to derive test assertions, statements of conditions that are checked after a test case is run. Each test assertion is covered by one or more test cases consisting of a test protocol and the expected test results. The test case protocol specifies detailed procedures for setting up the test, executing the test, and measuring the test results.

3 Scope

The scope of this specification is limited to software tools capable of presenting recovered data stored within SQLite databases. This also includes stand-alone SQLite forensic tools that provide additional functionality. This specification is general and capable of being adapted to other database formats should they become more widely used.

The intended audience for this document is forensic examiners with an existing basic knowledge of SQLite. This document does not address every detail about how SQLite works. Rather, the scope of this material is at a higher level. Should the reader be interested in additional file format details they can be found at: www.sqlite.org or they should seek specialized SQLite forensics training.

The topics covered within this specification addresses recovered SQLite data. The reporting of active SQLite data is commonplace in modern mobile forensic tools and therefore, covered in *Mobile Device Forensic Tool Specification, Test Assertions, and Test Cases.*

4 Definitions

This glossary defines terms used within this document.

Active Data – Table information that comprises the current state of the database (and all associated journal mode files) as of the latest successful commit.

Binary Large OBject (BLOB) – A Binary Large Object is a string of binary data stored as a single entity within a database management system. BLOB's can typically be images, audio, plist or other multimedia objects.

Commit – This SQLite command is the transactional command used to save changes invoked by a transaction to the database.

Data Element – Data contained in a single cell of a row in the table of a given SQLite database.

Journal Mode – Functionality that provides rollback abilities in accordance with Atomic, Consistent, Isolated, and Durable (ACID) transactions. This refers to either a -journal or -wal file.

Journal Sequencing – Ordering of transactions within the -wal journal file and any related data in the database.

Page – A fixed-size contiguous block of data within an SQLite database, WAL, or journal file. The size of a page is a power of two between 512 and 65,536 bytes. All pages within a database are of the same size.

Recoverable Row – This refers to row data in the SQLite database and its associated journal mode file that is no longer active. Either through row deletion or row modification.

Recovered Data – Table information that is not part of the current state of the database and all associated journal mode file(s) as of the latest commit.

Rollback Journal – This file is associated with an SQLite database that holds information used to restore the database to its previous state during a transaction while in journal mode. The setting of the journal_mode **PRAGMA** can be used to determine if a rollback journal is being used. This file resides in the same folder as the database and has the string "-journal" appended to its filename.

SQLite – SQLite is an embedded SQL relational database engine that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

Table – A data structure that organizes information into rows and columns. It can be used to store and display data in a structured format.

Vacuum – A command that rebuilds the database file, repacking it into a minimal amount of disk space. When a vacuum occurs, data may be overwritten or deleted.

WAL Timelining – See Journal Sequencing.

Write-Ahead Log (WAL) – A file that records SQLite transactions that have been committed, but not yet applied to the database. This file is in the same directory as the database with the string "-wal" appended to its filename. As of version 3.7.0 (dated 7/21/2010) this file type is the most commonly used method when SQLite journaling mode is enabled.

5 Requirements

This section lists the SQLite Data Recovery Tool requirements. There are requirements for core features that all tools must meet and requirements for optional features as well. The requirements for optional features only apply if the tool supports the feature.

5.1 Required Features

The following requirements shall be met by all tools:

SFT-CR-01. The tool shall not modify the files being analyzed.
SFT-CR-02. The tool shall report the database configuration parameters pertinent to data recovery.
SFT-CR-03. The tool shall report the schema structure of the database tables.
SFT-CR-04. The tool shall report the data content of all recovered rows of any table in the database.
SFT-CR-05. The tool shall report the source for all recovered data elements.

5.2 Optional Features

SFT-RO-01. The tool shall report additional schema data within the database for all recovered data.
SFT-RO-02. The tool shall report the metadata for all recovered rows.
SFT-RO-03. The tool shall report the detailed metadata for all recovered data elements.
SFT-RO-04. The tool shall be able to perform journal sequencing (or wal timelining) of the associated -wal journal mode file.

6 Test Assertions

Here is a set of test assertions based on the requirements:

6.1 Core Assertions

Assertion	Req
SFT-CA-01. The MD5 (or SHA-1) hash value of the database and associated journal mode file (e.g., -journal, -wal) shall not be altered between when analysis begins and analysis is complete.	CR-01
SFT-CA-02. The associated journal mode file (e.g., -journal, -wal) shall not be deleted after analysis is complete.	CR-01
SFT-CA-03. The tool shall interpret the SQLite Page Size (in bytes).	CR-02
SFT-CA-04. The tool shall report the SQLite Journal Mode (write version)	CR-02
SFT-CA-05. The tool shall report the SQLite Journal Mode (read version)	CR-02
SFT-CA-06. The tool shall report the number of pages in the database	CR-02
SFT-CA-07. The tool shall report the SQLite database text encoding.	CR-02
SFT-CA-08. The tool shall report all table names for each table within the database.	CR-03
SFT-CA-09. The tool shall report all column names for each table in the database.	CR-03
SFT-CA-10. The tool shall report the number of rows for each table in the database.	CR-03

SFT-CA-11. The tool shall report on all recoverable rows that are contained within the database.	CR-04
SFT-CA-12. The tool shall report on all recoverable rows that are contained within the associated journal mode file (e.g., -journal, -wal).	CR-04
SFT-CA-13. The tool shall report the source file name for each recovered data element.	CR-05

6.2 Optional Test Assertions

Test Assertions for Optional Features	Req
SFT-AO-01. The tool shall report all CREATE TABLE statements for each table in the database.	RO-01
SFT-AO-02. The tool shall report the data type for each column within each table in the database.	RO-01
SFT-AO-03. The tool shall report which column is the primary key for each table in the database.	RO-01
SFT-AO-04. The tool shall report if the row was recovered because of a deletion or an update within the database.	RO-02
SFT-AO-05. The tool shall report if the row was recovered because of a deletion or an update in the associated journal mode file (e.g., -journal, -wal).	RO-02
SFT-AO-06. The tool shall report the file offset for each recovered data element presented.	RO-03
SFT-AO-07. The tool shall report the table name for each recovered data element presented.	RO-03
SFT-AO-08. The tool shall be able to present the sequence of transactions in the associated -wal file.	RO-04

7 SQLite Forensics Tool Test Cases

Core Test Cases	Assert
SFT-01. SQLite header parsing. This test case verifies that the tool provides the following (5) attributes as contained in the SQLite header: ▪ Page Size ▪ Journal Mode Information ▪ Number of Pages ▪ Text Encoding (i.e., UTF-8, UTF-16 Little Endian, and UTF-16 Big Endian) *Note: Header results will remain consistent when journal_mode is set to any of the following: DELETE, MEMORY, OFF, PERSIST or TRUNCATE.* *Test Actions:* SFT-01-UTF8-WAL – Create SQLite file with specified parameters. 1. SQLITE3 SFT-01-UTF8-WAL.sqlite 2. PRAGMA journal_mode = WAL 3. PRAGMA encoding = 'UTF-8'; 4. PRAGMA page_size = 4096; 5. Create Table 6. Create 100 Rows of Data within Table 7. .quit 8. Read header and validate: Page Size, Journal Mode, Number of Pages, and Encoding. 9. If files have changed, investigate each set of test actions to determine where the change occurred. *Test Actions:* SFT-01-UTF16BE-PERSIST – Create SQLite file with specified parameters. 1. SQLITE3 SFT-01-16BE-PERSIST.sqlite 2. PRAGMA journal_mode = PERSIST 3. PRAGMA encoding = 'UTF-16be'; 4. PRAGMA page_size = 1024; 5. Create Table 6. Create 100 Rows of Data within Table 7. .quit 8. Read header and validate: Page Size, Journal Mode, Number of Pages, and Encoding. 9. If files have changed, investigate each set of test actions to determine where the change occurred. *Test Actions:* SFT-01-UTF16LE-OFF – Create SQLite file with specified parameters. 1. SQLITE3 SFT-01-16LE-OFF.sqlite 2. PRAGMA journal_mode = OFF	CA-01 CA-03 CA-04 CA-05 CA-06 CA-07

3. PRAGMA encoding = 'UTF-16le'; 4. PRAGMA page_size = 8192; 5. Create Table 6. Create 100 Rows of Data within Table 7. .quit 8. Read header and validate: Page Size, Journal Mode, Number of Pages, and Encoding. 9. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Data reported matches data contained within header as specified in each test action.	
SFT-02. SQLite Schema Reporting. This test case verifies that the tool provides a listing of all: 1. Tables 2. Column names for each table 3. Row information for each table *Test Actions:* SFT-02 – Schema Reporting 1. Using SFT-01-UTF8-WAL.sqlite, SFT-01-UTF16BE-PERSIST and SFT-01-UTF16LE-OFF 2. Read table data and validate: Table Names, Column Names, and number of rows. 3. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Data reported matches data contained within the database as specified in each test action.	CA-01 CA-08 CA-09 CA-10
SFT-03. SQLite Recoverable Rows This test case verifies that the tool reports the file name (e.g., source) and recovered information for all recoverable rows (e.g., deleted and updated): ▪ SQLite database file ▪ SQLite database journal mode file (e.g., -journal, -wal) *Test Actions:* SFT-03-PERSIST – Create SQLite file with an associated -journal file. 1. SQLITE3 SFT-03-PERSIST.sqlite 2. PRAGMA journal_mode = PERSIST 3. Create Table 4. Create ~2,000 rows of data within table 5. Delete 100 rows (randomly) 6. Modify 100 rows (randomly) 7. Perform SQLite database recovery	CA-01 CA-02 CA-11 CA-12

8. Validate reporting of 100 deleted rows and 100 modified rows. 9. If files have changed, investigate each set of test actions to determine where the change occurred. *Test Actions:* SFT-03-WAL – Create SQLite file with an associated -wal file. 1. SQLITE3 SFT-03-WAL.sqlite 2. PRAGMA journal_mode = WAL 3. Create Table 4. Create ~2,000 rows of data within table 5. Modify 100 rows (randomly) 6. BEGIN TRANSACTION; Delete 100 rows (randomly) END TRANSACTION; 7. Stop the SQLite process (i.e., close command/console window or CTRL-C) 8. Perform SQLite database recovery 9. Validate reporting of 100 deleted rows and 100 modified rows. 10. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Data reports deleted and modified row data as specified in each test action.	
SFT-04. SQLite Data Element metadata This test case verifies that the tool reports the file name (e.g., source) for all recovered data elements: ▪ SQLite database file ▪ SQLite database journal mode file (e.g., -journal, -wal) *Test Actions:* SFT-04 – PERSIST 1. Using SFT-03-PERSIST.sqlite and SQLITE3 SFT-03-PERSIST.sqlite-journal 2. Perform SQLite database recovery 3. Verify that tool reports the file name (e.g., source) where each recoverable data element is located. 4. If files have changed, investigate each set of test actions to determine where the change occurred. *Test Actions:* SFT-04 – WAL 1. Using SFT-03-WAL.sqlite and SQLITE3 SFT-03-WAL.sqlite-wal 2. Perform SQLite database recovery 3. Verify that tool reports the file name (e.g., source) where each recoverable data element is located. 4. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Data reports deleted and modified row data as specified in each test action.	CA-01 CA-02 CA-13

Optional Test Cases	
SFT-05. SQLite schema data reporting This test case verifies that the tool reports the SQLite metadata for all create table statements and type (e.g., Storage Class, datatype, or affinity) for each column, and that it identifies which column is the primary key for each table in the database. *Test Actions:* SFT-05 – Schema Reporting 1. SQLITE3 SFT-05.sqlite 2. Create Table with at least (6) columns: Primary Key, Int, Float, Text, Blob, Boolean 3. Create 100 Rows of Data within Table 4. Read table data and report all create table statements, associated column types, and the primary key for each table. 5. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Data reported matches data contained within the database as specified in each test action.	CA-01 AO-01 AO-02 AO-03
SFT-06. Recovered row metadata This test case verifies that the tool reports the recovered row because of either a deletion or an update within the database file, or the associated journal mode file (e.g., -journal, -wal). *Test Actions:* SFT-06 – PERSIST 1. Using SFT-03-PERSIST.sqlite and SQLITE3 SFT-03-PERSIST.sqlite-journal 2. Perform SQLite database recovery 3. Tool reports the file name (e.g., source) and if the row was the result of an update or a deletion. 4. If files have changed, investigate each set of test actions to determine where the change occurred. *Test Actions:* SFT-06 – WAL 1. Using SFT-03-WAL.sqlite and SQLITE3 SFT-03-WAL.sqlite-wal 2. Perform SQLite database recovery 3. Tool reports the file name (e.g., source) and if the row was the result of an update or a deletion. 4. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Recovered information matches the actions from each test case.	CA-01 CA-02 AO-04 AO-05
SFT-07. SQLite recovered data information	CA-01 CA-02

This test case verifies that the tool reports the following metadata for all recoverable data elements: 1. Offset within the file 2. Identify the table name associated with the row *Test Actions:* SFT-07 – PERSIST 1. Using SFT-03-PERSIST.sqlite and SQLITE3 SFT-03-PERSIST.sqlite-journal 2. Perform SQLite database recovery 3. Tool reports the offset and length of the data within the payload for each recovered cell. 4. Tool reports the table name for each row of recovered data. 5. If files have changed, investigate each set of test actions to determine where the change occurred. *Test Actions:* SFT-07 – WAL 1. Using SFT-03-WAL.sqlite and SQLITE3 SFT-03-WAL.sqlite-wal 2. Perform SQLite database recovery 3. Tool reports the offset and length of the data within the payload for each recovered cell. 4. Tool reports the table name for each row of recovered data. 5. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Recovered metadata matches the actions from each test case.	AO-06 AO-07
SFT-08. Journal sequencing/wal timelining This test case verifies that the tool reports the sequence of transactions in the associated -wal file. *Test Actions:* SFT-08 – WAL 1. Using SFT-03-WAL.sqlite and SQLITE3 SFT-03-WAL.sqlite-wal 2. Perform SQLite database recovery 3. Order recovered transactions within the -wal journal file. 4. If files have changed, investigate each set of test actions to determine where the change occurred. *Conformance Indicator:* Recovered data is sequenced matching the chronological actions executed during testing (SFT-08-WAL).	CA-01 CA-02 AO-08

192