Specific actions for directories (sub list specifications) etc.

I) Object creation (different operation for emn and ob-object, differentiated by a third parameter?)
   A) Non scratch
      open directory (with object creation bit-on)
      text-name (use block data file?)
      data to define object
   B) Scratch
      open directory (with object creation bit-on)
      text-name (use block data file?)
      accessory mumber
      data to define object

II) Object access (produces closed version of the object)
    open directory
    text-name (use block data file?)
    accessory
    c-list index
    for return capability [use capability when feature?] [probably!]
The directory is scanned by some algorithm for a match with the text name. If no match is found, the action fails, either by error or F-return. I think F-return is best. If a match is found, then the access is authenticated and an option bit field constructed as follows:

If the access key is not null, the access list for the entry is scanned for a matching access key number. If not found, then the action fails (error? F-return?). If found, an option bit field is formed by and-ing the option bits associated with the access key number in the access list with the option bit field of the presented access key.

If the access key is null, and the implicit access bit is on in the capability for the directory then:

- If the scratch bit is not on in this entry, then an option bit field with all options on is formed.
- If the scratch bit is on, then an option bit field with only the destruction bit on is formed.

If this entry was a soft link entry, the above actions are re-entrant with the directory pointed to by the link. The text name to be used is that in the link. The access key to be used has the same number as that in the link, and the option bit field just constructed. Of course an actual access key does not have to be constructed. [The director has, of course, to be opened.]

[Then closed when done]
Having received either an ownership entry or a hard link entry, a capability, representing the closed object is constructed as follows:

- The option bit field is that last constructed during the above part of the algorithm.
- The type field is that associated with this kind of object, (specified in the entry object part of the entry.) The data word is the full object part of the entry.
III entry modification

A) adding an access key to an access list

open directory

 directors

 text name  (use block data xfa?)

 access key

 access key number

First an entry is found and authenticated as in object access.

Then if the access key number is not already in the list it is

placed in the list. Finally the option bit field (constructed

is associated with this access key number in the access list.

B) deleting an access key from an access list

open directory

 directors

 text name  (use block data xfa?)

 access key

 access key number

If an entry is found as in A), if the option bit field constructed does

not have prohibition error, ELSE the access key number specified

is removed from the access list.
rename an entry

open directory

entry name

access entry

new entry name

as before, the constructed option bit field must have proper bit on.
also soft links are not followed (i.e. osin A)

destroy an entry

open directory

entry name

access entry

find and authenticate an access as in III a)

if destruction bit is on in constructed option bit field, then proceed,
else error.

if hard link entry, or soft link entry, just remove from the directory
if ownership entry, then object destruction bit must also be on,
and destroy object and remove entry from directory.
A description of the contents of the object designation field (and hence the data field of the efs capability representing the closed object) will appear in later documents. (As well as specifications for construction as well as implications of destruction.)

- directories
- files
- subprocess descriptors
- access keys
- global efs object
- efs goodie

Implemented as low level disk files