LDB
an LDAP-like API for a database

Simo Sorce
Samba Team

idra@samba.org
simo.sorce@xsec.it
http://www.samba.org/~idra

What is LDB ?

- LDB is a database interface
- LDAP-like data model
  - support LDAP like search expressions
  - but it is schema-less
- Modular
  - available backends uses TDB or LDAP
  - modules stack over backend to provide extended functionality
- Very fast indexing (TDB Backend)

Once were TDB

- Samba is database driven internally
- SMBD process need a way to notify other process when certain events occur
- SMBD process also need to share data like locking tables
- TDB is a multiple-writer hash table that resembles Berkley DB
- In samba4 we noticed that a lot could be gained from better search and indexing capability

Why LDB ?

- TDB had a number of limitations
  - single key – single value mappings
  - every record is a binary object
  - no indexes, only a traverse function
  - programmers need to manually convert data structures to binary strings
  - programmers need to manually keep indexes if more than one index is needed
  - programmers need to manually check data endianess and handle structure upgrades

Why LDB ? (2)

- LDB has the advantages of an LDAP db
  - custom indexes
  - very powerful search strings
  - hierarchical
  - structures are easily modified or extended
- LDB has also the advantages of a TDB
- LDB will be used for persistent databases
- TDB will be kept for caches (like locking)
  - no index generation overhead

How is it implemented ?

- All the complexity of handling complex data in a TDB has been standardized and concealed behind an LDAP like API
- LDB takes care of building indexes for fast searches
  - when new indexes are added all the db is scanned automatically to rebuild them
- LDB does not need a schema
  - arbitrary attribute-value pairs can be stored in any object

Current Limitations

- Greatest limitations compared to LDAP:
  - no asynchronous calls
  - no paged results (this may be fixed shortly)
  - key must be representable as a NULL terminated string and can't contain comas or braces
  - not transactional, nor journaled
  - no pre/post indexes
- API limitations compared to TDB:
  - Explicit locking call
    - basic implementation for tdb backend
    - currently an error is returned with the ldap backend

LDB utilities

- LDB has a full set of user space utilities
  - ldbsearch
  - ldbadd
  - ldbdelete
  - ldbrename
  - ldbmodify
  - ldbedit
- Each command has a set of default switches:
  - mandatory:
    - -H ldb_url   choose the database (or $LDB_URL)

ldbsearch

```
An example: ldbsearch

$ ./bin/ldbsearch -H tdb://lib/ldb/test.ldb '(&(objectclass=organizationalUnit)
(ou=Groups))'
# returned 1 records
# record 1
dn: ou=Groups,o=Xsec,c=IT
objectclass: organizationalUnit
ou: Groups
```

- Syntax is quite similar to LDAP utilities
- The -H url defines the tdb (ldap server) to be used
- No authentication at this point, file permission define access controls

ldbedit

- ldbedit is very useful
  - it let you explore and change the database in a text editor
  - it uses well known ldif as representation format
  - you can use it to backup and restore databases
  - you can use the text editor you prefer
  - you can choose to use a filter to edit a subset of objects in the database
  - be careful when editing the objects with option -a, do not touch “internal” objects unless you know exactly what you are doing

speacial dns: @<something>

- dn names that start with an @ sign are special
  - the @ sign is used by reserved internal dn names
- you may set useful properties in these objects
  - indexes
    - the special dn @INDEXLIST controls indexing
  - case sensitivity
    - the special dn @ATTRIBUTES controls attributes behavior
  - class hierarchy
    - the special dn @SUBCLASSES is used to define subclasses
  - modules to be loaded
    - the special dn @MODULES set the list of modules to be loaded

LDB API

- The LDB API is clean and simple
  - ldb_connect
  - ldb_search
  - ldb_add
  - ldb_modify
  - ldb_delete
  - ldb_rename
  - ldb_errstring
- No close or free functions, talloc makes it

How to use LDB API

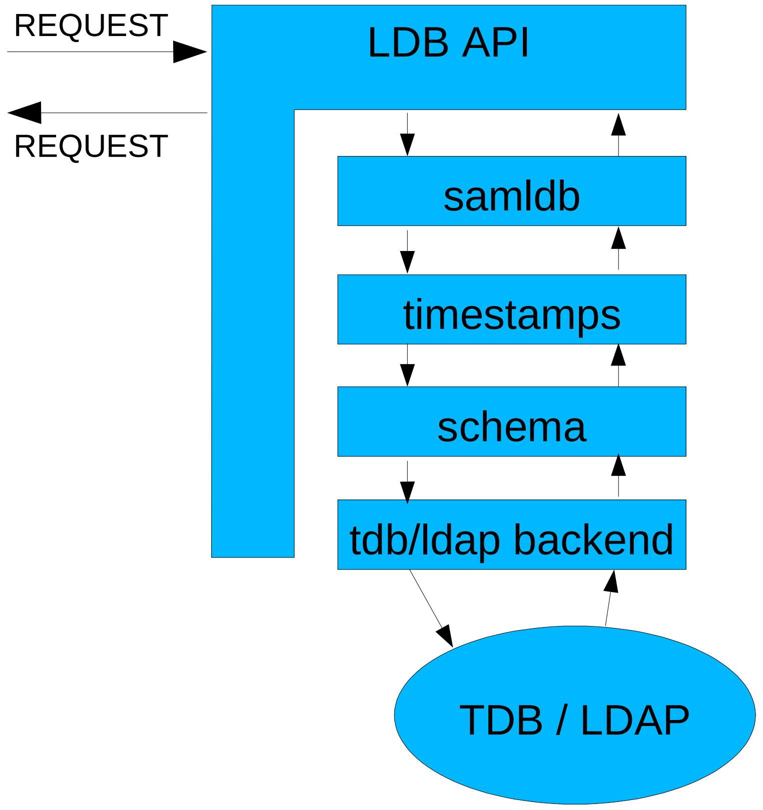
```
int count;
char *Sid;
const char * const *attrs = { “objectSid”, NULL };
struct ldb_message **res;

count = ldb_search(ldb_context, “dc=samba,dc=org”, LDB_SCOPE_SUBTREE,
“cn=Simo”, attrs, res);

Sid = talloc_strdup(mem_ctx, res[0]->elements[0].values[0].data);
```

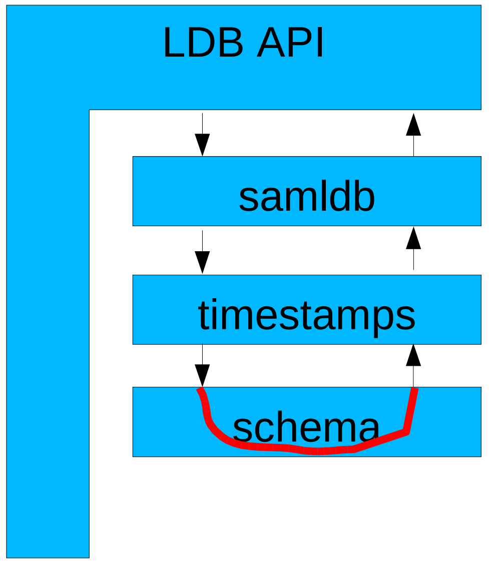
- The API is very similar to the LDAP API
  - On search you can specify complex filters and also which attributes you want back
  - you can also specify the base and scope of the search of course

What about extending LDB?

- Recently I extended the LDB code to support loading modules
  - modules can intercept any ldb api call
  - modules are stacked, each module call the next one
  - a backend (tdb, ldap) is just the last module that is called in the stack
  - modules can be loaded in the desired order (order often matters)
  - modules can be loaded automatically when opening an ldb file

modules stack





Schema module do not like the request. The request is not forwarded. An error is given back.

Available modules

- Currently 3 modules are available in samba4
  - timestamps
  - schema
  - samldb
- samldb is the most used module in samba4
  - handles all the user/group/machine adding operation
  - quasi-compatible with the way AD operate through the MS LDAP interface
  - automatically fills user/group objects with required attributes on creation

How to write a module ?

- as an example look at lib/ldb/modules/skel.c
- you must implement all the functions defined there
- functions may just call the next module or modify the data before the call

```
static const struct ldb_module_ops skel_ops = {
        "skel",
        skel_search,
        skel_add_record,
        skel_modify_record,
        skel_delete_record,
        skel_rename_record,
        skel_named_lock,
        skel_named_unlock,
        skel_errstring
};
```

writing a module

- modules are initialized when the ldb file is loaded
- you can set up private data structures
- never use static data, keep in mind that modules should be reentrant (ex: the samldb module calls ldb_search while ldb_add is in progress)
- during initialization you should set up a destructor if you need to clean up on close (ex: to close files, close sockets, free structures, etc...)

Loading modules

- How to make a module available to ldb once you made one?
  - currently you need to modify ldb_modules.c
  - ASAP we will have a dynamic loader that will be able to load .so objects
- How to activate a specific module on an ldb?
  - through -o modules:modname,2nd,etc.. option
  - through the @MODULES special dn
    - @LIST: samldb,timestamps,schema,...

LDAP server in samba4 ?

- AD is not a standards compliant LDAP
- openLdap may be changed to follow AD
  - I made an experimental ldb backend for openLdap
  - Need to create overlays to cope with AD
- we used LDB to make our own LDAP
  - an experimental not complete LDAP server is available
  - basic schema LDB module (very experimental)
  - basic rootDse available
  - no authentication available

What is LDB used for in samba4 ?

- The primary usage is for the new SAM
- Samba4 is going to be 100% compatible with an Active Directory Domain Controller
  - LDB is a good solution to have an LDAP like user database
  - we can better interoperate with AD by keeping a similar data structure
- There are also other databases like secrets.ldb
- It may be used to store samba4 configuration instead of using a text file like the current smb.conf

Using LDB

- Can I use it ?
  - The Samba Team encourages people to use LDB in their own projects
- Where can I find it?
  - Currently it is available only by downloading the samba4 source code
- Do I need to build and install samba4 to use it?
  - No, you can build LDB alone

Requisites

- What libraries does LDB depends on ?
  - libc
  - tdb
  - talloc
  - ldap libraries if you want to build the ldap backend
- What kernel/OS can I use it on ?
  - most of our test has been done on linux kernel 2.4/2.6
  - tdb needs well working locking (don't use it on nfs)
  - Samba Team take care of making things portable on most Posix operating systems

Licenses ?

- My Project has a Funny License, can I use LDB with it?
- Unlike the rest of the code in samba, LDB uses the GNU LGPL license instead of the GNU GPLv2
- This make it possible to:
  - use LDB in any GPL licensed program
  - use LDB with any other free software licensed program
    - note: currently the talloc library is GPLed bu we are available to talk about changing it's license to LGPL if that blocks the adoption of LDB by other OpenSource projects

References

- Source
  - samba4 source code:
    - svn co svn://svnanon.samba.org/samba/branches/SAMBA_4_0 samba4
  - tdb fork on sourceforge.net:
    - http://sourceforge.net/projects/tdb
- Developer resources
  - Mailing List:
    - samba-technical@samba.org
  - IRC Channel:
    - #samba-technical on freenode.net

Questions ?