

Contents

|                           |          |
|---------------------------|----------|
| What is Net::LDAP?.....   | Slide 2  |
| Connecting .....          | Slide 3  |
| Authentication .....      | Slide 4  |
| Return Values .....       | Slide 5  |
| Searching .....           | Slide 6  |
| Entry Object.....         | Slide 7  |
| Entry Object — 2.....     | Slide 8  |
| Displaying an Entry ..... | Slide 9  |
| Limit Returns .....       | Slide 10 |
| Adding New Entries .....  | Slide 11 |
| Adding Entries.....       | Slide 12 |
| Deleting an Entry.....    | Slide 13 |
| Modifying an Entry.....   | Slide 14 |
| Modify — add.....         | Slide 15 |
| Modify — delete.....      | Slide 16 |
| Modify — replace .....    | Slide 17 |
| Using Start TLS.....      | Slide 18 |
| References .....          | Slide 19 |

1Introduction to Net::LDAP21

Programming LDAP with Perl

Net::LDAP

Nick Urbanik <nicku@nicku.org>

Copyright Conditions: Open Publication License

(see <http://www.opencontent.org/openpub/>)

A computing department

SNM — ver. 1.3

Network Directories and their Structure - p. 1119

What is Net::LDAP?

- Mature and fully-featured Perl library
- Pure Perl; very easy to install on any platform
- On Windows, do
  - D:\>**ppm**
  - PPM - Programmer's Package Manager version 3.1.
  - Copyright (c) 2001 ActiveState SRL. All Rights Reserved.
  - ...
  - ppm> **install perl-ldap**
  - On other platforms, do:
    - \$ **sudo perl -MCPAN -e 'install Net::LDAP'**
- Excellent documentation
- Start with
  - \$ **perldoc Net::LDAP**
- Helpful mailing list

SNM — ver. 1.3

Network Directories and their Structure - p. 2119

Connecting

- Connect when construct the Net::LDAP object:

```
my $ldap = Net::LDAP->new( $hostname )
    or die "Unable to connect to $hostname: $!";
```
- See perldoc Net::LDAP for many other parameters you can pass in constructor

1Net::LDAP Operations31

Authentication

- The *bind* operation
- Three types: *anonymous*, *simple*, *SASL*
- Anonymous:

```
my $result = $ldap->bind;
```
- Simple:

```
my $result =
    $ldap->bind( $dn, password => $password );
```
- Danger! Password sent in clear text unless use TLS (see slide \$18)

SNM — ver. 1.3

Network Directories and their Structure - p. 419

Return Values

- Most Net::LDAP methods return an object
- returned object provides method to obtain results of operation
  - result code returned by `$result->code`
  - error message returned by `$result->error`
- Example:

```
warn $result->error if $result->code;
```

Network Directories and their Structure - p. 3119

SNM — ver. 1.3

Network Directories and their Structure - p. 5119

## Searching

- Need three things for a search:
    - search base**, **scope** and **filter**
- ```
my $result = $ldap->search(  
    base => 'dc=tyict,dc=vtc,dc=edu,dc=hk',  
    scope => 'sub',  
    filter => '(uid=nicku)'  
);  
die $result->error if $result->code;
```
- The result also contains the matching entries:

```
foreach my $entry ( $result->entries ) {  
    $entry->dump;  
}
```
  - Methods of the object that results from a search documented in `perldoc Net::LDAP::Search`

SNM — ver. 1.3

Network Directories and their Structure - p. 6/19

## Entry Object

- Entry object is used:
  - to create new entries and
  - is available from a search
- Documented in `perldoc Net::LDAP::Entry`
- Methods:
  - dn** returns the DN for the entry:

```
my $dn = $entry->dn;
```
  - exists** tests if an attribute exists in the entry:

```
do_something() if $entry->exists( 'cn' );
```

SNM — ver. 1.3

Network Directories and their Structure - p. 7/19

## Entry Object — 2

- Methods:
  - get\_value** obtain the value(s) for an attribute in the entry

```
my $value = $entry->get_value( 'cn' );
```
- Multivalued attributes:** Some attributes have more than one value. For these, **get\_value** returns the first value in a scalar context, and all of them in a list context:

```
my $first = $entry->get_value( 'objectClass' );  
my @values = $entry->get_value( 'objectClass' );
```
- attributes** returns a list of attributes the entry contains

```
my @attrs = $entry->attributes;
```

SNM — ver. 1.3

Network Directories and their Structure - p. 8/19

## Displaying an Entry

- If all attributes can be printed, then this function could display an entry:

```
sub display_entry {  
    my $entry = shift;  
    my @attrs = $entry->attributes;  
  
    foreach my $attr ( @attrs ) {  
        my @value = $entry->get_value( $attr );  
  
        foreach my $value ( @value ) {  
            print "$attr: $value\n";  
        }  
    }  
}
```

SNM — ver. 1.3

Network Directories and their Structure - p. 9/19

## Controlling What's Returned

- By default, LDAP server returns attributes and their values for each entry.
- Can ask server for just the types; then value returned for each attribute is empty:

```
my $r = $ldap->search(  
    base => 'dc=tyict,dc=vtc,dc=edu,dc=hk',  
    filter => '(cn=Nick*)',  
    typesonly => 1,  
);
```
- Access control limits what attributes are returned; can limit further by specifying a list of required attributes:

```
my $r = $ldap->search(  
    base => 'dc=tyict,dc=vtc,dc=edu,dc=hk',  
    filter => '(cn=Nick*)',  
    attrs => [ qw(uid cn) ],  
);
```

SNM — ver. 1.3

Network Directories and their Structure - p. 10/19

## Adding New Entries

- Net::LDAP supports four ways of adding new entries to a directory:
  - the **add** method;
  - the **Entry** class;
  - LDIF**: Same as adding with the **Entry** class, except **Entry** is read from a file via the **LDIF** module
  - DSML**: Same as adding with the **Entry** class, except **Entry** is read from a file via the **DSML** module

SNM — ver. 1.3

Network Directories and their Structure - p. 11/19

## Adding Entries

- Pass an array reference of attribute and value pairs to the **add** method:

```
my $r = $ldap->add( $dn,  
    attrs => [  
        cn => 'HP5000-A204e',  
        objectClass => [ qw(device ieee802Device/ ) ],  
        description => 'Printer in A204e',  
    ],  
);
```
- ... or, create an **Entry** object and call the **update** method:

```
my $dn = 'ou=devices,dc=tyict,dc=vtc,dc=edu,dc=hk';  
my $entry = Net::LDAP::Entry->new;  
$entry->dn( $dn );  
$entry->add( cn => 'HP5000-A204e' );  
$entry->add(  
    objectClass => 'device',  
    description => 'Printer in A204e',  
);  
$msg = $entry->update( $ldap );
```

SNM — ver. 1.3

Network Directories and their Structure - p. 12/19

## Deleting an Entry

- Can delete an entry by passing a DN:

```
my $dn = 'ou=dev,dc=tyict,dc=vtc,dc=edu,dc=hk';  
my $r = $ldap->delete( $dn );
```
- ... or like many **Net::LDAP** methods, you can pass an entry where a DN is expected:

```
$entry = find_entry_to_delete();  
$r = $ldap->delete( $entry );
```

SNM — ver. 1.3

Network Directories and their Structure - p. 13/19

## Modifying an Entry

- **modify** operation has four sub-operations:
  - add**
    - add new attributes
    - add values to existing multivalued attributes
  - delete**
    - delete whole attributes
    - delete values from within existing attributes
  - replace** replace attributes or add if necessary
  - moddn** rename an entry under same or different parent

SNM — ver. 1.3

Network Directories and their Structure - p. 14/19

## Modify — add

- Add a new attribute, or a new value to an existing multi-valued attribute:

```
$r = $ldap->modify( $dn,  
    add => {  
        mail => 'nicku@nicku.org'  
    }  
);
```
- An error is returned if:
  - the attribute exists and is not multi-valued;
  - the attribute exists and is multi-valued and the value already exists;
  - the schema does not allow the attribute.

SNM — ver. 1.3

Network Directories and their Structure - p. 15/19

## Modify — delete

- To delete all instances of the attribute in the entry:

```
$r = $ldap->modify( $dn,  
    delete => [ 'mail' ]  
);
```
- You can delete specific values:

```
$r = $ldap->modify( $dn,  
    delete => { 'mail' => [ 'nicku@abc.com' ] }  
);
```

SNM — ver. 1.3

Network Directories and their Structure - p. 16/19

## Modify — replace

- Replace whole attributes:

```
$r = $ldap->modify( $dn,  
    replace => { 'mail' => 'nicku@xyz.com' }  
);
```
- Multi-valued:

```
$r = $ldap->modify( $dn,  
    replace => {  
        'mail' => [ qw(nicku@xyz.com  
                        nick@iohk.com) ]  
    }  
);
```

SNM — ver. 1.3

Network Directories and their Structure - p. 17/19

## Using Start TLS

- LDAPv3 supports the *Start TLS* extension
- Allows a client to request that the server begin encrypting traffic with client
- Essential when using simple authentication; avoid password being sent in clear text over the network
- Here is the simplest use, where there is no requirement to store local copies of the certificates, but the identity of the server is not checked:

```
my $r = $ldap->start_tls( verify => 'none' );
```
- See `perldoc Net::LDAP` and `perldoc Net::LDAP::Security` for details and examples.

SNM — ver. 1.3

Network Directories and their Structure - p. 18/19

## References

- See the excellent documentation with Net::LDAP:

|                                 |                      |
|---------------------------------|----------------------|
| Net::LDAP                       | Net::LDAP::FAQ       |
| Net::LDAP::Constant             | Net::LDAP::Filter    |
| Net::LDAP::Control              | Net::LDAP::API       |
| Net::LDAP::Control::Paged       | Net::LDAP::LDIF      |
| Net::LDAP::Control::ProxyAuth   | Net::LDAP::Message   |
| Net::LDAP::Control::Sort        | Net::LDAP::Reference |
| Net::LDAP::Control::SortResult  | Net::LDAP::RFC       |
| Net::LDAP::Control::VLV         | Net::LDAP::RootDSE   |
| Net::LDAP::Control::VLVResponse | Net::LDAP::S         |
| Net::LDAP::DSML                 | Net::LDAP::Schema    |
| Net::LDAP::Entry                | Net::LDAP::Search    |
| Net::LDAP::Examples             | Net::LDAP::Security  |
| Net::LDAP::Extra                | Net::LDAP::Util      |
- See the web site for Net::LDAP: <http://ldap.perl.org/>
- Graham Barr wrote slides on which these notes are based:  
<http://ldap.perl.org/perl-ldap-oscon2001.pdf>
- David N. Blank-Edelman, *Perl for System Administration*, O'Reilly, July 2000, ISBN: 1565926099.
- Gerald Carter, *LDAP System Administration*, O'Reilly, March 2003, ISBN: 1565924916.
- Clayton Donley, *LDAP Programming, Management and Integration*, Manning, 2003, ISBN: 1590140405.

Network Directories and their Structure - p. 19/19