NAME

PoolScan – locate the objects in a Shore pool

SYNOPSIS

```
VASResult       shore_vas::openPoolScan(
      const lrid_t  &pool,
      Cookie        *cookie
);
VASResult       shore_vas::openPoolScan(
      const   Path   name,
      Cookie               *cookie
);

// get oid of next object in pool
VASResult       shore_vas::nextPoolScan(
      Cookie           *cookie,
      bool        *eof,    // TRUE if no result
                                  // if FALSE, result is legit
      lrid_t           *result,
      LockMode         lock=NL,
      SysProps         *sysprops = 0 // optional
);

VASResult       shore_vas::nextPoolScan(
      Cookie                 *cookie,
      bool             *eof,    // TRUE if no result
                                  // if FALSE, result is legit
      lrid_t           *result,     // snapped ref
      ObjectOffset     offset,
      ObjectSize       requested,  // -- could be WholeObject
      const vec_t      &buf,    // -- where to put it
      ObjectSize       *used = 0,   // - amount copied
      ObjectSize       *more = 0,   // requested amt not copied
      LockMode         lock=NL,
      SysProps         *sysprops = 0 // optional
);


VASResult       shore_vas::closePoolScan(
      const Cookie      &cookie
);
```

DESCRIPTION

OpenPoolScan establishes context for the caller (the *cookie*), in which the given pool is locked and scanned. A scan on a pool can be opened by naming the pool (by path) or by giving the pool's object identifier. **NextPoolScan** retrieves information about the next object in the pool, within the context of the open scan. **ClosePoolScan** closes the scan and discards all the context information.

ARGUMENTS

The argument *cookie* is a pointer into the caller's address space, from which the SVAS retrieves opaque context information on each operation, and into which the SVAS writes context information after each operation.

The argument *eof* is also a pointer into the caller's address space, where the SVAS writes the value TRUE (1) if the operation yielded no object, and FALSE (0) if the operation resulted in another object. If ∗*eof* is returned TRUE, the values of the other return arguments are meaningful.

The argument *result* points to an area in the caller's address space where the SVAS will write the logical object identifier for the next object in the pool. *result* may not be a null pointer. The value written into ∗*result* is the snapped form of the object's identifier.

The argument *sysprops* points to an area in the caller's address space where the SVAS will place the system properties of the object, if *sysprops* is a non-null pointer.

The argument *lock* can be used to increase (but not decrease) the strength of the lock acquired on the object when it is inspected by the SVAS.

The second form of **nextPoolScan** takes several more arguments, whose purpose is to make the method as general as **readObj.** They allow the object to be read in full or in part, by the scan.

ENVIRONMENT

These methods are available on the client and server. All these methods acquire locks, and so they can deadlock. TODO: need a note about what kinds of locks are acquired.

BUGS

A transaction can have no more than one pool scan open at any time. (TODO: extend SSM)

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

sysprops(svas), readObj(svas), mkVolRef(svas), errors(svas), and **transaction(svas).**