XEROX

BUSINESS SYSTEMS

Systems Development Division

June 9, 1978

To: ConComp Interest

From: Bob Ayers

Subject: Naming of Modules and DeSoto <==> Librarian

Stored: concompnaming.bravo

Currently:

DeSoto has one string name for the module in its master list, say the string "iris<pilot>20>streams.mesa" This string supplies both the content's remote file name (the whole string) and the local file name (using the ftp pruning algorithm -- henceforth the "fpa").

Librarian has three string names for a module. There is the libject name, say "PilotStreamsInterface" There is the libject version's local name, say "streams.mesa" And there is the libject version's remote name, say "iris<pilot>20>streams.mesa". Note well that the latter two names are associated with a particular version of the libject and may change from version to version.

Claim:

This is too many names. The rest of this memo is an attempt to arrange things so that names flow "automatically" from the objects and to suggest how Librarian snapshots fit into the concomp picture.

Proposed Usage:

A project gets a unique project prefix which is identical to its ifs account.

A project creates module libjects of the form <pilot>streams.mesa -- the project prefix plus the normal MESA module name. [This memo does not discuss non-module libjects.]

The project manager assigns libject's remote names to sub-directories as he pleases:

iris<pilot>20>streams.mesa. However the remote name differs from the libject name only in the sub-directory space.

The "local name" which is associated with a module version is always it's fpa-ed libject name (which is the same as it's fpa-ed remote name).

On a DeSoto/Librarian "CheckIn & StoreSource" the librarian takes the name of the module as supplied by DeSoto, removes any sub-directory information, forces the suffix to be ".mesa" and does a checkin on the libject with that name.

On a Desoto/Librarian "StoreObject" (which is really "Checkout & CheckIn & StoreObject") the librarian creates a libject name by the algorithm above, then replaces the ".mesa" with ".bcd" and does a checkout & checkin cycle on that libject.

Snapshots:

When a project wishes to make a "release", it has a list of modules (libjects) that comprise the release and a librarian's snapshot that defines the appropriate versions of all those modules (probably just the snapshot "current"). A "master-maker" tool takes a list of modules (previous master list?) and a snapshot and produces a master list with all of the modules' bcds listed as terminals and with their exact remote names, *including ftp versioning.*

This master list is then handed out to users. If they are engaged in maintaining or extending modules in the release, then they will edit their master list to remove the terminal-ness from "their" modules.

Minor Modifications to support the above:

DeSoto accepts a name of the form "iris<pilot>20>streams.bcd!5" for a terminal and then uses the indicated exact ftp version.

The Local Librarian strips off all <> prefix notation and any dot-suffix before showing a libject's name *in the graphical display.* A "display properties" continues to show the full name.

Operational Games:

The "StoreObject" algorithm described above stores, linearly, successive copies of a module's bcds. The only bcds of interest will be a) the latest one and b) the bcds that match releases and were therefore explicitly generated into master lists. It is the project's job, if it needs the space, to delete from ifs the bcds that are not of interest.