ID(SSM) ID(SSM)

NAME

id – Identifier Classes for Volumes, Indexes, Files, Pages and Records

```
SYNOPSIS
```

```
// Long Volume ID
// In common/
#include <lid_t.h>
struct lvid_t {
    w_base_t::uint4 high; // usually generated from net addr of creating server
    w_base_t::uint4 low; // usually generated from time of day when created
   lvid_t();
   lvid_t(w_base_t::uint4 hi, w_base_t::uint4 lo);
    static const lvid_t null;
};
// Volume handle
// In common/
#include <vid_t.h>
struct vid_t {
   vid_t() : vol(0) {}
    vid_t(w_base_t::uint2_t v) : vol(v) {}
   // Data Members
   uint2_t
            vol;
    static const vid_t null;
};
// Index and File Identifiers : Store ID
// in common/
#include <stid_t.h>
typedef w_base_t::uint4_t snum_t; // 4 bytes: store number
struct stid_t {
   vid_t vol;
   fil12
            filler; // vol is 2 bytes, store is now 4
    snum_t
    stid_t();
    stid_t(const stid_t& s);
    stid_t(vid_t vid, snum_t snum);
    static const stid_t null;
};
// Short Page ID
// in common/
#include <basics.h>
typedef w_base_t::uint4_t shpid_t; // 4 bytes
```

ID(SSM) ID(SSM)

```
// Long Page ID
// in sm/
#include <sm_s.h>
class lpid_t {
public:
              _stid;
    stid_t
    shpid_t
             page;
    lpid_t();
    lpid_t(const stid_t& s, shpid_t p);
    lpid_t(vid_t v, snum_t s, shpid_t p);
    operator bool() const;
                     const {return _stid.vol;}
    vid_t
             vol()
              store() const {return _stid.store;}
    const stid_t& stid() const {return _stid;}
    \ensuremath{//} necessary and sufficient conditions for
    // is_null() are determined by default constructor, q.v.
    bool
            is_null() const { return page == 0; }
    static const lpid_t bof;
    static const lpid_t eof;
    static const lpid_t null;
};
// Record ID
// in common/
#include <basics.h>
typedef w_base_t::int2_t slotid_t; // slot # for a record on a page: 2 bytes
// and in sm/
#include <sm_s.h>
class rid_t {
public:
    lpid_t
              pid;
    slotid_t
                slot;
             filler; // for initialization of last 2 unused bytes
    rid_t();
    rid_t(vid_t vid, const shrid_t& shrid);
    rid_t(const lpid_t& p, slotid_t s) : pid(p), slot(s) {};
    stid_t stid() const;
    static const rid_t null;
};
```

ID(SSM)

DESCRIPTION

Struct <code>lvid_t</code> represents a globally unique, 8-byte <code>long volume ID</code> . It is written on the volume header of each SSM volume. Upon mounting a volume, the volume is given a <code>local handle</code> . All objects on the volume are identified to the SSM by identifiers that contain the local handle, which is a <code>vid_t</code>.

Struct vid_t is the short handle for a mounted volume.

Struct **stid_t** is an identifier for an index or a file. It contains a short *volume handle and a store number*, **snum_t**.

The long page ID class, lpid_t identifies a page within a store, and it consists of a store ID and a short page ID, shpid_t.

A record ID comprises a long page ID and a slot number, which is of type slotid_t.

Thus, a record ID consists of a slot id, a page id, a store id and a volume handle.

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

lid(common)