To: INCITS Technical Committee T10

From: Kevin Butt, IBM

Date: November 9, 2004 10:39 am

Document: T10/04-312r0

Subject: SSC-3: Definition of WORM

1. Revisions

2. Introduction

I received an action item to providea definition for WORM. This document is an initial discussion point to begin that effort.

3. Discussion

3.1 WORM Behavior

Some streaming devices support specially formatted WORM media which enables tamper-resistant WORM write and append methods. An interface and control mechanisms which allow an application or system to manage this as needed. The control and status mechanisms for this can be added to mode parameters. A new medium identifier is also needed. Generally, the WORM function may be used without application awareness, however if the write/append behavior is violated (an illegal overwrite is attempted) then a write perm will occur.

3.2 Common Constructs Used on Media

Many common label, header, trailer, and end of volume constructs may be used by application clients to help delineate user data. In the following lists:

- BOP indicates that this sequence must start at the logical beginning of a partition (logical block number 0),
- <FM> indicates a filemark,
- <?> indicates any character,
- <?..?> indicates one or more records of the same type with any character in any order,
- <#> indicates a sequential numeric character,
- <n..#> indicates one or more records of the same type with sequential numerical character(s) starting with n,

- Items listed in quotes indicate that a record must start with that string, in either ASCII or EBCDIC (note that all records in the same construct must be either ASCII or EBCDIC, intermix will terminate the construct). While most common label constructs consist of records of 80 bytes (or more), this is not enforced by the identification process. A minimum record size of 4 bytes is required for identification. There is no further limit for maximum record size for identification, beyond the maximum supported device logical block length.
- In the case of "VOL1" at least six more characters indicating the VOLSER in the same encoding are expected (VOL1 records smaller than 10 bytes are not recognized).
- The identification process treats synchronizes (Write Filemarks 0 or implicit) as transparent.
- Repositions are allowed and handled during construct recognition, however, constructs are only recognized within the current mount. If media is unloaded with a partial construct or construct not followed by user data, then the next series of writes must overwrite the existing construct with a new complete construct, otherwise the partial construct on media is treated as user data and subject to normal overwrite rules. In this case, the leading filemark in the general forms below is not required for construct identification.

Some of these constructs which WORM devices may recognize follow:

3.2.1 LABELS:

The general form (items listed in brackets [] are optional) is:

BOP<"VOL1<volser>”>[<“VOL<2..#>”>][<“UVL<1..#>”>][<"HDR<1..#>">[<"UHL<?..?>">]]<FM>[<FM>]

The following special form is also recognized:

BOP<"VOL1”<volser>>

Some examples of the general form include (but are not limited to):

BOP<"VOL1”<volser>><FM>

BOP<"VOL1”<volser>><FM><FM>

BOP<"VOL1”<volser>><"VOL2"><"VOL3"><"HDR1"><FM>

BOP<"VOL1”<volser>><"HDR1"><"HDR2"><FM><FM>

BOP<"VOL1”<volser>><"HDR1"><"HDR2"><"HDR3"><"UHL6"><"UHLA"><FM>

3.2.2 HEADERS:

The general form (items listed in brackets [] are optional) is:

<FM><"HDR<1..#>”>[<“UHL<?..?>”>]<FM>[<FM>]

Some examples include (but are not limited to):

<FM><"HDR1"><FM>

<FM><"HDR1"><"HDR2"><FM><FM>

<FM><"HDR1"><"UHLC"><"UHL4"><FM><FM>

3.2.3 TRAILERS:

The general form (items listed in brackets [] are optional) is:

<FM>[<"EOF<1..#>">[<"UTL<?..?>][<"OIB<?..?>">]<FM>][<FM>]

Some examples include (but are not limited to):

<FM>

<FM><FM>

<FM><"EOF1"><FM><FM>

<FM><"EOF1"><"EOF2"><FM>

<FM><"EOF1"><"EOF2"><"EOF3"><"UTL2"><"UTL1"><FM>

<FM><"EOF1"><"OIBD"><"OIB6"><"OIBZ"><FM>

<FM><"EOF1"><"EOF2"><"UTL_"><"UTL5"><"OIB-"><"OIB$"><FM><FM>

3.2.4 END OF VOLUME:

The following END OF VOLUME constructs would normally be written shortly after early warning indicators are raised and will never be overwritten.

The general form (items listed in brackets [] are optional) is:

<FM><"EOV<1..#>">[<"UTL<?..?>][<"OIB<?..?>">]<FM>[<FM>]

Some examples include (but are not limited to):

<FM><"EOV1"><FM>

<FM><"EOV1"><"EOV2"><FM><FM>

<FM><"EOV1"><"EOV2"><"EOV3"><"UTLX"><FM><FM>

<FM><"EOV1"><"OIBK"><"OIB3"><"OIB+"><FM><FM>

<FM><"EOV1"><"EOV2"><"UTLL"><"UTLL"><"OIBK"><"OIB3"><"OIB+"><FM><FM>

3.2.5 Construct example:

BOP<"VOL1"<volser of 123456>><"HDR<1..2>"><"UHL<1..3>"><FM><FM>

which can be expanded to:

BOP<"VOL1123456"><"HDR1"><"HDR2"><"UHL1"><"UHL2"><"UHL3"><FM><FM>

and appears on medium as a sequence where:

BOP indicates this sequence must start at LB 0

LB 0 is a record of 10 bytes or more starting with "VOL1123456"

LB 1 is a record of 4 bytes or more starting with "HDR1"

LB 2 is a record of 4 bytes or more starting with "HDR2"

LB 3 is a record of 4 bytes or more starting with "UHL1"

LB 4 is a record of 4 bytes or more starting with "UHL2"

LB 5 is a record of 4 bytes or more starting with "UHL3"

LB 6 is a filemark

LB 7 is a filemark

3.3 WORM Write/Append General Behavior

To accomplish data permanence and maximum application compatibility, methods to identify common label and trailer constructs need to be implemented, and a new ASC/ASCQ (Data Protect, WORM MEDIUM – OVERWRITE ATTEMPTED) has been added to indicate that writing is not allowed at the current location. There are several general rules surrounding WORM writing. Mode specific details can be found in following section(s).

- Appending is always allowed at the EOD location for non-locked medium.
- Writing is never allowed at any location prior to the start of an identified non-buffered label or trailer construct.
- Writing may not be allowed after any construct which is followed by data which has been explicitly synchronized. This includes: Write Filemarks 0, Write Filemarks non-immediate, Unload. (depending on WORM mode).
- Writing may not be allowed after any construct which is followed by data which has been implicitly synchronized by host or operator action. This includes: Space, Locate, Rewind, Load/Unload, etc. (depending on WORM mode).
- Writing may not be allowed after any construct which is followed by data which has been implicitly synchronized by the drive. This includes: Buffer flush, buffer full conditions, etc. (depending on WORM mode).
- Writing may be allowed up to and including an overwrite of all or part of the last construct when a construct is either incomplete, or is not followed by additional data or filemarks (depending on WORM mode).
- Writing may be allowed up to and including an overwrite of the last construct if a permanent write error occurs prior to or during the first host invoked synchronize of the data following that construct (depending on WORM mode).
- Buffered constructs (those where all filemarks are written immediate, and no other synchronizing events occur) may have a longer overwrite horizon in the permanent error cases. This may extend to a location earlier than constructs which were fully written to media. However, on any normal host invoked synchronizing event, the appropriate overwrite location will be locked (depending on WORM mode).

- Writing may not be allowed at points other than the last complete logical block when a tape is loaded and it's ending disposition can not be definitively determined. This can occur in loss of power and other severe error conditions.

3.4 WORM mode X'2' Write/Append Behavior

Refer to information in the general section to identify some of the possible operation possibilities and for overall behaviors. Specific details for this mode follow:

- Writing is not allowed prior to any non-construct record which has been completely written to medium.
- Overwriting of all or part of the most recent construct (label, header or trailer) is allowed, provided that the constuct is partially or fully recognized and is not followed by unrecognized records (i.e. user data).