

**NAME**

append\_rec, create\_file, create\_id, create\_rec, create\_rec\_id, destroy\_file, destroy\_rec, lfid\_of\_lrid,  
truncate\_rec, update\_rec, update\_rec\_hdr – Class ss\_m Methods for File/Record Operations

**SYNOPSIS**

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

/* Logical-ID version */
static rc_t          create_file(const lvid_t& lvid,
                                serial_t&      lfid,
                                store_property_t property);

/* Physical-ID version */
static rc_t          create_file( vid_t          vid,
                                stid_t&         fid,
                                store_property_t property,
                                const serial_t&   logical_id = serial_t::null,
                                shpid_t          cluster_hint = 0); // not used

/* Logical-ID version */
static rc_t          destroy_file(const lvid_t&      lvid,
                                const serial_t&      lfid);

/* Physical-ID version */
static rc_t          destroy_file(const stid_t&      fid);

/* Logical-ID version */
static rc_t          create_rec(const lvid_t& lvid,
                                const serial_t& lfid,
                                const vec_t&    hdr,
                                smsize_t        len_hint,
                                const vec_t&    data,
                                serial_t&       lrid);

/* Logical-ID version */
static rc_t          create_id(const lvid_t& lvid,
                                int          id_count,
                                serial_t&    id_start);

/* Logical-ID version */
static rc_t          create_rec_id(const lvid_t& lvid,
                                const serial_t& lfid,
                                const vec_t&    hdr,
                                smsize_t        len_hint,
                                const vec_t&    data,
                                const serial_t& lrid);

/* Physical-ID version */
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,
        const serial_t&    serial = serial_t::null );

    /* Logical-ID version */
    static rc_t            destroy_rec(const lvid_t& lvid,
                                       const serial_t&      lrid);

    /* Physical-ID version */
    static rc_t            destroy_rec(const rid_t&  rid);

    /* Logical-ID version */
    static rc_t            update_rec(const lvid_t&  lvid,
                                       const serial_t&      lrid,
                                       smsize_t          start,
                                       const vec_t&      data);

    /* Physical-ID version */
    static rc_t            update_rec(const rid_t&  rid,
                                       smsize_t          start,
                                       const vec_t&      data);

    /* Logical-ID version */
    static rc_t            update_rec_hdr(const lvid_t&  lvid,
                                       const serial_t&      lrid,
                                       smsize_t          start,
                                       const vec_t&      hdr);

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

    /* Logical-ID version */
    static rc_t            append_rec(const lvid_t&  lvid,
                                       const serial_t&      lrid,
                                       const vec_t&      data);

    /* Physical-ID version */
    static rc_t            append_rec(const rid_t&  rid,
                                       const vec_t&      data,
                                       bool              allow_forward);

    /* Logical-ID version */
    static rc_t            truncate_rec(const lvid_t&  lvid,
                                       const serial_t&      lrid,

```

```

                                smsize_t          amount);

    /* Physical-ID version */
    static rc_t                  truncate_rec(const rid_t& rid,
                                smsize_t          amount);

    // lfids_of_lrid converts a logical record ID into a logical file ID
    /* Logical-ID version */
    static rc_t                  lfids_of_lrid(const lvid_t&      lvid,
                                const serial_t&      lrid,
                                serial_t&            lfids);

```

## DESCRIPTION

The above class **ss\_m** methods all deal with manipulating files and records. The logical-ID and physical-ID APIs have direct analogues, except when it comes to creating records. When using logical IDs, it is possible to pre-allocate logical IDs to apply to records upon creation of the records. For this, there is no counterpart in the physical-ID API.

### Common Parameters

There are a number of common parameters for these methods:

**lvid** Logical volume ID of volume containing a file/record.

**lfid** Logical file ID, the serial number of a file.

**lrid** Logical record ID, the serial number 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(lvid, lfids, property)

The **create\_file** method creates a new file on the volume *lvid*, and returns its serial number in *lfids*. 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(lvid, lfids)

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(lvid, lfids, hdr, len\_hint, data, lrid)

The **create\_rec** method creates a record in a file. The ID of the new record is returned in the *lrid* 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.

**create\_id(lvid, id\_count, id\_start)**

The **create\_id** method generates *id\_count* new IDs that can be used later by **create\_rec\_id** to associate a records with the IDs. The first ID is returned in *id\_start*. The other IDs should be obtained by calling **id\_start::increment(1)** *id\_count* -1 times.

**create\_rec\_id(lvid, lfid, hdr, len\_hint, data, lrld)**

The **create\_rec\_id** method is identical to **create\_rec** except that the record ID is specified by the caller with the *lrld* parameter rather than being generated and returned in *lrld* as is done in **create\_rec**.

**destroy\_rec(lvid, lrld)**

The **destroy\_rec** method destroys the specified record.

**update\_rec(lvid, lrld, start, data)**

The **update\_rec** method updates a range of bytes in the body of the record specified by *lvid*, *lrld*. 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(lvid, lrld, start, hdr)**

The **update\_rec\_hdr** method updates a range of bytes in the header of the record specified by *lvid*, *lrld*. 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(lvid, lrld, data)**

The **append\_rec** method appends the bytes pointed to by *data* to the end of the record body.

**truncate\_rec(lvid, lrld, amount)**

The **truncate\_rec** method removes *amount* bytes from the end of a record body.

**lfid\_of\_lrld(lvid, lrld, lfld)**

The **lfid\_of\_lrld** method returns, in *lfld*, the ID of file containing the record, *lrld*.

**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
// continuous bytes of zeroes
zvec_t     zdata(1000);

rc = ss_m::create_rec(lvid, lfid, 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

Copyright (c) 1994-1999, Computer Sciences Department, University of Wisconsin -- Madison. All Rights Reserved.

## SEE ALSO

**vec\_t(common)**, **pin\_i(ssm)**, **scan\_file\_i(ssm)**, **intro(ssm)**,