

Python and LDAP

CRUD: Create, Read, Update, and Delete with Python-LDAP

Skill Level: Intermediate

[Noah Gift](#)

Software Engineer
Giftcs

[Grig Gheorghiu \(grig@gheorghiu.net\)](mailto:grig@gheorghiu.net)

Director of Technology
RIS Technology

28 Oct 2008

At some point in their careers, most systems administrators need to interact with an LDAP server. This article shows how LDAP can be used for Apache authentication, as well as how to perform CRUD, or Create, Read, Update, and Delete operations on an OpenLDAP database, using the Python module `python-ldap`.

Introduction

In this article, we show how to install an instance of OpenLDAP running on an Amazon EC2 virtual machine, set up Apache/LDAP authentication, and then use Python to perform CRUD, or Create, Read, Update and Delete operations. It is important to note that LDAP can be installed on Fedora, Ubuntu, Red Hat, AIX®, and more. For the purpose of this article, though, we have decided to focus on Amazon EC2 virtual machines. You can follow along at home on any Linux® distribution, or whatever environment that you have handy. Finally, we cover a lot of code and complicated techniques in the articles. You may want to download the [sample code](#) from the very beginning, and keep it handy while going through the article.

Programmatically controlling LDAP is often associated with sysadmin-related work, and so it should not be a surprise that a library exists for working with LDAP from

Python. The python-ldap module has been around for quite some time, and there are links to official documentation located in the [resources](#) section.

We assume that you are familiar with general LDAP concepts such as directory schema, Distinguished Names (DN), Common Names (CN), and the concepts of filters and attributes. This article is not an LDAP tutorial; we prefer to talk less about theory and more about practical examples of using and administering an LDAP database.

What is LDAP and what is it used for?

So what is LDAP, anyway? By strict definition, the term LDAP stands for Lightweight Directory Access Protocol. The name has become synonymous with an actual directory architecture, though. Usually when someone mentions LDAP, he is referring not to a protocol, but to a directory service.

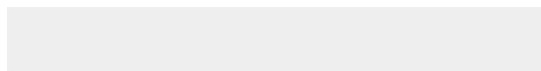
The most current version of LDAP is V3, or version 3. LDAP was designed to be a general-purpose directory, but there are a few conventions. A record consists of a DN, and one or more attributes that are defined in attribute definitions. LDAP databases have a schema that is much different from a typical Relational Database schema definition language. One example of this would be that a typical Relational Database is table based, while LDAP incorporates inheritance. If you would like to know more about general LDAP theory, we highly recommend reading the OpenLDAP book listed in the [resources](#) section.

Now to the question about what it is used for. Almost exclusively, it is used for authentication for people who build IT infrastructures. It plays nicely with Samba as well, so an experienced sysadmin can set up a very sophisticated IT infrastructure with zero cost, unlike other proprietary directory solutions. To be clear, "authentication" simply means that all the machines in the infrastructure can use the same username and passwords by simply talking to an LDAP Directory Server.

Initial LDAP setup and population

If you would like to follow along in the LDAP setup portion of the article, you will need an instance of Fedora Core 8. In our case, we used an Amazon EC2 machine instance running Fedora Core 8 32-bit. You can also follow along by installing OpenLDAP onto a physical server or on a virtual machine using the technology of your choice. Note that we are using an example domain called unisonis.com for all of the examples, in spite of the fact that an RFC exists that recommends the use of example.com.

Step 1: Install openldap packages using yum:



```
[root@domU ]# yum install
openldap openldap-devel
openldap-servers
openldap-clients

[root@domU ]# yum list
installed | grep openldap
openldap.i386
2.3.39-4.fc8 installed
openldap-clients.i386
2.3.39-4.fc8 installed
openldap-devel.i386
2.3.39-4.fc8 installed
openldap-servers.i386
2.3.39-4.fc8 installed
```

Step 2: Set the administrator password (we will paste the SSHA hash into `slapd.conf`). Note that `slapd` stands for Standalone LDAP service, so this is the service that controls LDAP itself:

```
[root@domU ]# slappasswd
New password:
Re-enter new password:
```

Step 3: Edit the `slapd.conf` configuration file and add entries that pertain to the general LDAP installation, such as the root DN and the root/administrator password:

```
[root@domU ]# vi /etc/openldap/slapd.conf

#Add entries:

database bdb
suffix "dc=unisonis,dc=com"
rootdn "cn=Manager,dc=unisonis,dc=com"
rootpw {SSHA}pasted_from_slappasswd_output
directory /var/lib/ldap
```

Step 4: Start the LDAP service:

```
[root@domU ]# service ldap start
Starting slapd: [ OK ]
```

Step 5: Test existing setup by running an LDAP search for the 'namingContexts' attribute:

```
[root@domU ]# ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectclass=*)
# requesting: namingContexts
#
#
dn:
namingContexts: dc=unisonis,dc=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Step 6: Add more entries to the LDAP database with `ldapadd` using an LDIF file. Note that LDIF stands for LDAP Data Interchange Format, and it is a structure to format data for large updates to an LDAP database:

```
[root@domU ]# cat unisonis.ldif
dn: dc=unisonis,dc=com
objectclass: dcObject
objectclass: organization
o: Example Company
dc: unisonis

dn: cn=Manager,dc=unisonis,dc=com
objectclass: organizationalRole
cn: Manager

[root@domU ]# ldapadd -x -D "cn=Manager,dc=unisonis,dc=com" -W -f
unisonis.ldif
Enter LDAP Password:
adding new entry "dc=unisonis,dc=com"

adding new entry "cn=Manager,dc=unisonis,dc=com"
```

Step 7: The next step is to populate the LDAP directory with sample entries that we can act upon. We will use the information for the three stooges (entries inspired from an LDAP article available at <http://www.yolinux.com/TUTORIALS/LinuxTutorialLDAP.html>):

```
[root@domU ]# cat stooges.ldif; # to conserve space, we show the LDAP data for
only one of the three stooges
dn: ou=MemberGroupA,dc=unisonis,dc=com
```

```

ou: MemberGroupA
objectClass: top
objectClass: organizationalUnit
description: Members of MemberGroupA

dn: ou=MemberGroupB,dc=unisonis,dc=com
ou: MemberGroupB
objectClass: top
objectClass: organizationalUnit
description: Members of MemberGroupB

dn: cn=Larry Fine,ou=MemberGroupA,dc=unisonis,dc=com
ou: MemberGroupA
o: stooges
cn: Larry Fine
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
mail: LFine@unisonis.com
givenname: Larry
sn: Fine
uid: larry
homePostalAddress: 15 Cherry Ln. Plano TX 78888
postalAddress: 215 Fitzhugh Ave.
l: Dallas
st: TX
postalcode: 75226
telephoneNumber: (800)555-1212
homePhone: 800-555-1313
facsimileTelephoneNumber: 800-555-1414
userPassword: larrysecret
title: Account Executive
destinationindicator: /bios/images/lfine.jpg

[root@domU ]# ldapadd -x -D "cn=Manager,dc=unisonis,dc=com" -W -f stooges.ldif

```

If you are curious, you can now start experimenting with various searches against the LDAP database we created. Here is an example of searching for all LDAP entries related to the 'stooges' organization:

```
[root@domU conf.d]# ldapsearch -x -b 'dc=unisonis,dc=com' '(o=stooges)'
```

In the next section, we show how to configure Apache to authenticate to LDAP, before we jump into Python and LDAP.

Set up Apache LDAP authentication

One of the most common uses of LDAP is to provide authentication data for services such as Web servers. In this section we take our pre-populated LDAP database and use it to control access to an Apache virtual host.

To get started, we need to create an Apache configuration file for a virtual host that

is using LDAP authentication. We will require a valid e-mail and password for the user who is trying to log in. Here is what that looks like:

```
[root@domU ]# cat /etc/httpd/conf.d/unisonis.conf
<VirtualHost *:80>
  ServerName www.unisonis.com
  DocumentRoot "/ebs1/www/unisonis"
  <Directory "/ebs1/www/unisonis">
    AuthType Basic
    AuthName "unisonis.com: please login with email address"
    AuthBasicProvider ldap
    AuthLDAPURL ldap://localhost:389/dc=unisonis,dc=com?mail?sub?(o=stooges)
    require valid-user
    Order Allow,Deny
    Allow from all
    Options Indexes FollowSymLinks
    AllowOverride None
  </Directory>
</VirtualHost>
```

LDAP authentication is provided for Apache by the `mod_auth_ldap` module, which is installed by default in the Fedora Core 8 `httpd` package. All access to the Apache virtual host defined above will require a valid e-mail and password in the 'stooges' organization for the user trying to log in. Note that the `AuthLDAPURL` directive, which is what specifies the query we use in order to authenticate the user against the LDAP server. We search for the 'mail' attribute and we apply the filter `(o=stooges)`. For the complete syntax of the `AuthLDAPURL` directive, see http://httpd.apache.org/docs/2.0/mod/mod_auth_ldap.html#authldapurl.

Please see the [resources](#) section for more information on configuring LDAP with Apache.

Performing CRUD operations with Python-LDAP

Now we are ready to use Python to interact with LDAP. In order to do this, you must have the `python-ldap` module installed. If you refer to the [resource](#) section, you can find a link to more detailed information on installing the module. In a nutshell, you will need to perform an "easy install." First download the `easy_install` script here:

```
http://peak.telecommunity.com/dist/ez_setup.py
```

and then type:

```
sudo easy_install python-ldap
```

Note that there are some dependencies for the package that differ slightly depending on your operating system. Make sure you read the installation instructions for the package if you have trouble installing it.

With the install of python-ldap out of the way, we are ready to get into CRUD operations. Let's write a class to do that.

Python LDAP CRUD class

```
#!/bin/env python

import sys, ldap

LDAP_HOST = 'localhost'
LDAP_BASE_DN = 'dc=unisonis,dc=com'
MGR_CRED = 'cn=Manager,dc=unisonis,dc=com'
MGR_PASSWD = 'mypasswd'
STOOGE_FILTER = 'o=stooges'

class StoogeLDAPMgmt:

    def __init__(self, ldap_host=None, ldap_base_dn=None, mgr_cred=None, mgr_passwd=None):
        if not ldap_host:
            ldap_host = LDAP_HOST
        if not ldap_base_dn:
            ldap_base_dn = LDAP_BASE_DN
        if not mgr_cred:
            mgr_cred = MGR_CRED
        if not mgr_passwd:
            mgr_passwd = MGR_PASSWD
        self.ldapconn = ldap.open(ldap_host)
        self.ldapconn.simple_bind(mgr_cred, mgr_passwd)
        self.ldap_base_dn = ldap_base_dn

    def list_stooges(self, stooge_filter=None, attrib=None):
        if not stooge_filter:
            stooge_filter = STOOGE_FILTER
        s = self.ldapconn.search_s(self.ldap_base_dn, ldap.SCOPE_SUBTREE, stooge_filter, attrib)
        print "Here is the complete list of stooges:"
        stooge_list = []
        for stooge in s:
            attrib_dict = stooge[1]
            for a in attrib:
                out = "%s: %s" % (a, attrib_dict[a])
                print out
                stooge_list.append(out)
        return stooge_list

    def add_stooge(self, stooge_name, stooge_ou, stooge_info):
        stooge_dn = 'cn=%s,ou=%s,%s' % (stooge_name, stooge_ou, self.ldap_base_dn)
        stooge_attrib = [(k, v) for (k, v) in stooge_info.items()]
        print "Adding stooge %s with ou=%s" % (stooge_name, stooge_ou)
        self.ldapconn.add_s(stooge_dn, stooge_attrib)

    def modify_stooge(self, stooge_name, stooge_ou, stooge_attrib):
        stooge_dn = 'cn=%s,ou=%s,%s' % (stooge_name, stooge_ou, self.ldap_base_dn)
        print "Modifying stooge %s with ou=%s" % (stooge_name, stooge_ou)
        self.ldapconn.modify_s(stooge_dn, stooge_attrib)

    def delete_stooge(self, stooge_name, stooge_ou):
        stooge_dn = 'cn=%s,ou=%s,%s' % (stooge_name, stooge_ou, self.ldap_base_dn)
        print "Deleting stooge %s with ou=%s" % (stooge_name, stooge_ou)
        self.ldapconn.delete_s(stooge_dn)
```

The method names in our class are fairly self explanatory, so let's walk through some operations that would occur when we implement our class. If you have already followed the previous steps to populate an LDAP database, then you may want to download the code examples, as well.

First let's make an instance of the class:

```
l = StoogeLDAPMgmt()
```

At this point your patience has paid off, and we are ready to perform CRUD.

Next, add something programmatically with Python. Warning, you probably want to paste this example in from the source code you download, as this can be a lot to get correct by hand typing! Here is the "C" in CRUD:

LDAP Create

```
# add new stooage: Harry Potter
stooage_name = 'Harry Potter'
stooage_ou = 'MemberGroupB'
stooage_info = {'cn': ['Harry Potter'], 'objectClass': ['top', 'person',
'organizationalPerson', 'inetOrgPerson'],
                'uid': ['harry'], 'title': ['QA Engineer'], 'facsimileTelephoneNumber':
['800-555-3318'], 'userPassword': ['harrysecret'],
                'postalCode': ['75206'], 'mail': ['HPotter@unisonis.com'],
'postalAddress': ['2908 Greenville Ave.'],
                'homePostalAddress': ['14 Cherry Ln. Plano TX 78888'], 'pager':
['800-555-1319'], 'homePhone': ['800-555-7777'],
                'telephoneNumber': ['(800)555-1214'], 'givenName': ['Harry'], 'mobile':
['800-555-1318'], 'l': ['Dallas'],
                'o': ['stooages'], 'st': ['TX'], 'sn': ['Potter'], 'ou': ['MemberGroupB'],
'destinationIndicator': ['/bios/images/hpotter.jpg'], }
try:
    l.add_stooage(stooage_name, stooage_ou, stooage_info)
except ldap.LDAPError, error:
    print 'problem with ldap',error
```

Let's perform an "R," or Read, with this entry:

LDAP Read

```
# see if it was added
l.list_stooages(attrib=['cn', 'mail', 'homePhone'])
```

Now let's Update this, or perform the "U":

LDAP Update

```
# now modify home phone
stooage_modified_attrib = [(ldap.MOD_REPLACE, 'homePhone', '800-555-8888')]
try:
    l.modify_stooage(stooage_name, stooage_ou, stooage_modified_attrib)
except ldap.LDAPError, error:
    print 'problem with ldap',error
```

Finally, let's get the last letter from our acronym finished, "D", for Delete:

LDAP Delete

```
# now delete Harry Potter
try:
    l.delete_stooage(stooage_name, stooage_ou)
except ldap.LDAPError, error:
    print 'problem with ldap',error
```

Conclusion

This article briefly touched on the process of installing OpenLDAP on an Amazon EC2 Fedora instance. We populated an LDAP database with test data, and briefly explored interacting with it from the command line. We showed how to configure Apache to authenticate our sample LDAP database. Finally, we got into programmatically controlling LDAP from Python. One of the nice things about the Python example is how pleasant the code looks compared to what would be involved in scripting it from, say, bash. Python's reputation for clean, readable code certainly is evident in dealing with a fairly complex operation like programming LDAP.

We didn't really explore any pragmatic examples of LDAP and Python, but dealt with documenting how to use the API to perform common CRUD operations. Here is a practical idea for using the python-ldap library. You may want to populate different TRAC project management websites with all of the users in an LDAP group each time a new TRAC instance is created. This is easy to do using the techniques we covered: you query an LDAP group, then insert a permission in TRAC for each member of that group. There are many other practical ways to script LDAP, so hopefully this article got you excited about what you can do in your own projects next.

Special thanks to David Goodger for helping to review the article.

Downloads

Description	Name	Size	Download method
Sample LDAP script with tests	ldap_crud_code.zip	4KB	HTTP

[Information about download methods](#)

Resources

Learn

- [Amazon EC2](#) makes web-scale computing easier for developers.
- [Programming Amazon Web Services](#) shows you how companies can take advantage of Amazon Web Services.
- [Mastering OpenLDAP](#) show you how build, and Integrate Secure Directory Services with OpenLDAP server in a networked environment.
- Get the [LDAP client API for Python](#) official documentation
- [OpenLDAP documentation](#)
- [Apache LDAP documentation](#)
- The [Trac Integrated SCM and Project Management](#) is an enhanced wiki and issue tracking system for software development projects.
- [Modern Usage of "Baroque"](#)
- [The AIX and UNIX developerWorks zone](#) provides a wealth of information relating to all aspects of AIX systems administration.
- [developerWorks technical events and webcasts](#): Stay current with developerWorks technical events and webcasts.
- [Podcasts](#): Tune in and catch up with IBM technical experts.

Discuss

- - [AIX forum](#)
 - [AIX forum for developers](#)
 - [Cluster Systems Management](#)
 - [IBM Support Assistant Forum](#)
 - [Performance Tools Forum](#)
 - [Virtualization Forum](#)
 - [More AIX and UNIX forums](#)

About the authors

Noah Gift

Noah Gift is the co-author of "Python For Unix and Linux" by O'Reilly. He is an author, speaker, consultant, and community leader, writing for publications such as IBM developerWorks, Red Hat Magazine, O'Reilly, and MacTech. His consulting company's website is www.giftcs.com, and his personal website is www.noahgift.com. Noah is also the current organizer for www.pyatl.org, which is the Python User Group for Atlanta, GA. He has a Master's degree in CIS from Cal State Los Angeles, B.S. in Nutritional Science from Cal Poly San Luis Obispo, is an Apple and LPI certified SysAdmin, and has worked at companies such as, Caltech, Disney Feature Animation, Sony Imageworks, and Turner Studios. In his free time he enjoys spending time with his wife Leah, and their son Liam, playing the piano, and exercising religiously.

Grig Gheorghiu

Grig Gheorghiu is the Director of Technology for RIS Technology, a Web hosting company based in Los Angeles. Grig has 15 years of industry experience, during which time he has worked as a programmer, research lab manager, system/network/security architect, IT consultant, and lead test engineer. Grig is an active member of the Python and agile testing communities.

He maintains a blog, <http://agiletesting.blogspot.com> dedicated to agile testing, Python programming, and automated testing tools and techniques. Grig is the founder of the [Southern California Python Interest Group](#) (a.k.a. 'the SoCal Piggies'). He lives in Los Angeles with his wife and two children.