PADL SOFTWARE

Extending OpenLDAP

Luke Howard <lukeh@padl.com>
PADL Software Pty Ltd

Why extend OpenLDAP?

- Custom backends
- OpenLDAP is a general purpose directory
  - Application or usage-specific extensions are unlikely to be included in OpenLDAP itself
  - Code forks are expensive
  - *slapd*(8) is often the wrong place for extensions
- Plug-ins!

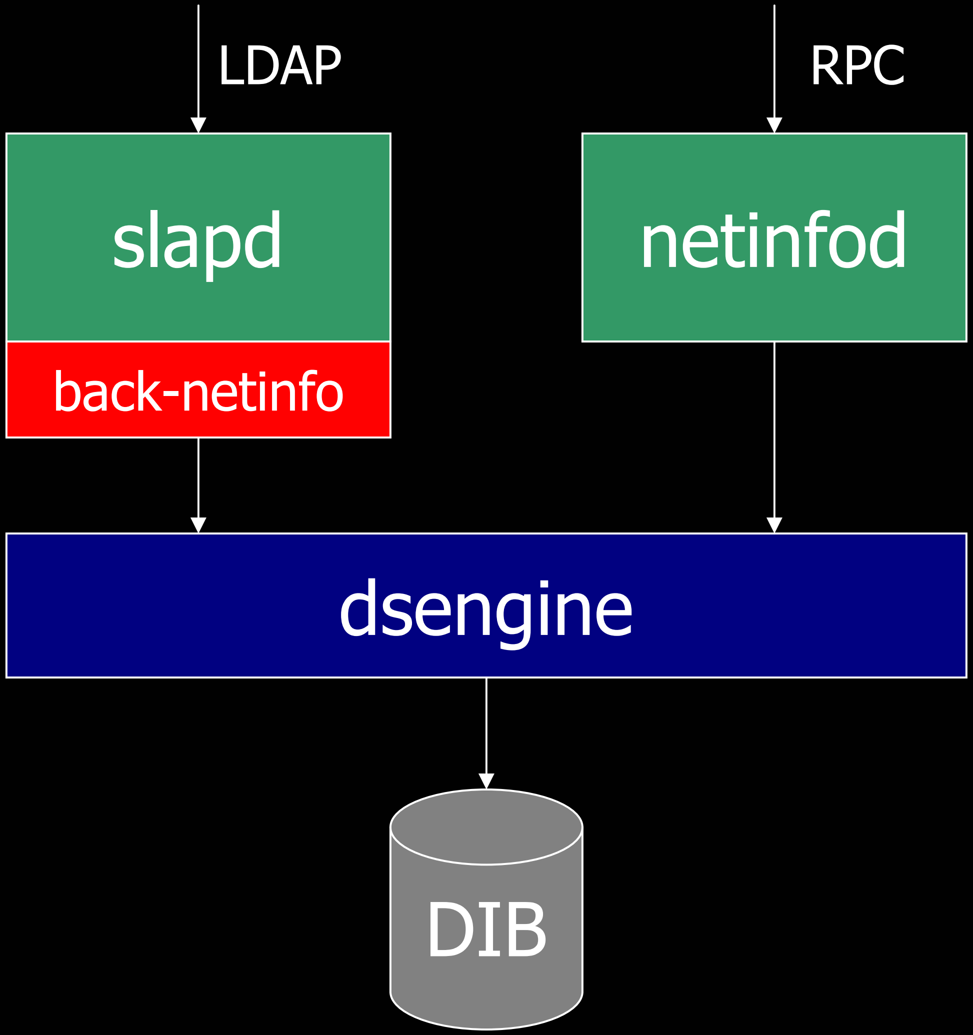
Writing a backend

- Native plug-in or statically linked
- Implement LDAP operations:
  - Search
  - Modify
  - Add
  - *etc*
- Perfectly valid to return `LDAP_UNWILLING_TO_PERFORM`

Example: NetInfo Bridge

- Bridge between NetInfo and LDAP
- NetInfo differs to X.500/LDAP:
  - information and naming model
  - authorisation model
  - schema
- Bridge must thus map these
- Mac OS X 10.2 shipped bridge using OpenLDAP 2.1 (Open Directory)

NetInfo Bridge: Architecture



OpenLDAP native plug-ins

- Specific to OpenLDAP
- Complete access to *slapd*(8) internals
- API subject to change
- Required for certain extensions:
  - Syntaxes
  - Matching rules
- Limited hooks into operation flow

Writing a native plug-in

- Modules must implement:
  `int init_module(int argc, char *argv[]);`
- Initialisation function can call internal slapd API, e.g:
  - `register_syntax()`
- Private headers required to build
- Plugins are configured in `slapd.conf`:
  `moduleload libfoo-plugin.so`

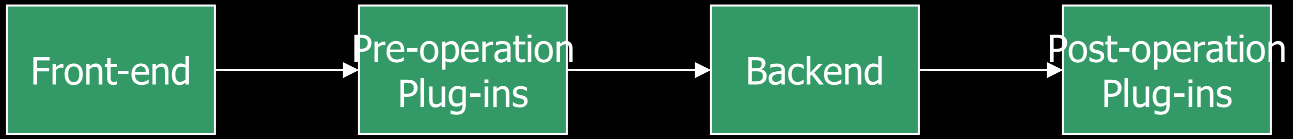
SLAPI: A History

- Directory server plug-in API
- Introduced with Netscape DS 3.x
- Implemented by IBM in SecureWay/IDS
- Sun enhanced DS 5.x SLAPI:
  - computed attributes
  - object extensions
- IBM ported IDS SLAPI to OpenLDAP
- PADL adding support for DS 5.x API

SLAPI Plug-in Types

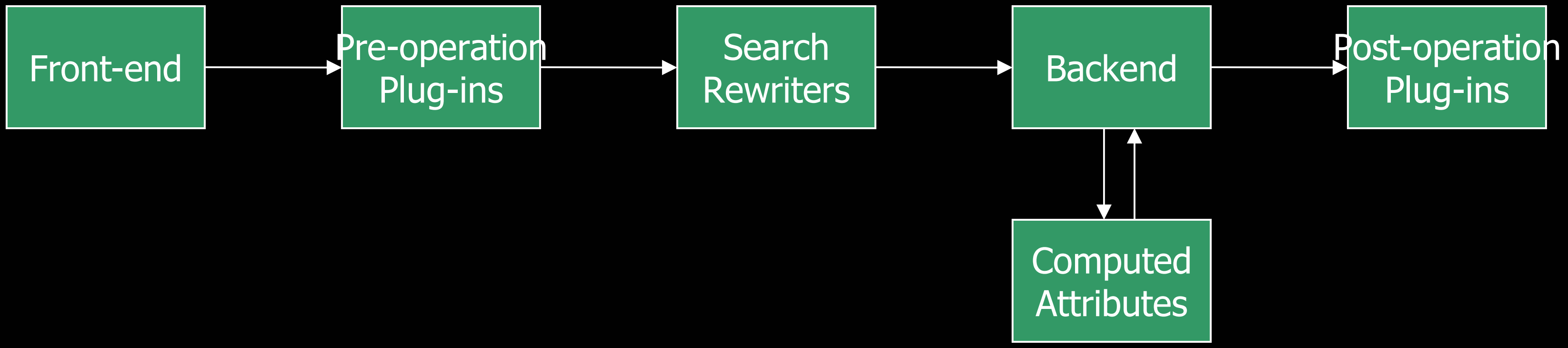
- Operation
  - Pre-operation (modify, add, bind, *etc*)
  - Post-operation (modify, add, bind, *etc*)
  - Extended
- Object
  - Computed attribute
  - Controls
  - Search rewriter

Operation Plug-in Flow



- Pre-operation plug-ins may abort an operation
- If so, they are responsible for sending the LDAP result to the client
- Post-operation plug-ins are called after backend
- Result will have been sent to client

Search (Object) Plug-in Flow



- Search rewriters: rewrite the search filter (useful for searching on computed attributes)
- Computed attributes: plug-ins are called by `send_search_entry()`

SLAPI API

- Most plug-ins have the prototype:
  - `int myPluginFunc(Slapi_PBlock *);`
- A ‘pblock’ is a parameter dictionary:
  - Operation
  - Connection
  - Modifications
  - Entry being added
- Computed attributes: different API

Controls & Extended Operations

- `slapi_register_supported_control()`
- Use `slapi_control_present()` to check for control in operation plug-in
- Controls are stored in `SLAPI_REQCONTROLS` parameter
- Extended operation is another plug-in type
- Check for extended operation OID

Object Extensions

- Associate arbitrary state with an object:
  - Connection
  - Operation
- Useful for passing state from pre-operation to post-operation plug-ins
- Plug-in registers constructor, destructor
  `slapi_register_object_extension()`
- SLAPI returns object extension handle
  `slapi_{get,set}_object_extension()`

Plug-in configuration

- `slapd.conf`
- Plug-ins are invoked in the order specified in the configuration file
- Syntax:

```
plugin <type> <path> <init_fn>
```

- Example:

```
plugin preoperation libfoo-plugin.so \
foo_preop_init
```

Sample: pre-operation (1)

```
int abort_on_kurt_preop_add(Slapi_PBlock *pb)
{
    Slapi_Entry *e;
    char *uid;
    int rc = 0;

    slapi_pblock_get(pb, SLAPI_ADD_ENTRY, &e);
    uid = slapi_entry_attr_get_charptr(e, “uid”);
    if (uid != NULL) {
        if (strcasecmp(uid, “kurt”) == 0) {
            slapi_send_ldap_result(
                pb, LDAP_CONSTRAINT_VIOLATION,
                NULL, “Sorry, Kurt!”, 0, NULL);
            rc = -1; /* abort operation */
        }
        slapi_ch_free_string(&uid);
    }
    return rc;
}
```

Sample: pre-operation (2)

```
int default_drink_preop_add(Slapi_PBlock *pb)
{
   Slapi_Entry *e;

   slapi_pblock_get(pb, SLAPI_ADD_ENTRY, &e);

   /* set default value for favouriteDrink attribute */
   slapi_entry_attr_set_charptr(e, “favouriteDrink”,
       “Schnaps”);

   return 0;
}

int default_drink_preop_init(Slapi_PBlock *pb)
{
   return slapi_pblock_set(pb, SLAPI_PLUGIN_PRE_ADD_FN,
   (void *)default_drink_preop_add);
}
```

Sample: computed attribute

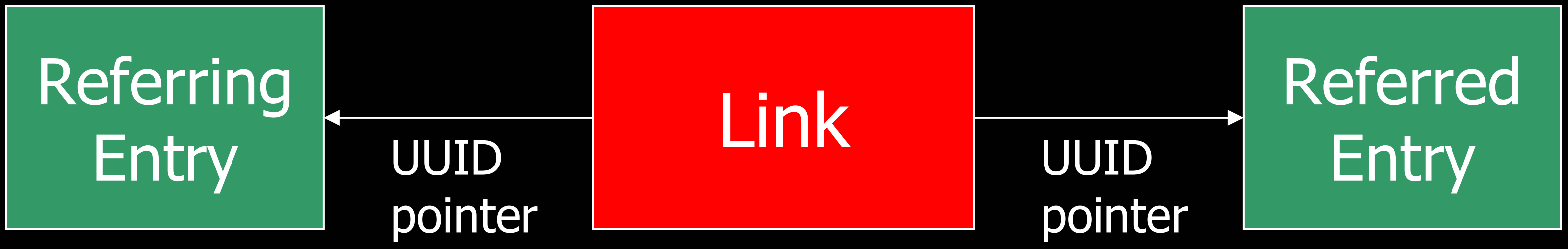
```
int favourite_food_compute(computed_attr_context *c,
   char *type, Slapi_Entry *e, slapi_compute_output_fn
   outputfn)
{
   int rc = -1; /* no attribute sent */
   if (strcasecmp(type, “favouriteFood”) == 0) {
        Slapi_Value *value;
        Slapi_Attr *attr;

        attr = slapi_attr_init(slapi_attr_new(),
                               “favouriteFood”);
        value = slapi_value_new_string(“Schnitzel”);
        slapi_attr_add_value(attr, value);
        rc = (*outputfn)(c, attr, e);
        slapi_value_free(&value);
        slapi_attr_free(&attr);
   }
   return rc;
}
```

LinkEngine: Overview

- Complex SLAPI plug-in:
  - Pre-operation, post-operation
  - Computed attributes
- Distributed referential integrity service
- Assumes responsibility from backend for managing DN references
- Links are made by UUID; DNs may change
- Back-links

LinkEngine: Architecture



- Each linked attribute has an integer ID
- Even/odd ID = forward link/back-link
  - e.g. member (2) / memberOf (3)
- Links are stored in private DIT
- Proxy objects cache remote DSA references

LinkEngine: Example

- Group entry (forward link)

```
dn: cn=OpenLDAP,cn=Users,dc=padl,dc=com
objectClass: Group
member: cn=Luke Howard,cn=Users,dc=padl,dc=com
```

- User entry (back-link)

```
dn: cn=Luke Howard,cn=Users,dc=padl,dc=com
objectClass: User
memberOf: cn=OpenLDAP,cn=Users,dc=padl,dc=com
```

- member/memberOf are computed at search time

LinkEngine: Implementation

- Pre-operation plug-in:
  - “Captures” updates on linked attributes
  - Back-end will not see them
  - Does pre-commit check
- Post-operation plug-in:
  - Commits linked attributes to DIB
- Computed attribute plug-in:
  - Activates linked attributes from links (resolves UUID to DNs)

OpenLDAP SLAPI Extensions

- Denoted by `SLAPI_X_`*`XXX`*
- `slapi_x_filter_append()`
- `slapi_x_compute_get_pblock()`
- `SLAPI_X_CONN_CLIENTPATH`
- `SLAPI_X_CONN_SERVERPATH`
- `SLAPI_X_CONN_IS_UDP`
- `SLAPI_X_CONN_SSF`

Conclusion

- OpenLDAP is extensible
- Backend API
- Native plug-ins and SLAPI are complementary at this time
- Expect native plug-in API to evolve over time
- Use SLAPI for operation notifications, computed attributes, or portability

Further information

- PADL Software
  http://www.padl.com/
- OpenLDAP
  http://www.openldap.org/
- Apple Open Directory
  http://developer.apple.com/darwin/projects/opendirectory/
  http://www.padl.com/Articles/AdvancedOpenDirectoryConf.html
- SLAPI
  http://docs.sun.com/db/doc/816-6701-10
  http://docs.sun.com/db/doc/816-6702-10