To: Distribution

From: Janice B. Phillipps

Date: 11/07/75

Subject: Revision of MTB-209; tape_in, tape_out

A few months ago, MTB-209 was circulated proposing enhancements to a module which would facilitate file transfer between the storage system and magnetic tape. The objective behind these proposed enhancements was the following: install a command supporting file transfer between a file on any kind of 9-track tape that can be written on Multics, and a file of any kind in the storage system file, indexed sequential files excepted.

This objective was to be met in the following manner:

1) Revise an existing uninstalled program -- tape_in, tape_out -- to meet installation in standard service system library as a between-storage-system-and-tape file transfer command.

2) Revise the existing tape_in, tape_out from that which supports file transfer only between unlabeled 9-track tape files and unstructured files in the storage system to that which supports file transfer between any ANSI, IBM, MST, or non-standard tape could be read or written in the file transfer operations.

3) Have tape_in, tape_out support all of the features offered by the four system tape IO modules passing on the flexibility and versatility of the existing tape IO Modules. of the existing tape IO modules, including tape label handling, and that it become a general purpose command in the Multics tape facility.

After circulating MTB-209, it became clear from the comments that we were not sure just whose needs the tape_in, tape_out commands should be designed to serve: how powerful or how limited should it be? In subsequent discussion, which gave birth to the design to the copy_file command, it was decided to aim usage at the unsophisticated user who might be transfering files from OS or GCOS to Multics using standard format tapes. In

itemizing the minimum options a user would need to write useful tapes, all but two or three of the tape IO module attach options were deamed necessary.

This memo will trying to restate the design of tape_in, tape_out while trying to simplify it.

Item 1) do as stated above.

Item 2) will be amended as follows.

Revise the existing tape_in, tape_out from that which supports file transfer only between unlabeled 9-track tape files and unstructured files in the storage system to that which supports file transfer between any ANSI standard format, IBM standard format, IBM format unlabeled tapes and any unstructured or sequential format storage system files.

Item 3) will be amended as follows.

tape_in, tape_out will support most of the features offered by tape_ansi_ and tape_ibm_ IO modules and will support most of the features offered by vfile_ (i.e. not indexed sequential format files). The commands will be of limited general usefulness.

Again, these commands, tape_in, tape_out, are designed for the unsophisticated user. For the more sophisticated user, the copy_file command will be available.

Name: tape_in, tin
tape_out, tout

These commands allow the user to transfer files between the storage system and magnetic tape. tape_in reads from tape to the storage system; tape_out writes from the storage system to tape. To accomplish a file transfer, the tape_in and tape_out commands access either the tape_ansi_ or the tape_ibm_ IO module for the tape interface, and the vfile_ IO module for the storage system interface. Unstructured format storage system files (for stream I/O) and sequential format storage system files (for record I/O) may be specified; 9-track ANSI standard labeled tapes, 9-track IBM standard labeled tapes, and any 9-track unlabeled tape which is structured according to OS standard, may be be read or written.

Usage: tape_in pathname -control_arg1-
tape_out pathname -control_arg1- -control_arg2-

pathname — is the path name of the control file which governs the file transfer. If pathname does not end with the suffix .tci, .tci will be supplied.

control_args:

1) severityi, -svi — causes the tape_in, tape_out compiler's error messages with severity less than i (where i is 0, 1, 2, 3, or 4) not to be written into the "error_output" IO stream. The default value for i is 0. See APPENDIX A, Error Diagnostics, for further information on error reporting.

2) -force, -fc — specifies that the expiration date of a tape file to be overwritten is to be ignored. This option extends unconditional permission to overwrite a tape file, regardless of the file's "unexpired" status. This unconditional

permission supresses any query made by the IO module to inquire about tape file's expiration date. The -force option is only meaningful when used with the tape_out command. If the -force option is used with the tape_in command, an error is indicated.

THE TCL PROGRAM

The control file which governs the file transfer is actually a program, written by the user, in the tape control language (tcl). The contents of this control file specifies the name of the IO module to do the actual file transfer(s), the structural attributes of the file(s) in the storage system to be copied or created, and the structural attributes of the file(s) on the tape volume(s) to be copied or created. When the user issues the tape_in or tape_out command, the control file named in the command line (pathname) is compiled and if the compilation is successful, the generated object code is executed to accomplish the desired file transfer(s). The same control file may be used with the tape_in command, to read a file from tape into the storage system, as with the tape_out command, to write a file from the storage system onto tape.

Example:

A Simple TCL Control File

simple.tcl

```
Volume: 012345;
File: File_1;
Path: >udd>Project>User_dir>demo;
End;
```

This tcl program is a sample of a simple control file that can be written to accomplish file transfer. The simplest control file must have a Volume statement, a File statement, a Path

statement and an End statement; other statements are optional. This sample control file relies on the tcl defaults listed below under <volume-group> Defaults. The file transfers possible with this sample control file are either writing tape file "File_1" from storage system file "demo", or writing storage system file "demo" from tape file "File_1".

A tcl program consists of one or more <volume-group>s. A <volume-group> is a series of statements which specifies the file transfer(s) to be performed between the storage system and a particular tape volume. A <volume-group> must begin with a Volume statement, contain one or more <file-group>s, and terminate with an End statement. In addition, a <volume-group> may optionally contain one or more <global-statement>s.

tcl statements must begin with a keyword, may or may not take a keyword argument, and must end with a semi-colon. <global-statement> keywords and the "Volume", the "File", "Path", and the "End" statement keywords must begin with an uppercase letter. <local-statement> keywords must begin with a lower case letter.

The Volume Statement

Volume! <volid>;

The control file Volume statement specifies the tape volume to be used in file transfer. This statement causes a tape volume whose volume serial number is <volid> to be mounted on a 9-track drive. <volid> must consist of from 1 to 6 ASCII characters. If <volid> is less than 6 characters, it will be padded on the right with blanks to a length of six. If <volid> contains any of the following characters, <volid> must be enclosed in quote characters (")!

1) any ASCII control character - (tcl ignored breaks)
2) ! ; , or blank - (tcl breaks)
3) the sequence /* or */ - (tcl comment delimiters)
4) If <volid> itself contains a quote character, the quote must be doubled and the entire <volid> string enclosed in quotes.

Examples

```
Volume: 001234;                    mounts volume 001234
Volume: XJ56;                      mounts volume XJ56
Volume: "as"";56";                 mounts volume as";56
Volume:  -00451;                   mounts volume -00451
```

<volume-group> Defaults

Associated with a <volume-group> are a set of default characteristics. In the absence of overriding <global-statement>s or <local-statement>s, these defaults will apply to all <file-group>s within the <volume-group>. If no IO Module is specified in the control file, tape_ansi_ and the associated tape_ansi_ <volume-group> defaults will preside. If, however, the IO Module is specified in the control file, the <volume-group> defaults for that IO Module will preside until overridden.

1) tape IO Module used: tape_ansi_
2) density: 800 bpi
3) file expiration: immediate
4) storage system file format: unstructured
5) mode: 9mode, ascii character code
6) tape file record format: SB
7) physical block length: 2048 characters (maximum)
8) logical record length: 1044480 characters (maximum)

1) tape IO Module used: tape_ibm_
2) density: 1600 bpi
3) file expiration: immediate
4) storage system file format: unstructured
5) mode: ebcdic;
6) tape file record format: VBS
7) physical block length: 8192 characters (maximum)
8) logical record length: 1044480 characters (maximum)

<global-statement>

A <global-statement> changes a <volume-group> default. The IO_Module, and Density <global-statement>s may appear only once in a <volume-group>. The Storage, Mode, Format, Block and Record <global-statement>s may appear any number of times within a <volume-group>.

IO_Module: <io_module>;

The IO_Module <global-statement> specifies the tape IO Module to be used in the file transfer. <io_module> must be "tape_ansi_" for reading or writing ANSI tapes, and must be "tape_ibm_" for reading or writing IBM standard tapes or non-labeled tapes. This <global-statement> may appear only once within a <volume-group> or an error is indicated.

Density: <den>;

The Density <global-statement> indicates the density in which the volume is (to be) recorded. <den> must be either "800" or "1600" (bpi). Warning: the use of 1600 bpi for ANSI interchange tapes is non-standard. It will not be standard for interchange. This <global-statement> may appear only once within a <volume-group> or an error is indicated.

Storage: <structure>;

The Storage <global-statement> states the internal (logical) structure of the storage system file(s) to be specified by subsequent <file-group>s. The unstructured file is referenced as a series of 9-bit bytes, commonly called lines; the sequential file is referenced as a sequence of records, each record being a string of 9-bit bytes. <structure> must be either "unstructured" or "sequential". If the Storage <global-statement> is omitted from a control file <volume-group>, the assumed storage system file format will be "unstructured". If, then a sequential file is referenced within that <volume-group>, the results are undefined and an error is indicated.

Mode: <mode>;

The Mode <global-statement> specifies the tape mode and character code to be used with subsequent <file-group>s. <mode> may be either "ascii"" or "ebcdic" for IBM tapes (using tape_ibm_ IO module) and may be either "ascii", "ebcdic", or "binary" for ANSI tapes (using tape_ansi_ IO module). Warning: the use of ebcdic mode or binary mode is not standard for ANSI tapes.

Note: Refer to the relevant IO Module write_up for a description of the interaction between a given list of format, block and record specification.

Format: <form>;

The Format <global-statement> specifies the tape record format to be used with subsequent <file-group>s. <form> must be either "U", "F", "FB", "D", "DB", "S", of "SB" for ANSI tapes (using tape_ansi_ IO module) and "F", "FB", "U"",V", "VB", "VS", "VB", "or "VBS" for IBM tapes (using tape_ibm_ IO module).

Block: <blklen>;

The Block <global-statement> specifies the tape file (maximum) physical block length, in characters, to be used with subsequent <file-group>s. <blklen> must be a decimal integer, such that 18 ≤ <blklen> ≤ 8192. Warning: <blklen> greater that 2048 does not comply with the ANSI standard for tapes.

Record: <reclen>;

The Record <global-statement> specifies the tape file (maximum) logical record length, in characters, to be used with subsequent <file-group>s. <reclen> must be a decimal integer, such that 1 ≤ <reclen> ≤ 1044480.

The End Statement

An End statement must appear as the last statement of a <volume-group>.

End;

<file-group>

Every <volume-group> must contain one or more <file-group>s. Each <file-group> defines one tape to storage system file transfer. A <file-group> must begin with a File statement, and contain a Path statement. In addition, it may contain one or more <local-statement>s. A <file-group> is terminated by a <global-statement>, an End statement, or a File statement (new <file-group>).

The File Statement

File: <fileid>;

The File statement specifies that a new tape file is to be read or written. The tape file is identified by <fileid>. <fileid>, the tape file identifier, is from 1 to 17 characters inclusive, for ANSI tapes, and must be a valid DSNAME for IBM tapes. <fileid> for unlabeled tapes, which are discussed below, is the character "*". The File statement marks the beginning of any local attributes for a given tape file transfer.

The Path Statement

Path: <pathname>;

Every <file-group> must contain one Path statement. The Path statement specifies the path name of the storage system file to be read or written. <pathname> may be either a relative or absolute path name.

<local-statement>

A <file-group> may contain one or more <local-statement>s. A <local-statement> overrides the <volume-group> defaults in effect at the time a <file-group> is evaluated. A <local-statement> has no effect outside of the <file-group> in which it occurred, and may appear anywhere within the <file-group>.

The storage, mode, format, block and record <local-statement>s operate exactly as do their <global-statement> counterparts, except that they affect only the <file-group> in which they are contained.

The Number Statement

number: <number>;

The number <local-statement> further identifies the tape file to be used in file transfer. <number> is the file sequence number; it specifies the position of the tape file within the file set. <number> must be an integer between 1 and 9999 inclusive. If the control file is to be used with the tape_in command, <number> specified in a number statement, must refer to a file within the file set and both the <fileid> specified in the File statement and the <number> specified in the number statement must refer to the same tape file; otherwise an error is indicated. When the control file is to be used with the tape_out command, <number> specifies the file to be created or replaced. When using the tape_out command, the number statement (where <number = 1>) must be explicitly included in the <file-group> to create the first file on an entirely new file set. The tape_ansi_ and tape_ibm_ IO modules initialize tape volumes with a dummy file. If <number = 1> is not specified, in such a <file-group>, the first file that is written will be appended to the file set containing the initial dummy file making with the tape file sequence number 2 rather than 1.

Control File Execution

When the tcl control file is being executed in response to the tape_in command, the volume named in each <volume-group> of the control file is mounted in turn without a write ring. If any file output options appear in a control file being executed in response to the tape_in command, these statements will be ignored. When the control file is being executed in response to the tape_out command, the volume named in each <volume-group> of the control file is mounted in turn with a write ring.

Control File Comments

Comments may be inserted anywhere within the tcl program by surrounding the comment text with the comment delimiters. /* is the delimiter which begins a comment, and */ is the delimiter which terminates a comment.

Notes

File transfer is performed as described below. See APPENDIX A for the value of nelem associated with each tape record format.

1) tape_in: one logical record is read from the tape file, and as many characters as were read are written into the storage system file either as a line with new-line (NL) character appended in an unstructured format file or as one logical record in a sequential format file.

2) tape_out: an attempt is made to read either a line or a record from the storage system file to be written onto tape. For unstructured format storage system files, a line read is a line from the file up to and including the first new-line character (NL) encountered; for sequential format storage system files, a record read is one logical record of the file. The characters read from the storage system are then written on the tape as one logical record of the tape file.

Under certain circumstances, tape records being written must be padded in accordance with a set of per-format padding rules. (For a discription of record and block padding for all formats,

see the MPM write-ups of tape_ansi_ and tape_ibm_.) Because of padding rules and treatment of new-line characters when writing tape, a file that is written out to tape may not appear the same when read back in from tape. It is therefore recommended that the following suggestions be heeded:

1) do not use F or FB format.

2) to write character data with tape_ansi_, use D, DB, S, or SB format, with the maximum block length, and the record length chosen so that nelem is greater than the longest line in the storage system file.

3) to write binary data with tape_ansi_, use D, DB, S, or SB format, with the maximum permissible block and/or record lengths.

4) to write character data with tape_ibm_, use VBS format with the maximum block length, and the record length chosen so that nelem is greater than the longest line in the storage system file.

5) when transfering sequential format files to tape, use a variable length record format (D, DB, S, or SB) to avoid unwanted padding characters being inserted into records.

Examples

Below are examples of two typical control files. In the first example, the user wishes to produce 2 tapes, one for Multics, the other for an OS installation. The Multics tape will contain the source code of user subsystem SUBSYS, as well as it's object code. The OS tape will contain only the source code. In the second example, the user wishes to load into the storage system, the contents of volume "2314dp" which contains a dump of a disk pack containing source and data.

The numbers at the left-hand side of the page in the examples below do not actually appear in the control file, but are included only for annotation reference.

Example: sample1.tcl

command line: tape_out sample1.tcl

```
1         Volume: 001234;
2         /* Dump source in S format */
3         File: FILE_1;
4         number: 1;
5         Path: SUBSYS.pl1;
6         File:  FILE_2;
7         number: 2;
8         mode: binary;
9         Path: <object>SUBSYS;
10        format: U;
11        End;
12        Volume: DFG054;
13        /* append source to tape */
14        IO_Module: tape_ibm_;
15        File:  TEST_SAVE;
16        format: VBS;
17        block: 4096;
18        mode: ebcdic;
19        Path: SUBSYS.pl1;
20        End;
```

Annotations for sample1.tcl

1) causes volume 001234 to be mounted. The volume defaults are set to tape_ansi_, 800 bpi, ascii, SB format, block length = 2048, and record length = 1044480.

2) is a comment in the control file. As the storage statement is missing, the default storage system file format will be set to transfer unstructured files.

3) causes the tape to be positioned so that the first file that will be written on tape will be FILE_1.

4) this is required for writing the first file on a new ANSI or IBM file set at file position one on the tape.

5) specifies the path name of the storage system file to be written to tape. As the <file-group> contains no <local-statement>s other that the number statement the file will be written according to the current volume defaults. The tape file will be created as a new file on the volume.

6) causes the tape to be positioned so that the file to be written will be FILE_2.

7) the file to be written named FILE_2 will be the second file on the tape.

8) specifies that the file is to be written in binary mode.

9) specifies the path name of the storage system file to be written to tape.

10) specifies that the file is to be written in U format. Note that the block length will be the current volume default block length (2048), and that the record length is not applicable to U format.

11) signifies end of <volume-group>. The IO switch is closed and detached. The volume set is taken down and the device is released.

12) causes volume DFG054 to be mounted. The volume defaults are set to tape_ansi_, 800 bpi, ascii, S format, block length = 2048, and record length = 1044480.

13) is a comment
Storage format is still unstructured.

14) changes the volume IO_Module to tape_ibm_ and changes the volume-group> defaults to those associated with tape_ibm_.

Default density will be 1600 bpi.

15) specifies name of file to be written onto tape.

16) changes the record format to VBS. The default record length for VBS format is 1044480 bytes.

17) changes the block length to 4096.

18) changes the mode to ebcdic.

19) specifies the path name of the storage system file to be written.

20) causes the volume to be rewound and taken down as no more <file-group>s in the control file reference the current tape volume.

Example: sample2.tcl

command line: tape_in sample2.tcl

```
1       Volume: 2314dp;
2       /* Source Pack being loaded */
3       IO_Module: tape_ibm_;
4       Storage: sequential;
5       Density: 800;
6       Format: fb;
7       Record: 80;
8       Block: 800;
9       File: FILE1;
10      Path: <setup>data_entry>FILE1;
11      File: FILE2;
12      Path: <setup>data_entry>FILE2;
13      File: FILE3;
14      Path: <setup>data_entry>FILE3;
15      File: FILE4;
16      Path: <setup>data_entry>FILE4;
17      File: FILE5;
18      Path: <setup>data_entry>FILE5;
 .
 .
 .
58      File:  FILE28;
59      Path: <setup>data_entry>FILE28;
60      End;
```

Annotations for sample2.tcl

1) mounts the volume 2314dp with a write ring.

2) comment

3) read an IBM standard labeled tape.

4) storage system files will be created in sequential format, ready for use in record i/o. Notice that the mode will be the default for the tape_ibm_ IO Module, namely, ebcdic.

5) tape is recorded at 800 bpi.

6) all files on tape are in fixed block format.

7) all logical records are 80 characters (card image files).

8) all files blocked to 800 characters.

9) first file to be read from tape is named FILE1.

10) read tape file, FILE1, into storage system file named FILE1. The relative path name, <setup>data_entry>FILE1, will be expanded.

11) continue reading files off the tape volume, one by one, into files in the storage system with the same name.

.
.
.

60) end of <volume-group> and end of control file.

ADDITIONAL OPTIONS AVAILABLE FOR THE TCL USER

A number of options are available to the user who wants to do more than the simple file transfer between storage and a new tape volume. These features need not be of concern to most users, but for the user with specialized needs, these additional options are explained below.

Multi-volume Files

Multi-volume files are specified in a control file by slightly more complicated Volume statement than shown above. The multiple <volid>s of such a volume set are separated from one another by commas and are listed either in the order in which they became members of the volume set, for input, or in the order in which they are candidates for volume set membership, for output. The entire volume set membership need not be specified in a Volume statement referencing a volume set, but the first (possibly only) member must be mentioned. Up to 64 <volid>s may be specified in a single control file Volume statement.

Volume switching for multi-volume files is handled automatically by the IO modules. If sufficient volume set members are given in the tcl control file, the volume switching will be transparent to the user. If insufficient members of a volume set are given or the membership is being developed, the user will be querried during execution, for names of additional volume set members.

Sending Messages to the Operator

If it is necessary for the user to have a message displayed on the operator's console, the comment phrase can be included in the Volume statement. The comment phrase consists of the keyword -comment followed by the text of the message; this phrase in enclosed in quotes. Whenever the volume with the <volid> immediately preceding the comment phrase is to be mounted, the specified message will be displayed on the operator's console. The message may be concerned with any subject, but it is

typically used to display the slot identifier of the tape being mounted when it differs from the volume label. The message, text, may be from 1 to 64 characters and must be a contiguous string with no embedded spaces.

```
Volume: 060082 -comment "tape_is_Smith's", 060083
        -comment "tape also Smith's";
```

Protecting Tape Files From Accidental Overwriting

```
Expiration: <date>;
expiration: <date>;
```

The Expiration <global-statement> and the expiration <local-statement> specify the expiration date of a file to be written (created). <date> must be a contiguous string, with no embedded spaces and must be of a form acceptable to the convert_date_to_binary subroutine, for example "09/12/77". (see MPM writeup on convert_date_to_binary subroutine.) Because overwriting a file on a tape logically truncates the file set at the point of overwriting, the expiration date of a file must be earlier than or equal to the expiration date of the previous file (if any) on the tape; otherwise an error is indicated. If an attempt is made to overwrite an unexpired file, the user will be querried for explicit permission at the time of writing.

Special Output Modes

Normally, when a user sets up a tcl control file <file-group> to write a storage system file onto a tape volume, that is for use with the tape_out command, it is intended that a new file be created on the tape volume. The tcl default output mode is create. This is the only output mode available for unlabeled tapes. For labeled tapes however, the tcl language offers three additional specialized output modes; they are the replace, extend, and modify. The replace mode causes the tape file labels to be rewritten using specified and default file structure attributes. The extend and modify <local-statement>s do not cause the tape file labels to be recomposed, so any file attributes specified in the control file <file-group> or

<volume-group> which do not match those recorded in the tape labels, will cause an error to be indicated. The description of the three <local-statement>s for specialized output mode designation follow.

replace: <fileid>;

If an existing tape file is to be replaced on an ANSI or IBM standard labeled tape and its name is known, the file to be overwritten is identified by <fileid> in the replace <local-statement> and the new file to be written is identified by <fileid> in the File statement. If the file identified in the replace statement does not exist, an error is indicated.

File: <fileid>;
replace: <fileid>;

extend;

The extend <local-statement> allows new data records to be appended to an existing file on an ANSI or IBM standard labeled tape without in any way altering the previous contents of the tape file. The tape file to be extended is identified by the File statement or by the File statement and number <local-statement> in combination. If the tape file to be extended does not exist on the tape, an error is indicated. Recorded in the labels of an ANSI or IBM labeled tape file is the version number. Initially it is zero when the file is created. Every time a file is extended, its version number is incremented. The version number field is two digits and is reset to zero when the one-hundredth revision is made.

modify;

The modify <local-statement> causes the entire contents of a file on an ANSI or IBM labeled tape to be replaced while retaining the structure of the file itself. The file to be modified is identified by the File statement, or by a combination of the File statement and the number statement.

Example: sample3.tcl

command line: tape_out sample3.tcl -fc

```
1     Volume: 070067 "-comment in_slot_10000" 070068;
2     IO_Module: tape_ansi_;
3     File: BIG_LISTING;
4     replace: FILE_20;
5     number: 20;
6     expiration: 2weeks;
7     format: db;
8     block: 2048;
9     record: 133;
10    Path: >udd>Example>Mega>test.list
11    End;
```

Annotations for sample3.tcl

1) The first member of the volume set, 070067, is mounted with a ring, displaying the message "in_slot_10000" on the operator's console. Later if necessary, the volume set member 070068 may be mounted to continue writing a large listing file. No message will appear upon mounting the second member of the volume set.

2) writing an ANSI standard tape.

3) tape file named BIG_LISTING, into which the storage system file is to be written.

4) is to replace tape file named FILE_20.

5) by the number statement FILE_20 is the 20th file on the current volume set.

As no density statement is included in the control file, the default for tape_ansi_, 800 bpi, will be used.

Upon execution of the control file, the tape will be positioned at the 20th file automatically.

As no Storage statement is present in the control file, the default storage system format is unstructured.

6) The file, BIG_LISTING, will be protected against accidental overwriting for two weeks, meaning that if the user attempts to overwrite the file within that time, he will first be querried for permission to do so.

The -force option in the command line will inhibit a query for permission to overwrite FILE_20, in case it has not yet expired.

7) BIG_LISTING will be recorded in variable length blocked record format.

Mode will be the default for tape_ansi_, namely ascii.

8) Blocking factor is maximum allowed, 2048.

9) record length is 133, for printer line width.

10) the listing file will be transfered from test.list in the storage system.

11) signifies termination of <volume-group> and of control file.

If after putting his listing file out onto tape, the user then wishes to delete the on-line listing, and at a later time, read the listing back from tape into storage, he might type the command line

```
tape_in sample3.tcl
```

The output statements in the control file, namely the replace <local-statement> and the expiration <local-statement> will be ignored on input.

370/DOS Tapes

The tape_ibm_ IO Module will process tapes created by or destined for IBM/DOS installations as well as tapes for IBM/OS installations. The DOS <global-statement> is used in the tcl control file to specify that the tape files referenced by the given <volume-group> are destined for or have been produced by a IBM/DOS installation. The important difference between tape files created by OS and those created by DOS operating system is that the tape file structure attributes are not recorded in the tape labels under DOS. It is therefore necessary for all of the structure attributes of a DOS tape file, namely encoding mode, logical record format, logical record length, and block size to be specified in the tcl control file. If not all tape file structure attributes have been specified by <global-statement>s and <local-statement>s for each <file-group> in the scope of the DOS <global-statement>, an error is indicated.

Example: sample4.tcl Control File for Reading DOS Tape

command line: tape_in sample4.tcl

```
1     Volume: 042281;
2     IO_Module: tape_ibm_;
3     DOS;
4     Density: 800;
5     Storage: structured;
6     Mode: ebcdic;
7     File: abc;
8     number: 1;
9     record: 80;
10    block: 800;
11    format: fb;
12    Path: >udd>Example>Foo>fargo.pl1
13    End;
```

Annotations for sample4.tcl

Note: Only selected statements in the control file are annotated here.

1) mount volume 042281 without a ring.

2) read IBM standard tape.

3) DOS tape will be processed.

5) read tape files into storage system as structured format files.

Unlabeled Tapes

The tape_ibm_ IO Module supports processing of unlabeled tapes, provided that the tapes are structured according to the OS standard. DOS leading tape mark (LTM) unlabeled format tapes cannot be processed however. The No_labels <global-statement> is included in the control file to specify that an unlabeled tape is to be processed. The No_labels <global-statement> is mutually exclusive with any statement, global or local, which refers to labeled tapes: namely, the DOS and Expiration <global-statement>s and the replace, modify and extend <local-statement>s. If any of these appear together within the same control file <file-group>, an error is indicated. When referencing unlabeled tape files in a given <file-group>, the argument of the File statement, <fileid>, is specified by "*", and the tape file desired must be specified by the number <local-statement>.

Example: sample5.tcl Control File for Reading an Unlabeled Tape

command line: tape_in sample5.tcl

```
1     Volume: 042381;
2     IO_Module: tape_ibm_;
3     No_labels;
4     Storage: unstructured;
5     File: *;
6     number: 3;
7     Path: >udd>Expamle>Foo>foobar.data
8     End;
```

Annotations for sample5.tcl

Note: Only selected statements in the control file are annotated here.

3) unlabeled tape is to be read. Files will be unnamed. This statement must appear when processing unlabeled tapes.

5) <fileid> is specified by "*" for unnamed files.

6) the number statement must be present when processing unlabeled tapes. The third file on the tape will be read.

The default tape file record format is VBS, the default tape file record length for VBS format is 1044480 characters, and the default tape file block length is 8192 characters.

APPENDIX A

Error Diagnostics

The error messages which are issued during tape_in, tape_out compilation are graded and have the form shown below.

prefix error_number, SEVERITY severity IN STATEMENT m OF LINE n
text of error message
SOURCE:
source_statement_in_error

where n is the line number on which the described statement began and m is a number identifying which statement in line n was in error. If line n contains only one statement then "STATEMENT m OF" is omitted from the error message.

The severity numbers produce one of the following prefixes:

severity	prefix	explanation
0	COMMENT	the error message is a comment.
1	WARNING	the error message warns that a possible error has been detected. However, the translation will still proceed.
2	ERROR	the error message warns that a probable error has been detected. However, the error is non-fatal, and the translation will still proceed.
3	FATAL ERROR	the error message warns that a fatal error has been detected. Processing of the input will still continue to diagnose further errors, but no translation will be performed.
4	TRANSLATOR ERROR	the error message warns that an error has been detected in the operation of the translator. No translation will be performed.

APPENDIX B

Execution Time Errors

Any fatal error from an IO module during execution of a control file will cause the user to be querried as to whether or not he wishes to continue processing the other <file-group>s and <volume-group>s in the control file or whether to terminate processing of the control file. Specific errors are described below.

storage system file not found

If a control file is being executed and a storage system file given in the control file cannot be located, the user will be querried as to whether he wishes to halt the processing of the control file and see if he can fix the problem of the missing storage system file, or whether he wishes to continue processing onto the next <file-group> or <volume-group> in the control file, if one exists or terminate processing if no other <file-group> or <volume-group>s exist. The user will resume processing after locating the file by typing "start".

tape file not found

If a control file is being executed and a specified tape file is not found, the user will be querried as to whether he wishes to continue processing the control file or whether he wishes to terminate processing.

tape volume not found

If a tape volume specified in a control file cannot be located, the user will be querried as to whether he wishes to continue processing the control file's other <volume-group>s or whether he wishes to terminate processing.

file name duplication

If a control file is being executed, and a storage system file with the path name specified in the control file already exists, the user will be querried as to whether he wishes to overwrite the existing storage system file or not. If he choses not to overwrite, he will be querried as whether to terminate processing or continue on to the next <file-group> in the control file.

APPENDIX C

Some tcl statements are for use with one IO module only. Below is a summary of tcl <global-statement>s and the name of the IO module which supports the tcl statement.

<global-statement>	Supporting IO Module
Density: <den>;	tape_ansi_ or tape_ibm_
DOS;	tape_ibm_ only
No_labels;	tape_ibm_ only
Storage: <structure>;	
Expiration: <date>;	tape_ansi_ or tape_ibm_
Mode: <mode>;	tape_ansi_ or tape_ibm_
Format: <form>;	tape_ansi_ or tape_ibm_
Block: <blklen>;	tape_ansi_ or tape_ibm_
Record: <reclen>;	tape_ansi_ or tape_ibm_

APPENDIX D

Summary of tcl <local-statement>s and the IO Module which supports the tcl statement

<local-statement>	Supporting IO Module
storage: <structure>;	
expiration: <date>;	tape_ansi_ or tape_ibm_
mode: <mode>;	tape_ansi_ or tape_ibm_
format: <form>;	tape_ansi_ or tape_ibm_
block: <blklen>;	tape_ansi_ or tape_ibm_
record: <reclen>;	tape_ansi_ or tape_ibm_

Additional <local-statement>s which have no global counterparts follow. Again, some <local-statement>s are for use with specific IO modules only.

number: <number>;	tape_ansi_ or tape_ibm_
replace;	tape_ansi_ or tape_ibm_
extend;	tape_ansi_ or tape_ibm_
modify;	tape_ansi_ or tape_ibm_

APPENDIX E

IO Module Compatibility and peiem Tables

tape_ansi_

mode: ascii (default) | binary | ebcdic
block length: 18 ≤ b ≤ 2048 bytes
density: d = 800 (default) | 1600
file sequence number: 1 ≤ n ≤ 9999
record length: < r < 1044480
format: f = fb | f | db | d | s | sb (default) | u

tape_ibm_

mode: ascii | ebcdic (default)
block length: 18 ≤ b ≤ 8192 bytes
density: d = 800 | 1600 (default)
file sequence number: 1 ≤ n ≤ 9999
record length: ≤ r ≤ 1044480
format: f = fb | f | vb | v | vbs (default) | u

Format	Record Length in bytes	Block Length in bytes	nelem
	r	b	
u	r is undefined	= amrl ≤ b ≤ 8192	= b
f	r = amrl	b = r	= r
fb	r = amrl	b must statisfy mod(b,r) = 0	= r
d	amrl+4 ≤ r ≤ 2048	b = r	= r
db	amrl+4 ≤ r ≤ 2048	b ≥ r	= r
s	amrl ≤ r ≤ 1044480	18 ≤ b ≤ 2048	= b
sb	amrl ≤ r ≤ 1044480	18 ≤b ≤ 2048	= b
v	amrl+4 ≤ r ≤ 8188	b = r + 4	= r
vb	amrl+4 ≤ r ≤ 8188	b ≥ r + 4	= b
vs	amrl ≤ r ≤ 1044480	20 ≤ b ≤ 8192	= b
vbs	amrl ≤ r ≤ 1044480	20 ≤ b < 8192	= b

Notes:

nelem is the number of bytes involved in a given i/o transaction, i.e. a read or a write operation. nelem is equal to either the block length or to the logical record length depending upon the given record format. The value of r is dependent on the choice of record format. amrl is the actual or maximum record length. In every case b must be an integer in range of 18 ≤ b ≤ 8192. For ANSI tapes, in order to comply with the ANSI standard, b must be in the range of 18≤ b ≤ 2048. For IBM tapes, the condition mod(b,4) = 0 must be satisfied. The tcl record statement should not be used for a U-format file transfer.

 (END)