0) open and close, special

For objects of all kinds, the following generalities are true:

[Some objects cannot be opened! (access errors)]

1) A global open count is maintained for each object. This is the number of processes holding the object open.

2) A local open count is maintained in each process for each object it holds open.

The global open count for an object is increased and decreased by 1 each time a local open count for the object departs from 0 or reaches 0, respectively.

3) For some kinds of objects, there is a different capability representing the open form of the object from that representing the closed form of the object.

For these objects, closing is sometimes done by presenting a capability for the open form and sometimes by presenting a capability for the closed form.

4) Whenever a disc process is destroyed, the global open count is decremented for all objects which have a non-zero local open count,
1) directories

(These are low level disk objects.)

A) creation

[See object creation of document of 4/17/70]

additional parameters would be:

i) max size (uls, a pseudo integer?)

ii) whether this directory is to have a directly associated accounting clock and if so, how much each allocated item is to place in it one.

B) open action

does not return any capability, but does insure that the low level file representing the directory is open. (i.e. if a o-level file, all in ECS) Hence the open directory represented by the same capability as the closed directory

C) close action

applied to the same capability as the closed directory.

after enough closes, the directory file representing the directory will be moved back to the d.3sr. (global open count = 0)

Notes: all directory actions may be performed on a closed directory.

It will be temporarily opened for the action.
II) disk files

(These are low level disk objects)

A) creation

additional parameters (over object creation of 4/17/70)

would be only shape.

B) open action

This returns a new kind of capability (pos file)

C) close action

see Bruce for what kind of capability needed here
III) **subprocess descriptors**

(These are low level clisp objects)

A) **Creation**

See a special document on subprocess descriptors
(a new one not yet ready)

B) **open**

Similar to directories. This just causes the low level file
representing the subprocess descriptor to be brought
into ESS. No new capability is returned.

All actions may be performed by the user on an
unopened descriptor. It will be temporarily opened
for the action.

C) **close**

Just protest the same capability

**Note:** For directories and subprocess descriptors, the
open action just makes repeated reference to the
object more efficient.
Access Keys

Each access key is just a number represented in the directory by that number represented in its capabilities by that number.

Access keys may not be opened and closed.

Creation requires no additional parameters. Creation does not make an ownership entry. The access keys may not be destroyed.
Global Ecs object

A) Implementation
   a clist containing the capabilities for the objects
   and a file containing a unique name for each of the objects

B) Directory representation
   the index in the c-list of A along with the corresponding
   unique name in the file of A

C) Creation
   (will be used only by system routines)
   supply a capability for an ecs object (without destruction bit on)
   the capability will be placed in an empty slot in
   the c-list of A, a new unique name will be assigned to it
   and the result placed in the directory
   (This creates an ownership entry) (This permits destruction, which
   removes the object from the c-list of A and resets the unique name in the file of A)

D) Open
   returns the capability from the c-list of A, option bits
   added with the option bits of the current form of the object,
   (also unique name is checked)

E) Close
   (not applicable)
VI) **ecs-goodie** (do not remember new name)

A) **implementation**

   - A clist containing capabilities for open objects
     and a hash table, hosted on **ecs-goodie** unique name,
     pointing to corresponding C-list location and containing
     global open count.

B) **creation**

   Just produces a new **ecs-goodie** unique name which
   is placed in directory. **Does not make an ownership**
   **entry**.

C) **open**

   2 kinds of open

   1) open **ecs-goodie** with a presented **ecs object**

      f-return if the unique name of the **ecs-goodie** already
      in the hash table of A)

      error if the destruction is not on in the capability for the presented object
      otherwise, The unique name is found in the hash table,
      a free slot is found in the C-list and the capability
      for the presented **ecs object** is placed in the C-list.
      Also, the hash table entry is made to point to the
      slot in the C-list and the global open count becomes 1.

   2) open an **ecs-goodie**, presenting no **ecs object**.

      f-return if the unique name of the **ecs-goodie**
is not in the hash table
otherwise; bump the open counts as needed
and return the capability from the indicated
place in the c-list, option bits masked by the
option bits of the capability for the closed form of the object.

0) close

must present a capability for the closed form of the object.
If global count goes to zero, the entry is removed from
the hash table and the corresponding per object
is destroyed.