NAME

pool – Shore pools and anonymous objects

SYNOPSIS

```
VASResult       shore_vas::mkPool(
    const Path          name,
    mode_t              mode,
    lrid_t              *result
);

VASResult       shore_vas::rmPool(
    const Path          name
);
```

DESCRIPTION

Within a Shore file system, objects are located in **Storage Manager files** When objects are created, they are created "in" a file, and that file is chosen by the creator of the object. This allows an application to determine a certain amount of logical and physical clustering of the objects it creates. A **pool** is a Shore object that has an associated file. When a pool is created, an empty file is created. A pool cannot be destroyed unless it is empty.

We say that objects reside "in" the pool, although, in fact they reside in the Storage Manager file associated with the pool. (A pool is a registered object, so it resides with all other registered objects, in the Storage Manager file for registered objects.)

The objects in a pool are **anonymous.** They have no place in the naming hierarchy on the volume. They are identified only by their object identifiers. A pool can be **scanned** to find the object identifiers of the objects in the pool, and to inspect or update the objects in the pool.

Anonymous objects inherit some system properties from their pools, and have no need for others (such as the reference counts). Consequently, the storage overhead for anonymous objects is minimal. The properties inherited from the pool are: *uid, gid,* and *mode.* These properties of a pool determine the access controls on all objects in the pool.

ARGUMENTS

The argument *name* to both **mkPool** and **rmPool** is the path name of the pool. The argument *mode* to **mkPool** is the permission bits for the pool and all objects in the pool. The argument *result* for **mkPool** is an address in the caller’s address space. It may not be null. After creating the pool and its associated file, the SVAS returns the logical object identifier for the pool by writing into this location.

ENVIRONMENT

All operations on pools take place on the server.

These methods can be used only in a transaction.

ERRORS

Deadlocks can occur while locks are being acquired. See **transaction(svas)** for information about deadlocks.

A complete list of errors is in **errors(svas).**

VERSION

This manual page applies to Version 1.1 of the Shore software.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518.

COPYRIGHT

SEE ALSO

file_system(svas), anonymous(svas), registered(svas), unixfile(svas), sysprops(svas), transaction(svas), errors(svas) and **poolscan(svas).**