GEAs

A GEA is an attribute name. Its format is:

```
struct GEA {
    unsigned char cbName;       /* length of name    */
    unsigned char szName[];     /* ASCIIZ name       */
};
```

The name length does not include the trailing NUL.

A list of GEAs is a packed set of GEA structures preceded by a length of the list (including the length itself) as indicated in the following structure:

```
struct GEAList {
    unsigned long cbList;    /* length of list     */
    struct GEA list[];           /* packed set of GEAs */
};
```

GEA lists are used for retrieving the values for a particular set of attributes. A GEA list is used as input only.

Name lengths of 0 are illegal and are considered in error. A value length of 0 has special meaning. Setting an EA with a value length of 0 will cause that attribute to be deleted (if possible). Upon retrieval, a value length of 0 indicates that the attribute is not present.

Setting attributes contained in an FEA list does not treat the entire FEA list as atomic. If an error occurs before the entire list of EAs has been set, all, some, or none of them may actually remain set on the file. No program should depend on an EA set being atomic to force EAs to be consistent with each other. Programs must be careful not to depend on atomicity, since a given file system may provide it.

Manipulation of extended attributes is associated with access permission to the associated file or directory. For querying and setting file EAs, read and write/read permission, respectively, for the associated file is required. No directory create or delete may occur while querying EAs for that directory.

For handle-based operations on extended attributes, access permission is controlled by the sharing/access mode of the associated file. If the file is open for read, querying the extended attributes is allowed. If the file is open for write, setting the extended attributes is allowed. These operations are DosQFileInfo and DosSetFileInfo.

For path-based manipulation of extended attributes, the associated file or directory will be added to the sharing set for the duration of the call. The requested access permission for setting EAs is write/deny-all and for querying EAs is read/deny-write. The path-based API are DosQPathInfo, DosSetPathInfo, and DosFindFirst2/DosFindNext.

For create-only operations of extended attributes, the extended attributes are set without examining the sharing/access mode of the associated file/directory. These operations are DosOpen2 and DosMkDir2.

The routing of EA requests is accomplished by the IFS routing mechanism. EA requests that apply to names are routed to the FSD attached to the specified drive. Those requests that apply to a handle (file or directory) are routed to the FSD attached to the handle. No interpretation of either FEA lists nor GEA lists is performed by the IFS router.

Note: It is the responsibility of each FSD to provide support for EAs.

It is expected that some FSDs will be unable to store EAs; for example, UNIX and MVS compatible file systems.

Note: The FAT FSD implementation will provide for the complete implementation of EAs. There will be no special EAs for the FAT FSD.

All EA manipulation is performed using the following structure: The relevance of each field is described within each API.

```
struct EAOP {
    struct GEAList far * fpGEAList; /* GEA set           */
    struct FEAList far * fpFEAList; /* FEA set           */
    unsigned long offError;         /* offset of FEA err */
};
```

See the descriptions of the file system function calls in OS/2 Version 2.0 Control Program Programming Reference for the relevance of each field.

In OS/2 Version 2.0, values of cbList greater than (64K-1) are not allowed. This is an implementation-defined limitation which may be raised in the future. Because this limit may change, programs should avoid enumerating the list of all EAs, but instead manipulate only EAs that they know about. For operations such as copying, the DosCopy API should be used. If enumeration is necessary, the DosEnumAttribute API should be used.

A special category of attributes, called create-only attributes, is defined as the set of extended attributes that a file system may only allow to be set at creation time. (Such attributes may be used to control file allocation and structure configuration.) File systems are expected to allow create-only attributes to be set at any time from when the object is created to when it is first modified, that is, data is written into a file or an entry added to a directory. Programs that copy objects should copy all of the EAs for an object before otherwise modifying it in order to assure that any create-only attributes from the source are properly applied to the target. The DosCopy API is the preferred method of copying files or directories.