NAME

append_rec, create_file, create_id, create_rec, create_rec_id, destroy_file, destroy_rec, truncate_rec, update_rec, update_rec_hdr − Class ss_m Methods for File/Record Operations

SYNOPSIS

```
#include <sm_vas.h>  // which includes sm.h

static rc_t       create_file( vid_t          vid,
                               stid_t&        fid,
                               store_property_t    property
                               shpid_t        cluster_hint=0 // not used
                   ); // not used

static rc_t       destroy_file(const stid_t&  fid);


static rc_t       create_rec(const stid_t&    fid,
                               const vec_t&   hdr,
                               smsize_t       len_hint,
                               const vec_t&   data,
                               rid_t&         new_rid
                   );

static rc_t        destroy_rec(const rid_t&   rid);


static rc_t        update_rec(const rid_t&    rid,
                              smsize_t        start,
                              const vec_t&    data
           );

static rc_t        update_rec_hdr(const rid_t& rid,
                              smsize_t        start,
                              const vec_t&    hdr
           );
           // see also pin_i::update_rec*()

static rc_t        append_rec(const rid_t&    rid,
                              const vec_t&    data
           );


static rc_t        truncate_rec(const rid_t&  rid,
                              smsize_t        amount
               );
static rc_t        truncate_rec(const rid_t&  rid,
                              smsize_t        amount,
                              bool &          should_forward
               );
```

DESCRIPTION

These **ss_m** methods manipulate files and records.

Common Parameters

There are a number of common parameters for these methods:

vid volume ID of volume containing a file/record.

fid file ID (a.k.a., store ID) of a file.

rid record ID of a record.

data A vector pointing to data used to fill/overwrite the body of a record.

hdr A vector pointing to data used to fill/overwrite the header of a record.

create_file(vid, fid, property)

The **create_file** method creates a new file on the volume *vid,* and returns its store ID in *fid.* The *property* parameter specifies whether the file is temporary or not. See **enum(ssm)** for more information on file properties.

See the "ROOT INDEX METHODS" section of **volume(ssm)** for information on how to keep track of the files on a volume.

destroy_file(fid)

The **destroy_file** method destroys all records in the file and deallocates space used by a file. The space used by the file is not available for reuse until the transaction destroying the file commits.

create_rec(fid, hdr, len_hint, data, rid)

The **create_rec** method creates a record in a file. The ID of the new record is returned in the *rid* parameter. The *len_hint* parameter is a "hint" about the final length of the record. If the creation will be followed by appends, *len_hint* should ideally be set to the final length of the record. This will allow the SM to place the record in a location with sufficient contiguous space for the record. A value of 0 should be used if the final length is unknown. No order is defined on the records in a file: when a new record is created, the I/O subsystem may place the record anywhere in the file.

destroy_rec(rid)

The **destroy_rec** method destroys the specified record.

update_rec(rid, start, data)

The **update_rec** method updates a range of bytes in the body of the record specified by *rid.* The byte offset, from the beginning of the record body, for the beginning of the range is specified by the *start* parameter. The length of the range is the length of the *data* vector. The range is replaced by the bytes pointed to by the *data* parameter. Note that **update_rec** cannot be used to change the size of the record. It is an error to specify a starting location and vector length that imply updating beyond the end of the record.

update_rec_hdr(rid, start, hdr)

The **update_rec_hdr** method updates a range of bytes in the header of the record specified by *rid.* The byte offset, from the beginning of the header, for the beginning of the range is specified by the *start* parameter. The length of the range is the length of the *hdr* vector. The range is replaced by the bytes pointed to by the *hdr* parameter.

Note: There are no methods for appending to a record header or truncating a record header (as there are for a record body). If these methods would be useful for you, please contact the Shore developers.

append_rec(rid, data)

The **append_rec** method appends the bytes pointed to by *data* to the end of the record body.

truncate_rec(rid, amount) and **truncate_rec(rid, amount, should_forward)**

The **truncate_rec** method removes *amount* bytes from the end of a record body. If the record is one that was once large but is now small, and should be forwarded (replaced by a VAS, if it can arrange to do this), for the purpose of saving disk space, the *should_forward* Boolean is set to 'true'.

UNINITIALIZED DATA

The functions **create_rec, append_rec,** and **update_rec** can be used to write blocks of data that are all zeroes, with minimal logging. This is useful, for example, when a value-added server creates a record of a known size but with uninitialized data. To make use of this feature, these functions are called with data vectors of a specialized type, *zvec_t,* whose constructor takes only a size:

```
rc_t    rc;
char    h[HEADER_SIZE];
vec_t   hdr(h, sizeof(h));

// ... fill in hdr

// create a vector representing 1000
// contiuous bytes of zeroes
zvec_t  zdata(1000);

rc = ss_m::create_rec(fid, hdr,
    HEADER_SIZE + 1000, zdata, result);
```

ERRORS

All of the above methods return a **w_rc_t** error code. If an error occurs during a method that is updating persistent data (the create, update, append, and truncate methods will update data) then the record/file could be in an inconsistent state. The caller then has the choice of aborting the transaction or rolling back to the nearest save-point (see **transaction(ssm)**).

See **errors(ssm)** for more information on error handling.

EXAMPLES

To Do.

VERSION

This manual page applies to Version 2.0 of the Shore Storage Manager.

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. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

COPYRIGHT

SEE ALSO

vec_t(common), id(ssm), pin_i(ssm), scan_file_i(ssm), intro(ssm)