MSCDEX Command Services.

There is a need for access to features from the MSCDEX redirector that transend DOS capabilities. This proposal documents a means the application can use to talk directly to MSCDEX to request information or set parameters that only MSCDEX can provide. This document outlines the services MSCDEX provides at present. Comments and suggestions are welcome.

Access to these functions is provided through an INT 2Fh interface. AH contains 15h which is what MSCDEX will use to tell its requests from those of other INT 2Fh handlers. AL will contain the code of the function to be performed. When writing applications for Windows, all pointers must be allocated in lower memory and properly translated to real mode addresses through the DOS Protected Mode Interface (DPMI). Information on DPMI is available from the Intel Corporation.

Function Request Command Codes:

Contents of AL	Function
00h	Get Number of CD-ROM drive letters
01h	Get CD-ROM drive device list
02h	Get copyright file name
03h	Get abstract file name
04h	Get bibliographic doc file name
05h	Read VTOC
06h	Reserved
07h	Reserved
08h	Absolute Disk Read
09h	Absolute Disk Write
0Ah	Reserved
0Bh*	CD-ROM Drive Check
0Ch*	MSCDEX Version
0Dh*	Get CD-ROM drive letters
0Eh*	Get/Set Volume Descriptor Preference
0Fh*	Get Directory Entry
10h**	Send Device Request
11h-0FFh	Reserved

* - New functions added in version 2.00
** - New functions added in version 2.10

- Get Number of CD-ROM drive letters

AX 1500h
BX Number of CD-ROM drive letters used
CX Starting drive letter of CD-ROM drive letters (A=0, B=1, ... Z=25)

MSCDEX will return the number of CD-ROM drive letters in BX and the starting drive letter in CX. The first CD-ROM device will be installed at the starting drive letter and subsequent drives will be assigned the next greater drive letter. A single device driver may be assigned to more than one drive letter, such as the case of a device driver that supports multiple units. MSCDEX keeps track of which sub-unit a particular drive letter is assigned to.

NOTE: This function can be used to determine if MSCDEX is installed by setting BX to zero before executing INT 2Fh. MSCDEX is not installed if BX is still zero on return.

Also, in a networking environment, one cannot assume that drive letters will always be assigned contiguously beginning with the starting drive letter. Use function Get CD-ROM drive letters instead.

- Get CD-ROM drive device list

AX 1501h
ES:BX Transfer address; pointer to buffer to copy drive letter device list

---

The buffer must be large enough to hold the device list. By calling function *Get Number of CD-ROM Drive Letters*, one can find out the number of CD-ROM drive letters and the buffer size will be a multiple of that. This will be an absolute maximum of 26. Each drive letter device entry will consist of one byte for the sub-unit followed by 4 bytes for the address of the device header assigned to that drive letter. This byte for the sub-unit takes care of the problem of distinguishing which unit is assigned to which drive letter for device drivers that handle sub-units.

For example: Suppose there are two installed CD-ROM device drivers, CD_ONE, which supports 1 sub-unit, and CD_TWO, which supports two sub-units, on a system with 2 floppy drives (A=0 and B=1) and a hard disk (C=2). Then asking

for the number of CD-ROM drive letters will report that there are 3 drive letters used starting at drive letter D=3. ES:BX must point to a buffer that is at least 3*5 = 15 bytes long. The buffer will be filled as follows:

```
ES:BX = Buffer
Buffer  DB      0                               ; sub-unit of CD_ONE on drive letter
D:
        DD      <far addr of CD_ONE device header>
        DB      0                               ; sub-unit of CD_TWO
                                                ;on drive letter E:
        DD      <far addr of CD_TWO device header>
        DB      1                               ; sub-unit of CD_TWO
                                                ; on drive letter F:
        DD      <far addr of CD_TWO device header>
```

- Get copyright file name

_____

```
        AX      1502h
        ES:BX Transfer address; pointer to a 38 byte buffer
        CX      CD-ROM drive letter (A=0, B=1, ... Z=25)
```

_____

MSCDEX will copy the name of the copyright file in the VTOC for that drive letter into the buffer space provided. The copyright filename is presently restricted in the High Sierra proposal to 8.3 but we require 38 bytes here for the possibility at a later date of handling 31 character file names plus 6 bytes for a ';' and 5 digit version number and 1 byte for a NULL at the end. Carry will be set if the drive letter is not a CD-ROM drive and error_invalid_drive (15) will be returned in AX.

- Get abstract file name

_____

```
        AX      1503h
        ES:BX Transfer address; pointer to a 38 byte buffer
        CX      CD-ROM drive letter (A=0, B=1, ... Z=25)
```

_____

MSCDEX will copy the name of the abstract file in the VTOC for that drive letter into the buffer space provided. The abstract filename is presently restricted in the High Sierra proposal to 8.3 but we require 38 bytes here for the possibility at a later date of handling 31 character file names plus 6 bytes for a ';' and 5 digit version number and 1 byte for a NULL at the end. Carry will be set if the drive letter is not a CD-ROM drive and error_invalid_drive (15) will be returned in AX.

- Get bibliographic documentation file name

_____

```
        AX      1504h
        ES:BX Transfer address; pointer to a 38 byte buffer
        CX      CD-ROM drive letter (A=0, B=1, ... Z=25)
```

_____

NOTE: This function is provided in advance of the ISO standard. For discs complying with the May 28th draft from the High Sierra Group, this function will return a null string as though the field is blank on the disc.
MSCDEX will copy the name of the bibliographic documentation file in the VTOC for that drive letter into the buffer space provided. The bibliographic documentation filename is presently restricted in the High Sierra proposal to 8.3 but we require 38 bytes here for the possibility at a later date of handling 31 character file names plus 6 bytes for a ';' and 5 digit version number and 1 byte for a NULL at the end. Carry will be set if the drive letter is not a CD-ROM drive and error_invalid_drive (15) will be returned in AX.

- Read VTOC

_____

AX 1505h
ES:BX Transfer address; pointer to a 2048 byte buffer
CX CD-ROM Drive letter
DX Sector index

_____

This function is provided to scan the Volume Descriptors on a disc. A sector index of 0 will read the first volume descriptor, 1 reads the second, etc. If there is no error, then AX will return 1 if the volume descriptor read was the standard volume descriptor, 0FFh if it was the volume descriptor terminator and there are no more volume descriptors to be read, and 0 for all other types.
If there is an error in processing the request, the Carry Flag will be set and AL will contain the MS-DOS error code. These will be either error_invalid_drive (15) or error_not_ready (21).

- Absolute Disk Read

_____

AX 1508h
ES:BX Disk Transfer Address; pointer to a buffer to copy data to
CX CD-ROM Drive letter (A=0, B=1, ... Z=25)
DX Number of sectors to read
SI:DI Starting sector

_____

This function corresponds to INT 25h. It will be converted directly into a READ_LONG device driver request and sent to the correct device driver. There are no requirements for this call to pop flags as there are with INT 25h. SI holds the high word and DI the low word for the starting sector to begin reading from.
If there is an error in processing the request, the Carry Flag will be set and AL will contain the MS-DOS error code. These will be either error_invalid_drive (15) or error_not_ready (21).

- Absolute Disk Write

_____

AX 1509h

ES:BX Disk Transfer Address; pointer to buffer to copy data from
CX CD-ROM Drive letter
DX Number of sectors to write
SI:DI Starting sector

This function corresponds to INT 26h. It is not supported at this time and is reserved. It is intended to be used by authoring systems.

- CD-ROM Drive Check

AX 150Bh
BX Signature word
CX CD-ROM Drive letter (A=0, B=1,...Z=25)

This function returns whether or not a drive letter is a CD-ROM drive supported by MSCDEX. If the extensions are installed, BX will be set to ADADh. If the drive letter is supported by MSCDEX, then AX is set to a non-zero value. AX is set to zero if the drive is not supported. One must be sure to check the signature word to know that MSCDEX is installed and that AX has not been modified by another INT 2Fh handler.

- MSCDEX Version

AX 150Ch
BX MSCDEX Version

This function returns the version number of the CD-ROM Extensions installed on the system. BH contains the major version number and BL contains the minor version. Values returned are binary. For example, BX would contain 0x020a for version 2.10. This function does not work on versions earlier than 2.00 so if BX is zero before and after this function is called, an earlier version of MSCDEX is installed.

- Get CD-ROM Drive Letters

AX 150Dh
ES:BX Transfer address; pointer to buffer to copy drive letter device list

---

The buffer must be large enough to hold a list of drive letters. The buffer size will be a multiple of the number of drives returned by the *Get Number of CD-ROM drive letters* function. There are a maximum of 26 drive letters. Each drive letter entry is a single byte (0=A:, 1=B: .. 25=Z:) that exactly corresponds each respective entry returned by the command *Get CD-ROM drive device list.* This command is included to allow applications to locate CD-ROM drives supported by MSCDEX. CD-ROM drive letters may sometimes be noncontiguous so this command is necessary.

For example: Suppose there is an installed CD-ROM device driver FOO supporting 3 sub-units on a system with 2 floppy drives (A=0 and B=1), a hard disk (C=2) and a network drive (E=4). Note the network drive occupies one of the drive letters normally taken by a CD-ROM drive. MSCDEX assigns that CD-ROM drive to the next available drive letter. Asking for the number of CD-ROM drive letters reports there are 3 drive letters used starting at drive letter D=3. ES:BX must point to a buffer that is at least 3 bytes long and will be filled as follows:

```
ES:BX = Buffer
Buffer  DB      3                       ; drive letter for CD-ROM (D=3)
        DB      5                       ; drive letter for CD-ROM (F=5)
        DB      6                       ; drive letter for CD-ROM (G=6)
```

- Get/Set Volume Descriptor Preference

_____

```
        AX      150Eh
        BX      0 - Get Preference. 1 - Set Preference
        CX      CD-ROM Drive letter (A=0, B=1,...Z=25)
        DX      if BX = Get Preference
                        DX = 0
                        MSCDEX will return preference settings in DX
                if BX = Set Preference
                        DH = volume descriptor preference
                                1 - PVD - Primary Volume  Descriptor
                                2 - SVD - Supplementary Volume Descriptor
                        DL = Supplementary Volume Descriptor Preference
                                if DH = PVD
                                        DL = 0
                                if DH = SVD
                                        1 - shift-Kanji (an unregistered ISO coded
character set)
```

_____

Normally, MSCDEX will scan for the PVD (Primary Volume Descriptor) when initializing a CD-ROM. This behavior can be altered for each individual drive to scan for a SVD (Supplementary Volume Descriptor) instead. A CD-ROM drive set to scan for an SVD will use the PVD if there is no SVD present. There can be more than one SVD on a CD-ROM but at present, MSCDEX will only recognize SVDs for shift-Kanji CD-ROMs. Carry will be set, AX will be set to error_invalid_function (1) and DX will be set to 0 if the coded character set is not recognized.

If BX contains Get_Preference, MSCDEX will report the present setting for that drive. If DX is still zero on return, that version of MSCDEX does not support this function or reading SVDs. Otherwise DX will contain the setting.

If the drive letter is not a CD-ROM drive, carry will be set and

error_invalid_drive (15) will be returned in AX. If BX is anything other than Get/Set_Preference, AX will be set to error_invalid_function (1) and carry will be set.

- Get Directory Entry

_____

AX	150Fh
CL	CD-ROM Drive letter (A=0, B=1,...Z=25)
CH	Copy flags (bit 0: 0 - direct copy, 1 - copied to structure)
ES:BX	Pointer to buffer with null-terminated path name
SI:DI	Pointer to buffer to copy directory record information
AX	0 is returned if the disc is High Sierra, 1 is returned if the disc is ISO-9660

_____

The pathname expected is a null-terminated string e.g. char far *path = "\\a\\b\ \c.txt"; (note: the "\\" characters map to a single '\' character in C so this would be '\a\b\c.txt' if printed). The path must consist only of valid High Sierra or ISO-9660 filename characters and must not contain any wildcards nor may it include entries for '.' or '..'.

The copy flags indicate how the data should be copied. If bit 0 (0x01) is 0, then the directory entry is copied as it appears on the disc. The directory record is a direct copy from the directory file and it is up to the application to choose what fields to use. If bit 0 is 1, then the directory entry is copied into a structure that removes any differences between High Sierra and ISO-9660 directory entries and makes some fields more accessible or easily interpreted.

If copying the directory entry as it appears on the disc, the buffer to copy the directory record should be at least 255 bytes long to include all system use information and if copying into the common structure at least 280 bytes long otherwise MSCDEX may overrun the end of the buffer.

Carry will be set and an error code returned if there were problems with the request. The error codes will be error_invalid_drive (15) if the drive letter is incorrect, error_not_ready (21) if the disc didn't initialize correctly, error_file_not_found (2) if the file was not found and error_no_more_files (18) if the pattern fails to find a match or if MSCDEX failed to allocate buffers.

The format of the directory record for High Sierra discs is:

```
/* High Sierra directory entry structure */
typedef struct hsg_dir_entry
{
        uchar    len_dr;            /* length of this directory entry
        */
        uchar    XAR_len;           /* XAR length in LBN's (logical
        */
                                                                    /
* blocks  numbers)                  */
```

```
        ulong   loc_extentI;        /* LBN of data Intel format
        */
        ulong   loc_extentM;        /* LBN of data Molorola format
        */
        ulong   data_lenI;          /* length of file Intel format
        */
        ulong   data_lenM;          /* length of file Motorola format
        */
        uchar   record_time[6];     /* date and time
        */
        uchar   file_flags_hsg;     /* 8 flags
        */
        uchar   reserved;           /* reserved field
        */
        uchar   il_size;            /* interleave size
        */
        uchar   il_skip;            /* interleave skip factor
        */
        ushort  VSSNI;              /* volume set sequence number Intel
        */
        ushort  VSSNM;              /* volume set sequence number
        */
                                                                        /
* Motorola    */
        uchar   len_fi;             /* length of name
        */
        uchar   file_id[...];       /* variable length name upto 32
        */
                                                                        /
* chars       */
        uchar   padding;            /* optional padding if file_id is
        */
                                                                        /
* odd length  */
        uchar   sys_data[...]       /* variable length system data
        */
        } hsg_dir_entry;
```

The format of the directory record for ISO-9660 discs is:

```
/* ISO-9660 directory entry structure */
typedef struct iso_dir_entry
{
        uchar   len_dr;             /* length of this directory entry
```

```
        */
        uchar   XAR_len;        /* length of XAR in LBN's
        */
        ulong   loc_extentI;    /* LBN of data Intel format
        */
        ulong   loc_extentM;    /* LBN of data Molorola format
        */
        ulong   data_lenI;      /* length of file Intel format
        */
        ulong   data_lenM;      /* length of file Motorola format
        */
        uchar   record_time[7]; /* date and time
        */
        uchar   file_flags_iso; /* 8 flags
        */
        uchar   il_size;        /* interleave size
        */
        uchar   il_skip;        /* interleave skip factor
        */
        ushort  VSSNI;          /* volume set sequence num Intel
        */
        ushort  VSSNM;          /* volume set sequence num Motorola
        */
        uchar   len_fi;         /* length of name
        */
        uchar   file_id[...];   /* variable length name upto 32
        */
                                                                    /
*chars*/
        uchar   padding;        /* optional padding if file_id is
        */
                                                                    /
*odd length   */
        uchar   sys_data[...]   /* variable length system data
        */

} iso_dir_entry;
```

Note that the difference between the two forms is the file flag byte moved to account for an additional byte of date and time used for a Greenwich mean time offset. Also, the C structs above are not syntactically correct as C does not allow variable length arrays as struct elements. They are meant to illustrate the differences between the directory entries. See the May 28th draft of the High

Sierra proposal or ISO-9660 for a more complete explanation of the fields.
If bit 0 is set to one in the Copy Flags, then the format of the directory entry structure that is returned is the following:

```
typedef struct dir_entry
 {
        uchar    XAR_len;          /* length of XAR in LBN's
        */
        ulong    loc_extent;       /* logical block number of file start
        */
        ushort   lb_size;          /* logical block size of disc
        */
        ulong    data_len;         /* length of file
        */
        uchar    record_time[7];   /* date and time
        */
        uchar    file_flags;       /* 8 flags
        */
        uchar    il_size;          /* interleave size
        */
        uchar    il_skip;          /* interleave skip factor
        */
        ushort   VSSN;
        /* volume set sequence number
        */
        uchar    len_fi;           /* length of the filename
        */
        uchar    file_id[38];      /* filename, null terminated
        */
        ushort   file_version;     /* version number of file
        */
        uchar    len_su;           /* length of valid system use bytes
        */
        uchar    su_data[220]      /* up to 220 bytes of system use data
        */
        } dir_entry;
```

The location of the extent is reported in logical block numbers and the caller can use the logical block size on the disc (logical block sizes of 512, 1024, and 2048 bytes per sector are valid for CD-ROM) to determine the exact sector and offset in that sector that the file extent begins in. If the logical block size is 2048 then logical block numbers are equivilent to logical sector numbers.
The Greenwich mean time offset byte in this structure for May 28 High Sierra discs is always 0. The file_id field contains the file identifier less the semicolon

and version number if present and is null terminated. The version number is already parsed and is present in binary form in file_version. All bytes beyond what are indicated by len_fi and len_su in the file_id and su_data fields are undefined except the null byte immediately following the file identifier which is not counted as part of the filename length.

- Send Device Driver Request

_____

AX 1510h
CX CD-ROM drive letter (A=0, B=1, ... Z=25)
ES:BX Address of CD-ROM device driver request header

_____

This function has been added to simplify communication with CD-ROM drivers and help prevent contention between applications that wish to communicate with the device driver. It is highly recommended that all applications communicate with device drivers through this function request. Applications using this function will not have to locate the device driver. The format of the request header is specified by the Microsoft MS-DOS CD-ROM Extensions Hardware-dependent Device Driver Specification. MSCDEX will supply the sub-unit field.

End.