NAME

readObj – inspect a Shore object

SYNOPSIS

```
VASResult shore_vas::readObj(
            const  lrid_t     &obj,
            ObjectOffset      offset,
            ObjectSize        nbytes,
            LockMode          lock,
            const  vec_t      &data,
            ObjectSize        *used,
            ObjectSize        *more,
            lrid_t            *snapped=NULL,
            bool              *page_cached=NULL
        );
```

DESCRIPTION

ReadObj reads all or part of an object into the caller's address space.

ARGUMENTS

The argument *obj* is the full logical object identifier of the object to be read.

The argument *offset* indicates the first byte of the object to be read.

The argument *nbytes* indicates the number of bytes to be read. It can take the value **WholeObject,** in which case, the SVAS will read as much of the object into the given buffer. *Data* is a buffer in which to copy the contents of the object. If the buffer is too small for the requested data (client side and server side), or if the object is fragmented over several disk blocks (server side only), the the SVAS copies what it can into the buffer and updates the value to which the result argument *more* points, to reflect the number of bytes that did not get copied into the buffer. *Used* is an output argument that indicates the number of bytes of the buffer that were written by **readObj.**

The argument *snapped* is an (optional) output argument into which the SVAS writes the snapped logical record ID of the object.

The argument *lock* takes one of the values

```
enum LockMode { NL=0,IS=1,IX=2,S=3,SIX=4,U=5,X=6 },
```

which indicates what kind of a lock the SVAS will acquire when the object is read. A value below S is ignored; a share lock (S) is the minimum lock acquired. The call to **readObj** blocks until the lock is acquired or until a deadlock occurs.

The argument *page_cached* is an (optional) output argument into which the SVAS writes *true* if the read resulted in a page of small (anonymous) objects being cached. See **page(svas)** for information about how to use the page of cached objects.

ENVIRONMENT

ReadObj is available on both the server and clients. On the client side, if the object being read is a *small, anonymous object*, the entire page of small objects is copied to the client's address space, because all the objects on the page are anonymous objects in the same pool, and therefore have the same permissions. A share lock (S) is acquired on the page.

If a full page of objects is copied to the client's address space, the page is put into shared memory or is sent over a network connection, depending on the second argument used when the value-added server is constructed with **new_svas().** (The second argument determines the number of bytes of shared memory allocated for such a purpose.)

If a large object is read, pages of the object's data are shipped to the client in the same manner, depending on the third argument given to **new_svas().** (The third argument determines the number bytes of shared memory allocated for the purpose of transferring large objects.)

ReadObj must be called when a transaction is active.

ERRORS

The return values are valid only if **readObj** returns ST_OK.

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

sysprops(svas), writeObj(svas), truncObj(svas), appendObj(svas), lockObj(svas), errors(svas), new_svas(svas), and **transaction(svas).**