SPC TRACE FILE FORMAT SPECIFICATION

Revision 1.0.1

Storage Performance Council (SPC)
1060 El Camino Real, Suite "F"
Redwood City, CA 94062-1645
Phone (650) 556-9384
Fax (650) 556-9385
www.storageperformance.org
spcadmin@storageperformance.org

SPC Membership

(as of July 2002)

DOCUMENT HISTORY

Effective Date	**Version**	**Description**
28-Feb-99	1.0.0	Initial draft
17-Jul-02	1.01	Allowed white space characters in field contents Miscellaneous formatting changes

0 INTRODUCTION

0.1 BACKGROUND

In order to create a list of the I/O parameters associated with a specific benchmark, the characteristics of the representative workloads must be analyzed and well understood. To accomplish this, the individual I/O commands issued by the host processor(s) must be collected and analyzed. Since many different analysis programs may be used, and since these programs will in all probability be run “after the fact”, a goal of the SPC is to collect I/O trace data from various systems for later analysis.

With the variety of hardware platforms, operating systems, and data collection techniques that exist, analysis of the collected trace data would be next to impossible without a well-defined common file format for the traces.

The goal of this document is to specify both a necessary and sufficient amount of information that must be contained in a trace file to allow reproduction of the essence of the original workload. This goal is constrained by the realization that not all relevant data may be collected from all operating systems, and that finite resources exist for collection, storage, and analysis of the trace data.

It is a non-goal of this document to specify the trace file capture techniques. It is an additional non-goal of this document to specify the trace file analysis techniques.

1 <u>TRACE FILE FORMAT</u>

1.1 OVERVIEW

The trace file is composed of variable length ACSII records, rather than binary data. Although this format is somewhat wasteful of storage space and places higher CPU demands on analysis programs, it offers many advantages from a legibility and portability standpoint.

Each record in the trace file represents one I/O command, and consists of several fields. The individual fields are separated by a comma (hex 2C), with the trace record being terminated by a newline character (\n). There is no special end-of-file delimiter; that function being left to the individual operating systems.

1.2 TRACE RECORD FORMAT

Each trace record represents a single I/O command. This record is divided into five required fields, with optional fields added at the end. White space characters, such as tabs and spaces, are optional, and may be used for visual clarity if desired. If white space characters are present in any of the five required fields, they may only exist between the comma character and the beginning of the next field. White space characters may exist in any position in the optional fields.

1.2.1 Field 1: Application specific unit (ASU)

The ASU is a positive integer representing the application specific unit (see the SPC Global Parameter specification document). This is a zero based, monotonically increasing number. The first record in the trace file need not have the ASU equal to zero, however unit zero must exist within the trace file. If there are a total of *n* units described in the complete trace file, then the trace file must contain at least one record for each of units *0* through *n-1*.

1.2.2 Field 2: Logical block address (LBA)

The LBA field is a positive integer that describes the ASU block offset of the data transfer for this record, where the size of a block is contained in the description of the trace file. This offset is zero based, and may range from 0 to *n-1*, where *n* is the capacity in blocks of the ASU. There is no upper limit on this field, other than the restriction that sum of the address and size fields must be less than or equal to the capacity of the ASU.

1.2.3 Field 3: Size

The size field is a positive integer that describes the number of bytes transferred for this record. A value of zero is legal, the result of which is I/O subsystem dependent. Although the majority of records are anticipated to be modulo 512, this constraint is not required. There is no upper limit on this field, other than the restriction that sum of the address and size fields must be less than or equal to the capacity of the ASU.

1.2.4 Field 4: Opcode

The opcode field is a single, case insensitive character that defines the direction of the transfer. There are two possible values for this field:

1. "R" (or "r") indicates a read operation. This implies data transfer *from* the ASU *to* the host computer.
2. "W" (or "w") indicates a write operation. This implies data transfer *to* the ASU *from* the host computer

1.2.5 Field 5: Timestamp

The timestamp field is a positive real number representing the offset in seconds for this I/O from the start of the trace. The format of this field is "s.d", where "s" represents the integer portion, and "d" represents the fractional portion of the timestamp. Both the integer and fractional parts of the field must be present. The value of this field must be greater than or equal to all preceding records, and less than or equal to all succeeding records. The first record need not have a value of "0.0".

1.2.6 Field 6 (through n): Optional field(s)

Since there will undoubtedly be large amounts of additional information that is present in the raw captured data, and since this information may well be worthy of detailed analysis, the provision for optional fields exists in the SPC trace files. The content of any optional field is not defined by this document, since the meaning and content may change from one trace to another. Moreover, the purpose of this document is to allow the creation of a standard suite of analysis programs, and the presence (and implied absence) of optional fields negates the concept of standardization.

Nonetheless, optional fields may be added to the each record in the trace file by the simple expedient of inserting a field separator (comma), followed by the field. If there is more than one field to be added, then each field must be separated from the preceding field by the comma field separator. Optional fields need not be present on every record of the trace file. Moreover, the number of optional fields may change within a single trace file.

Standard analysis programs should be written so that the presence, absence, and number of optional fields will not impact their function.

1.3 TRACE FILE EXAMPLE

The following is an example of the first few record of a trace file:

```
0,20941264,8192,W,0.551706,Alpha/NT
0,20939840,8192,W,0.554041
0,20939808,8192,W,0.556202
1,3436288,15872,W,1.250720,0x123,5.99,test
1,3435888,512,W,1.609859
1,3435889,512,W,1.634761
0,7695360,4096,R,2.346628
1,10274472,4096,R,2.436645
2,30862016,4096,W,2 448003
2,30845544,4096,W,2.449733
1,10356592,4096,W,2.449733
```

The first record occurs 0.551706 seconds after the start of the trace file, and is for ASU 0. The operation is to write 8,192 bytes of data to block 20,941,264. Note the optional field that has been added to the end of the first record.

The second I/O occurs 0.002335 seconds later (0.554041 seconds into the trace), and is an 8,192 byte write to block 20,939,840 of ASU 0.

The first I/O to ASU 1 occurs at the fourth record (1.25072 seconds into the trace), and is a write of 15,872 bytes to block 3,436,288. Note that this record has three optional fields, none of which is defined in this document.

Finally, note that the last two records in this example have an identical time stamp of 2.449733. This may be due to the commands in question being issued at exactly the same time from different host computers, or may be simply a case of poor resolution in the data capture routines.