OpenLDAP Development

Back-config –Configuration Backend

Howard Chu  hyc@symas.com
ODD/Wien July 18, 2003
Objectives

• Support runtime reconfiguration without requiring server restarts
  – Allow ACL reconfiguration
  – Allow schema modification

• Support remote administration of slapd
  – Enable performing all configuration via LDAP
Rationale

• The objectives are not mutually assured:
  – Could e.g. use SIGHUP to force reread of config file, thus allowing runtime changes, but not allowing remote administration
  – Could provide LDAP interface to rewrite config file, without any mechanism for slapd to reload the changed configuration

• Fulfilling both objectives is desirable
• Either one may require significant effort
Runtime Reconfiguration

• Preliminary support embodied in Gentle HUP processing:
  – Aimed at allowing a new slapd instance to be started with minimal impact on existing sessions
  – The new slapd instance can use the same BDB database as the old, or can use a separate database
Gentle HUP, cont’d

- Implementation is awkward at best
  - Requires descriptor-passing to avoid session interruption
  - Database sharing requires back-bdb and shared mutex support
- Some benefits from starting a new instance
  - New executables can be installed with minimal service impact
  - Can temporarily recover from memory leaks
Runtime Constraints

• Config processing is currently single-threaded
  – Config file is processed before threads are spawned
  – Config data is not mutex protected
  – Adding mutexes may harm overall performance
Ensuring Config Consistency

- Use a single rdwr lock for access to global variables
  - Highly invasive code change, requires locking in many places
  - Doesn’t ensure consistency within the life of an operation
- Disable the thread pool
  - Wait for all executing operations to complete
  - Prevent new operations from being dispatched until config changes are processed
Remote Administration

• Varying degrees of “LDAP enablement” possible
  – Expose slapd.conf as generic text attributes, with no semantic awareness
  – Map coarse set of objects onto slapd.conf, minimal semantic awareness
  – Replace slapd.conf with LDIF/attribute-based format

• Each approach has tradeoffs
Slapd.conf as generic text

• Implementation is fairly trivial
  – Models already exist (e.g. back-passwd) for using flat text files as backends.
  – Has no impact on current config processing code

• Major disadvantages
  – Very difficult to support runtime reconfig
  – Ignores “include” directives
  – Makes it too easy to shoot yourself in the foot
Slapd.conf with partial semantics

- Targets specific functionality with explicit attributes, leaves remainder as generic text
  - Handle include, access, and schema keywords
  - Optionally handle database keywords as separate objects
- Drawbacks
  - Loses config file comments
  - Still requires some changes to existing config parsing code
Slapd.conf as LDIF

- Provides the most client-friendly support
  - Defines schema for all existing config functionality
- Requires extensive changes in slapd
  - Config parsing must be completely rewritten for slapd and all backends
    - Needs to be table-driven
    - Needs OID allocation methodology, etc.
  - Requires support for per-backend schema to avoid config syntax clashes
Which is best?

• Using generic text precludes changes taking effect immediately
• Supporting a small set of keywords provides some essential features now, others later/never
• Migrating to LDIF requires major overhauling of slapd
Conclusions

• The pure generic text solution is not useful enough
• The full LDIF solution is taking too much effort to complete
• Will probably fall back to partial support
• Open to suggestions and assistance!