Customizing Apple's Open Directory for the Enterprise

John Anthony Grigutis
User Support Specialist III
Apple Certified System Administrator
grigutis@indiana.edu

Indiana University
Student Technology Centers

Overview

Open Directory is Apple's directory service solution that implements an OpenLDAP LDAP server, an MIT KDC, and Apple's own Password Server. While Apple has done a great job of integrating these technologies together, there is no easy way to customize Open Directory for large organizations with complex needs.

This paper documents the methods used to extend the functionality of Apple's Open Directory in Tiger Server for a large University environment (specifically, Indiana University). While concentrating on an educational environment, the principals outlined here can be used in any organization that wants greater flexibility from Open Directory.

Goals

Independent KDC

Open Directory includes MIT's Kerberos Key Distribution Center (KDC) which provides a very secure method of authentication. While this is a very good system for those who don't yet have a KDC, there is no provision within Open Directory for the use of a preexisting KDC. Many organizations (especially in higher ed.) already have a KDC and would like to leverage the effort already invested in their existing KDC instead of setting up a new one.

Since our university runs its own KDC, we (and our security office) didn't want to have another KDC to manage. We had to be able to run the LDAP portion of Open Directory without the KDC portion and use our University's existing KDC instead.

Multiple contexts (OUs)

With a standard Open Directory server, all clients see the same set of containers (users, groups, mounts, etc.). While this is fine for small workgroups, it presents problems for larger organizations with various different departments and groups. A good example are mount records. If a department wants to create mount records for its clients, all clients (even those outside the department) will have access to those mount records. What if the department wants to create a mount record exclusively for their own use or wants to supplant a mount record (/Network/Applications/ for example) created by another de-

partment? The traditional solution for this problem has been to have each department run their own Open Directory server.

Since our university is composed of multiple campuses, departments, and groups, we needed a single solution that could be used by everyone with a minimum of effort or hardware resources invested. Having each group set up their own Open Directory server seemed like an unnecessary waste of time and resources.

Distributed Administration
While there is a provision within Open Directory for restricting an administrator's access to certain user, group, and computer records, this involves modifying the records of all users that need such access. Since this cannot be done at the group level, it creates more work for the directory administrator and might not work properly if multiple OUs are implemented.

Ideally, an admin group record for each department should be able to use Workgroup Manager to manage their own part of the directory (OU) with little to no intervention from the directory administrator.

Centralized user base
In order to overcome some of the limitations already mentioned, some departments may want to run their own Open Directory servers. This creates a problem of user management. If a department needs to make its resources available to all members of the organization, it will have to create user accounts for potentially hundreds of thousands of users. This is a wasteful duplication of effort if every department has to do this on their own. It also introduces a greater security risk in that not every department will have the time or resources to ensure good security procedures or keep their servers up to date.

With over 290,000 users University-wide at IU, many of which cross departmental borders (e.g., students), having a centralized user base would save each department or group from the burden of having to create (or recreate) users. This combined with support for multiple contexts running on the same server would also eliminate the need for separate Open Directory servers for each department.

Open Directory Master Configuration

Creating the Open Directory Master
On a freshly installed Mac OS X Server, create an Open Directory Master in Server Admin. Be sure to enter the name of the Kerberos Realm you ultimately intend to use (i.e., your university's realm) so that all the services (afp, ftp, ssh, etc.) will already be configured for that realm (Figure 1).

Once the Open Directory Master has been created, stop and remove the KDC:

```
$ sudo sso_util remove -k -a diradmin -p password
$ sudo rm /etc/krb5.keytab
```

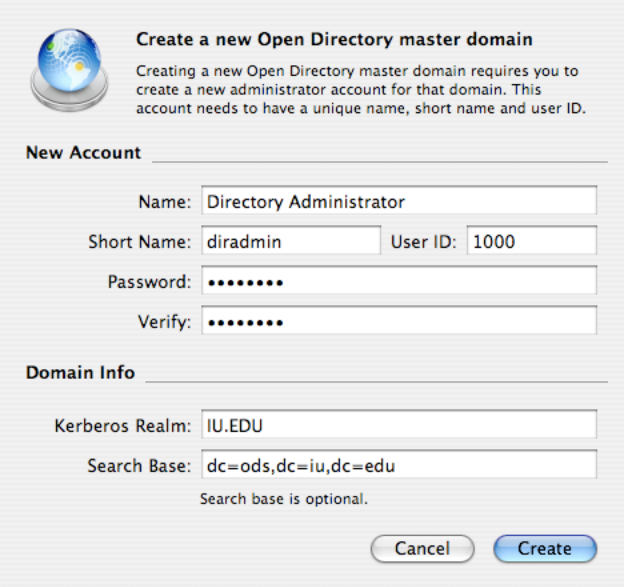


Figure 1. Setting up an Open Directory Master using Server Admin

Remnants of the KDC will still exist in the LDAP directory in:

cn=KerberosClient,cn=config,dc=ods,dc=iu,dc=edu
cn=KerberosKDC,cn=config,dc=ods,dc=iu,dc=edu

but we'll get to those later.

KDC keytab

Now that you've stopped the built-in KDC, you'll need to add a keytab from your organization's KDC. The default keytab included with Tiger's KDC includes all of the following services:

xgrid	XMPP	http	ftp
vpn	host	pop	afpserver
ipp	smtp	imap	ldap

You may not need all of these services, but I'd consider ldap and host to be the minimum for an Open Directory server. If you intend to host other services (AFP, Mail, etc.) on this server, you should request a keytab including those services as well.[1]

Depending upon how your organization delivers keytabs to you, you may be able to simply place the file in /etc/ with the name kerb5.keytab. Otherwise, use ktutil to combine and write the keytab to /etc/krb5.keytab. In either case, be sure the keytab is readable by the mail group if hosting a mail service:[2]

[1] Even though the service principals exist in the keytab, not all of these services are actually kerberized.

[2] Access to the keytab can also be managed via fie systems ACLs in Tiger.

```
-rw-r-----  1 root  mail   2954 Jun 27 21:43 krb5.keytab
```

See `man ktutil` for more information.

Cleaning up cn=config

Now is a good time to fix the remaining Kerberos issues in cn=config.

Since we no longer run our own KDC, we can remove the cn=KerberosKDC entry in cn=config. You can do this either via the ldapdelete command:

```
$ ldapdelete -x -D "uid=diradmin,cn=users,dc=ods,dc=iu,dc=edu" -W
"cn=KerberosKDC,cn=config,dc=ods,dc=iu,dc=edu"
```

or via the "All Records" tab in Workgroup Manager.

Select "Config" from the pull-down menu, then "KerberosKDC," then click the delete button in the toolbar (Figure 2).

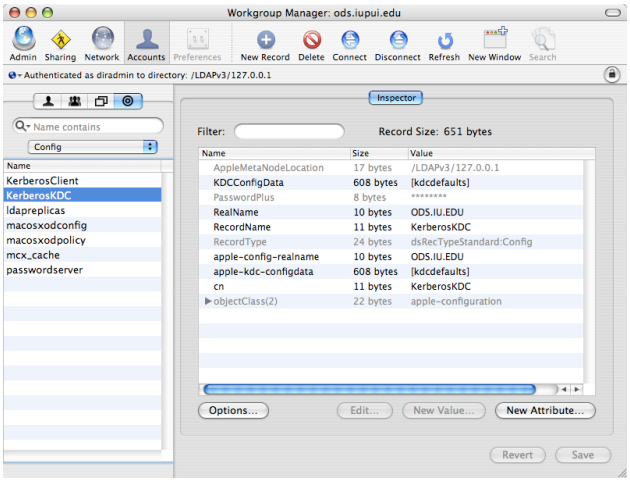


Figure 2. Deleting the KerberosKDC entry using Workgroup Manager

The other entry we need to modify is cn=KerberosClient. This entry is used by the kerberosautoconfig command to generate an edu.mit.Kerberos file for any clients that connect to this directory. The sso_util command that we issued earlier removed the server's hostname from this entry, but the realm and other data still exist. In any case, we'll want to replace this data with our organization's own edu.mit.Kerberos (aka krb5.conf) file.

The XMLPlist (aka apple-xmlplist) attribute in this entry is what we're interested in. In this case, using Workgroup Manager would be best since the ldapsearch command would only show us the raw data and not the actual text.

Using Workgroup Manager, click on the All Records tab (the bulls-eye) and locate the KerberosClient record in the Config section. Then view/edit the apple-xmlplist attribute, select all of the text, and paste it into a text file.

Once you have the data in a file, open it with Property List Editor (part of Apple's Xcode Tools). The format of the file should be obvious from the data that is already there. Now you can add your own realms and KDCs. Figure 3 is an example of what the file should look like.

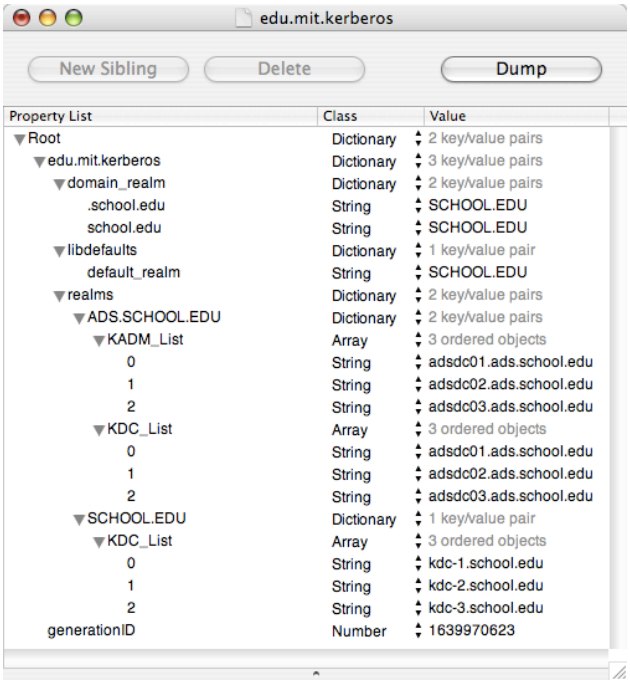


Figure 3. Editing the data from the KerberosClient entry using Property List Editor

The generationID is a timestamp that kerberosautoconfig uses in order to check if it needs to update the edu.mit.Kerberos file on the client. Since no one has connected to this directory yet (except the server itself), you could leave it alone, but I'd increment it by one just in case. See `man kerberosautoconfig` for more information.

Save the file in case you need to make changes later, then click the Dump button. This will convert the file to the original XML format (Figure 4).

Now it's just a matter of cutting and pasting this data back into the apple-xmlplist attribute in Workgroup Manager.

Finally, update the edu.mit.kerberos file on the server. To do this, run:

```
$ sudo kerberosautoconfig -u
```

Kerberos authAuthority in diradmin

Since we're using a different KDC now, the Kerberos authAuthority attribute for the Directory Administrator (diradmin) is no longer valid. To fix this, just delete the attribute or replace it with a user principal from your current KDC. Since diradmin still has the ApplePasswordServer authAuthority attribute, the Kerberos attribute is optional.

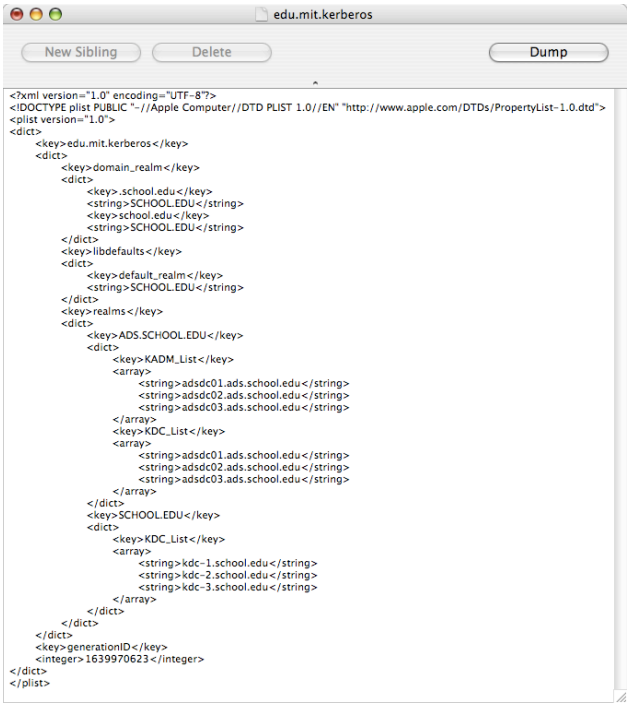


Figure 4. XML dump of the revised data for the KerberosClient entry

Congratulations, your Open Directory Master is now configured to use a third party KDC!

phpLDAPadmin

Before we get into the directory structure, there is one tool that you should know about, phpLDAPadmin:

http://phpldapadmin.sourceforge.net/

phpLDAPadmin is a web based LDAP browser and administration tool that provides a nice GUI for all the major LDAP functions and makes looking at "the big picture" a lot easier. Some of the following screen shots will be from this tool.

Directory Structure

Out of the box, Mac OS X Server creates all the necessary LDAP container objects (computers, users, groups, etc.) in the base context that you specified during the creation of the OD Master (in our case, dc=ods,dc=iu,dc=edu). This is fine if you have a small organization where every client can use the same set of users, groups, mounts, etc., but if you have a more complex organizational structure with multiple departments or campuses, sharing the same set of containers can be problematic.

Previously, the solution to this problem has been to set up separate Mac OS X Servers hosting separate Open Directory Masters for each group that wanted to control their own space in the directory. This seems like a waste of resources if a single Open Directory could be made to accommodate the needs of multiple groups.

Luckily, due to the way that Open Directory abstracts the directory structure from the client, it doesn't really matter where in the directory the containers are. As long as the search base and mappings are correct in Directory Access, a client can use containers anywhere in the directory tree. This means that each client could look for directory information in different branches of the tree and receive its own set of users, groups, managed client settings (MCX), etc. As we'll see later, LDAP aliases can also be used to create more complex structures that allow a combination of customizable directory data with shared "universal" data.

So in order to support multiple groups on one server we'll need to create new branches in the directory tree and recreate the default container structure in these new branches. The client machines can then be pointed directly at these new branches and group administrators can be granted rights to do whatever they like without affecting other areas of the directory.

The simplest structure is the one provided by Mac OS X Server out of the box. All the containers are at the root of the tree (Figure 5).

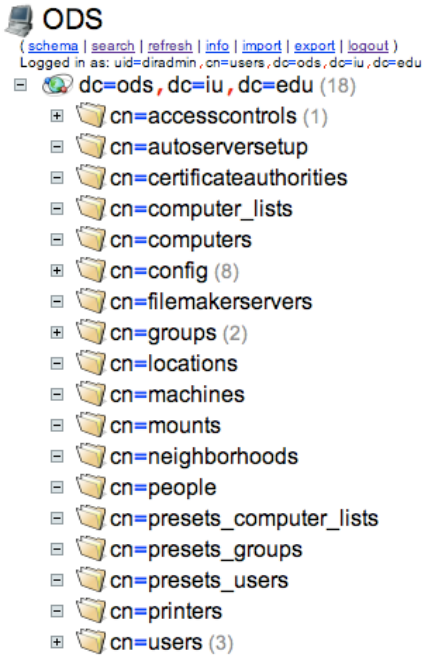


Figure 5. The default directory structure provided by Mac OS X Server

The easiest way to create new branches is via organizationalUnit (OU) objects. Since our university's Active Directory system already used a nested OU structure to organize campuses, departments, and groups, I decided to use this same structure for Open Directory. OUs at the first level represent campuses, OUs at the second level represent departments, and OUs at the third level represent groups within the department if necessary (Figure 6).

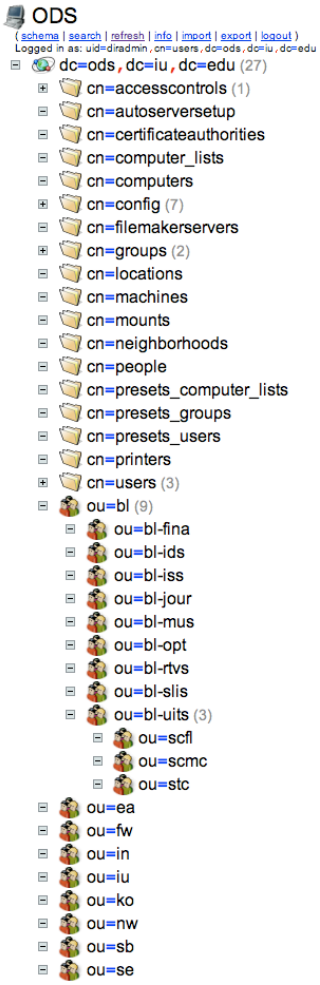


Figure 6. Campus/Department/Group OU structure

If you only need to create one or two OUs, you can use phpLDAPadmin, but if you're going to create several all at once, the command line tool ldapadd is much more efficient. Here's the file (oucreate.ldif) used to create the ou=bl subtree above:

```
dn: ou=bl,dc=ods,dc=iu,dc=edu
ou: bl
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-fina,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-fina
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-ids,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-ids
objectClass: top
objectClass: organizationalUnit
```

```
dn: ou=bl-iss,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-iss
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-jour,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-jour
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-mus,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-mus
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-opt,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-opt
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-rtvs,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-rtvs
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-slis,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-slis
objectClass: top
objectClass: organizationalUnit

dn: ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu
ou: bl-uits
objectClass: top
objectClass: organizationalUnit

dn: ou=scfl,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu
ou: scfl
objectClass: top
objectClass: organizationalUnit

dn: ou=scmc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu
ou: scmc
objectClass: top
objectClass: organizationalUnit

dn: ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu
ou: stc
objectClass: top
objectClass: organizationalUnit
```

The corresponding ldapadd command is:

```
$ ldapadd -x -D "uid=diradmin,cn=users,dc=ods,dc=iu,dc=edu" -W -f oucreate.ldif
```

Populating the branches

Once your OUs are created, you'll need to populate them with the containers that Mac OS X expects to find. It's possible to simply create separate instances of the container objects found at the root of the tree in each OU, but a more efficient approach is to use a combination of regular containers and LDAP aliases.

Since we don't want to burden each department with the creation and maintenance of user records, an LDAP alias can be used in place of the standard users container object to point to a "universal" users container (cn=users,dc=ods,dc=iu,dc=edu) that contains all the users in the organization. This container can then be maintained by a central authority and each department will see any updates as soon as they happen. Other objects that don't require a separate instance in each OU (cn=accesscontrols, and most of the objects in cn=config) can also be aliased.

The easiest way to do this is via ldapadd. Once you have a properly formatted ldif, it's easy to modify and reuse it for any OU you want to populate. Figure 7 shows the structure I'm using.

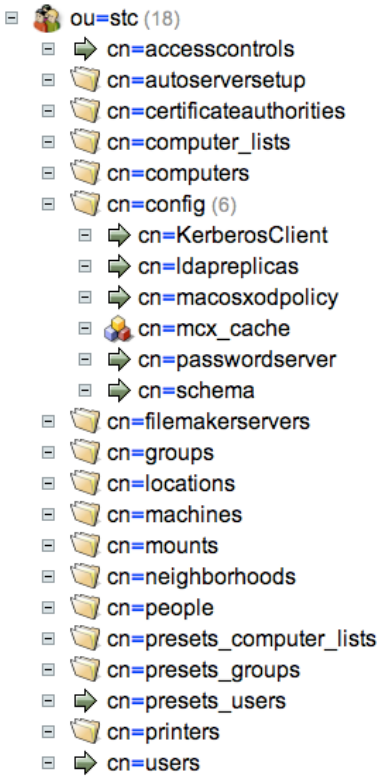


Figure 7. Custom container structure for each OU

The corresponding file (ousetup.ldif) is:

```
dn: cn=accesscontrols,ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu
cn: accesscontrols
objectClass: alias
```

objectClass: extensibleObject
aliasedObjectName: cn=accesscontrols,dc=ods,dc=iu,dc=edu

dn: cn=autoserversetup,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: autoserversetup
objectClass: container

dn: cn=certificateauthorities,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: certificateauthorities
objectClass: container

dn: cn=computer_lists,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: computer_lists
objectClass: container

dn: cn=computers,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: computers
objectClass: container

dn: cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: config
objectClass: container

dn: cn=filemakerservers,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: filemakerservers
objectClass: container

dn: cn=groups,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: groups
objectClass: container

dn: cn=locations,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: locations
objectClass: container

dn: cn=machines,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: machines
objectClass: container

dn: cn=mounts,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: mounts
objectClass: container

dn: cn=neighborhoods,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: neighborhoods
objectClass: container

dn: cn=people,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: people
objectClass: container

dn: cn=presets_computer_lists,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: presets_computer_lists
objectClass: container

dn: cn=presets_groups,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: presets_groups
objectClass: container

dn: cn=presets_users,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: presets_users
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=presets_users,dc=ods,dc=iu,dc=edu

dn: cn=printers,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: printers
objectClass: container

dn: cn=users,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: users
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=users,dc=ods,dc=iu,dc=edu

dn: cn=KerberosClient,cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: KerberosClient
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=KerberosClient,cn=config,dc=ods,dc=iu,dc=edu

dn: cn=ldapreplicas,cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: ldapreplicas
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=ldapreplicas,cn=config,dc=ods,dc=iu,dc=edu

dn: cn=macosxodpolicy,cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: macosxodpolicy
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=macosxodpolicy,cn=config,dc=ods,dc=iu,dc=edu

dn: cn=mcx_cache,cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: mcx_cache
objectClass: apple-configuration
objectClass: top

dn: cn=passwordserver,cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: passwordserver
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=passwordserver,cn=config,dc=ods,dc=iu,dc=edu

dn: cn=schema,cn=config,*ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu*
cn: schema
objectClass: alias
objectClass: extensibleObject
aliasedObjectName: cn=schema,cn=config,dc=ods,dc=iu,dc=edu

To to set up another OU, just replace the *ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu* portion of each entry with the specific OU you want the objects created in. The corresponding ldapadd command is:

```
$ ldapadd -x -D "uid=diradmin,cn=users,dc=ods,dc=iu,dc=edu" -W -f ousetup.ldif
```

Adding Users

There are two ways you can add users to Open Directory: dsimport or ldapadd. Since identity management and account generation is handled by a separate group at our university, using the standard LDAP methods (ldapadd, ldapdelete, ldapmodrdn) was the best choice since they were already familiar with these tools.

Here is an example ldif of a user with the minimal set of attributes that we needed:

```
dn: uid=grigutis,cn=users,dc=ods,dc=iu,dc=edu
uid: grigutis
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: apple-user
objectClass: extensibleObject
objectClass: organizationalPerson
objectClass: top
objectClass: person
homeDirectory: /Users/grigutis
uidNumber: 1028
authAuthority: ;Kerberosv5;;grigutis@IU.EDU;IU.EDU
gidNumber: 20
loginShell: /bin/bash
sn: Grigutis
apple-generateduid: 92648CE3-4F44-4B5F-8736-5C7EF0F11922
cn: John Anthony Grigutis
```

The other method is to use dsimport. The advantage of dsimport is that you can omit some of the attributes and dsimport will fill them in with the appropriate values (such as apple-generateduid, gidNumber, uidNumber). `man dsimport` for more information. Here is how the user in the ldapadd example above would look in a dsimport input file (user.txt):

```
0x0A 0x5C 0x3A 0x2C dsRecTypeStandard:Users 6 RecordName
AuthenticationAuthority RealName LastName NFSHomeDirectory UserShell
grigutis:;Kerberosv5;;grigutis@IU.EDU;IU.EDU:John Anthony Grigutis:
Grigutis:/Users/grigutis:/bin/bash
```

The corresponding command is:

```
$ dsimport -g user.txt /LDAPv3/127.0.0.1 -O -u diradmin
```

Client Configuration

There are two types of client (with a lowercase "c") configurations to consider: Mac OS X Clients and Mac OS X Servers.

Mac OS X Client

/etc/openldap/ldap.conf

Before we set up Directory Access, we'll need to modify /etc/openldap/ldap.conf so that the aliases we created are dereferenced properly. Change the following line:

```
#DEREF          never
```

to:

```
DEREF           always
```

Directory Access

In Directory Access, manually create a new LDAPv3 configuration (click "New…" then "Manual"). The configuration name can be whatever you'd like (I prefer to give it the same name as the OU it will connect to). The server name will be your Open Directory server of course. For the LDAP mappings, select "Open Directory Server" (Figure 8).

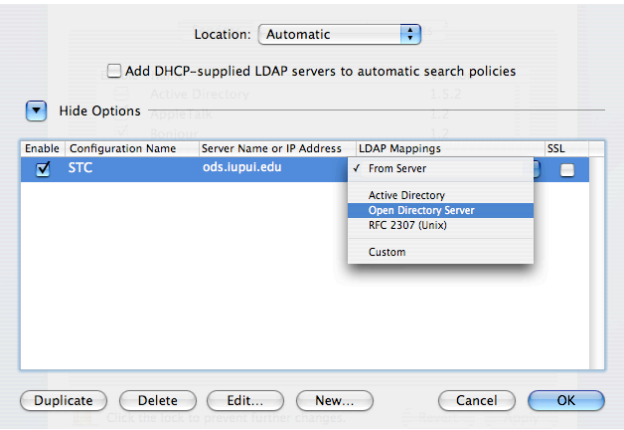


Figure 8. Configuring the client's LDAPv3 settings using Directory Access

In the Search Base Suffix field that appears, enter the DN of the OU you want the client to connect to (Figure 9).

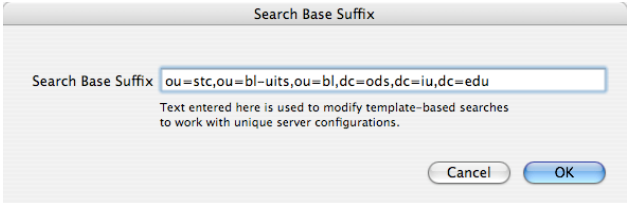


Figure 9. Specifying the OU to use

Be sure to add this new configuration to the Authentication and Contacts search paths (Figure 10).

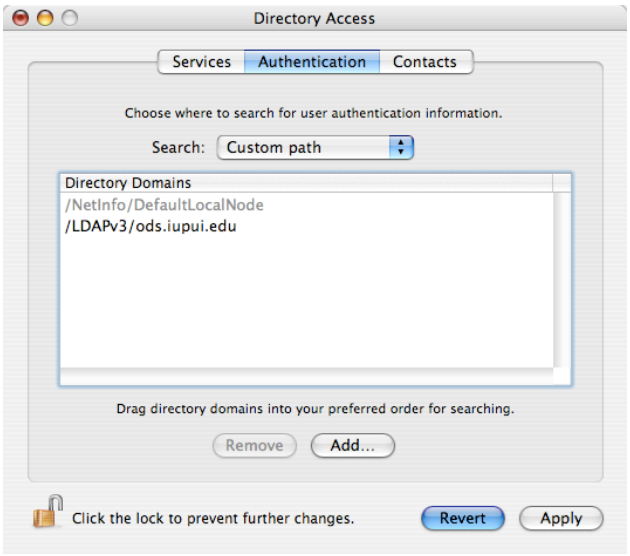


Figure 10. Adding the new configuration to the authentication path

Finally, if you want users to have a krbtgt upon login, you'll have to make a slight modification to /etc/authorization. In the system.login.console section, replace:

authinternal

with:

builtin:krb5authnoverify,privileged

Note that this only works if the realm listed in the user's Kerberbos authAuthority is the same as the default realm in /Library/Preferences/edu.mit.kerberos.

That's all! Your client is now ready.

Mac OS X Server
Setting up a server is the same as a client with three additional steps: change the server's role in Server Admin, install a keytab, and configure the services to use the keytab.

Open Directory Role
Follow the instructions for a Mac OS X Client above, then in the Open Directory section of Server Admin, set the role to "Connected to a Directory System" (Figure 11).

Install a keytab
Install a keytab for the services you intend to host on this server. Refer to the **KDC keytab** section on page 3 for more information.

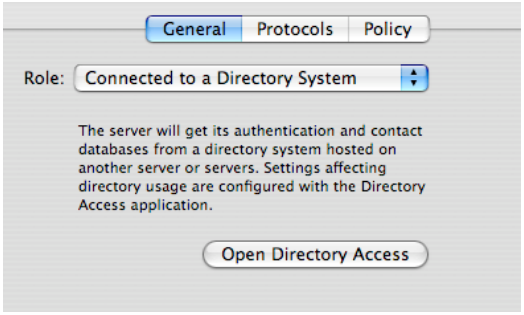


Figure 11. Setting a server's Open Directory role using Server Admin

Configure services with krbservicesetup

In order for the server to use Kerberos authentication, each service will need to know what service principal to use. Normally, this would involve editing various configuration files by hand, but Apple has included a little utility called krbservicesetup to do all this for you.

krbservicesetup can configure afp, ftp, imap, pop, smtp, and ssh. Since we already have service principals in the keytab from our organization's KDC, we'll need to create a plist that tells krbservicesetup to skip the creation of new principals and to configure the services using our existing principals. `man krbservicesetup` for more information. Here is an example plist that configures all the services (setup_file.plist):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Services</key>
        <array>
                <dict>
                        <key>servicePrincipal</key>
                        <string>afpserver/server.school.edu@SCHOOL.EDU</string>
                        <key>serviceType</key>
                        <string>afp</string>
                </dict>
                <dict>
                        <key>servicePrincipal</key>
                        <string>ftp/server.school.edu@SCHOOL.EDU</string>
                        <key>serviceType</key>
                        <string>ftp</string>
                </dict>
                <dict>
                        <key>servicePrincipal</key>
                        <string>imap/server.school.edu@SCHOOL.EDU</string>
                        <key>serviceType</key>
                        <string>imap</string>
                </dict>
                <dict>
                        <key>servicePrincipal</key>
                        <string>pop/server.school.edu@SCHOOL.EDU</string>
```

```
                <key>serviceType</key>
                <string>pop</string>
        </dict>
        <dict>
                <key>servicePrincipal</key>
                <string>smtp/server.school.edu@SCHOOL.EDU</string>
                <key>serviceType</key>
                <string>smtp</string>
        </dict>
        <dict>
                <key>servicePrincipal</key>
                <string>host/server.school.edu@SCHOOL.EDU</string>
                <key>serviceType</key>
                <string>ssh</string>
        </dict>
    </array>
    <key>configOnly</key>
    <true/>
</dict>
</plist>
```

The corresponding command is:

```
$ sudo krbservicesetup -f setup_file.plist
```

That's all! Your server is now ready.

Administrator Configuration

Three things need to be done in order to allow administrative access to an individual OU:

1. An ApplePasswordServer authAuthority attribute needs to be created for the users that will act as the OU's administrators.

2. An admin group needs to be created in the OU (with the administrative users as members).

3. The LDAP ACL needs to be modified to allow this admin group to write to the respective OU.

Password Server authAuthority

You may have noticed that for user records, we're using only the Kerberos authAuthority (no Password Server authAuthority). For most users in our environment, this is enough. Having a Password Server authAuthority for each user would be nice, but integrating the Password Server with our existing Identity Management Systems requires some additional work. This is something that we might do in the future, but for the moment, Kerberos is good enough for most of our needs.

One critical application where a Password Server authAuthority is necessary is Workgroup Manager. Admin users will not be able to authenticate without one. So for users that will act as administrators of an OU, I create a password for them in the Password Server and add the appropriate value to their authAuthority attribute.[3]

To create a entry in the Password Server for a user, issue this command from the server:

```
$ sudo mkpassdb -u user -b -p
```

If successful, the result of this command will be a hex GUID that represents the user's slot in the database. The format of the Password Server authAuthority attribute is:

;ApplePasswordServer;*GUID*,*Public Key*:*IP of Password Server*

To find out what the Password Server's Public Key is, use:

```
$ sudo mkpassdb -key
```

Or to get the public key along with a list of all the hex GUIDs, use:

```
$ sudo mkpassdb -dump
```

Another way to find out the proper format for the Password Server authAuthority attribute is to copy the attribute for an existing user (e.g., diradmin) and just replace the GUID portion.

---

[3] Another option is to create separate admin users for use with Workgroup Manager. Since authAuthority is the only attribute that we need, other attributes can have values that prevent the account from being used as a normal user account. Here is an ldif for such a user [note that I've created a separate container (cn=admins) for these users]:

```
dn: uid=adm_grigutis,cn=admins,cn=users,dc=ods,dc=iu,dc=edu
apple-generateduid: 5A0BB5F1-8918-4EDA-BB29-CE8A6396301E
authAuthority: ;ApplePasswordServer;0x434d928329141da…
cn: adm_grigutis
gidNumber: -2
homeDirectory: /var/empty
loginShell: /usr/bin/false
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: apple-user
objectClass: extensibleObject
objectClass: organizationalPerson
objectClass: top
objectClass: person
sn: adm_grigutis
uid: adm_grigutis
uidNumber: -2
```

Once you have a properly formated Password Server authAuthority, add it to the user's record (via ldapmodify, Workgroup Manager, or phpLDAPadmin).

The user can change the password later in Workgroup Manager or via the pwpolicy command from any client connected to the directory:

```
$ pwpolicy -a authenticator -u user -setpassword newpassword
```

Note that if you intend to let the Password Server handle authentication for other non-Kerberized services besides Workgroup Manager (e.g., phpLDAPadmin), you should make the Password Server the primary value for the authAuthority attribute and let Kerberos be the secondary value. Otherwise, authentication will fail.

admin group

As far as Workgroup Manager is concerned, any users in the admin group are considered to be directory admins and have full rights to modify the directory. Whether this is true according to the OpenLDAP ACL is a separate issue that is covered in the following section.

To create the admin group for the OU, use the dsimport command. This time you'll have to use it from a client machine that has been configured for the OU in question. I'm using the machine I set up in the **Client Configuration** section above. Here is the import file (admin.txt):

```
0x0A 0x5C 0x3A 0x2C dsRecTypeStandard:Groups 3 RecordName
PrimaryGroupID GroupMembership
admin:80:grigutis
```

The corresponding command is:

```
$ dsimport -g admin.txt /LDAPv3/ods.iupui.edu -O -u diradmin
```

OpenLDAP ACL (cn=accesscontrols)

The last step is to modify the LDAP access controls to give the admin group in each OU write access to the OU itself. In a plain vanilla OpenLDAP installation, these access controls are located in the slapd.conf file, but in Open Directory, these have been moved into the directory itself. This is a good thing in terms of replication since the controls don't have to be synced separately from the directory data.

The object we need to look at is:

```
cn=default,cn=accesscontrols,dc=ods,dc=iu,dc=edu
```

The access directives are exactly like those that would be found in a slapd.conf file. The main difference being that each directive is numbered so that the server knows the evaluation order (more specific directives should be evaluated before more general ones). The following is the directive I used for the stc OU (reformatted for easier reading):

```
1500:
access to dn.subtree="ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu"
        by group/posixGroup/memberUid="cn=admin,cn=groups,ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu" write
        by group/posixGroup/memberUid="cn=admin,cn=groups,dc=ods,dc=iu,dc=edu" write
        by * read
```

You can use Workgroup Manager, ldapmodify, or phpLDAPadmin to add directives. The following is the file (acl.ldif) for the ldapmodify method:

```
dn: cn=default,cn=accesscontrols,dc=ods,dc=iu,dc=edu
changetype: modify
add: apple-acl-entry
apple-acl-entry: 1500:access to dn.subtree="ou=stc,ou=bl-uits,ou=bl,dc=ods,dc=iu,dc=edu"
    by group/posixGroup/memberUid="cn=admin,cn=groups,ou=stc,ou=bl-
    uits,ou=bl,dc=ods,dc=iu,dc=edu" write
    by group/posixGroup/memberUid="cn=admin,cn=groups,dc=ods,dc=iu,dc=edu" write
    by * read
```

Note that the apple-acl-entry should be on a single line (no returns). Here is the corresponding command:

```
$ ldapmodify -x -D "uid=diradmin,cn=users,dc=ods,dc=iu,dc=edu" -W -f acl.ldif
```

For more information on OpenLDAP's access controls, see `man slapd.access` and <http://www.openldap.org/doc/admin22/slapdconfig.html#Access Control>.

OU Administration

Workgroup Manager

Now that everything is set up, how can local administrators use Workgroup Manager to create groups, computer lists, and managed preferences in their own OUs?

Normally, one would connect directly to the Open Directory server to manage the users, groups, computers, etc. (Figure 12).

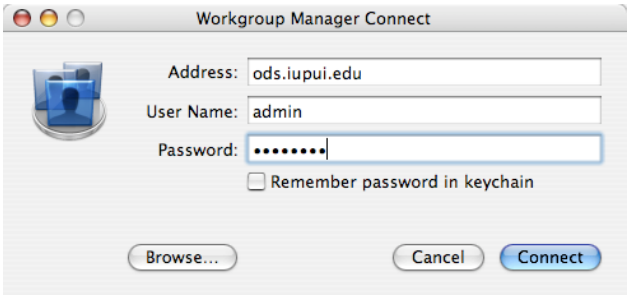


Figure 12. Standard "Connect…" dialog in Workgroup Manager

This can still be done by the directory administrator (diradmin), but it will only modify the containers at the root level of the tree (the default configuration). There is no facility within Workgroup Manager to transverse the directory tree, but we can use a certain feature to get around this limitation.

From a Client

Administration of an OU can be performed from any client machine that has been configured to use the OU (as outlined in the **Client Configuration** section above). Instead of using the Connect dialog that appears when Workgroup Manager first opens, select the View Directories command (Figure 13).

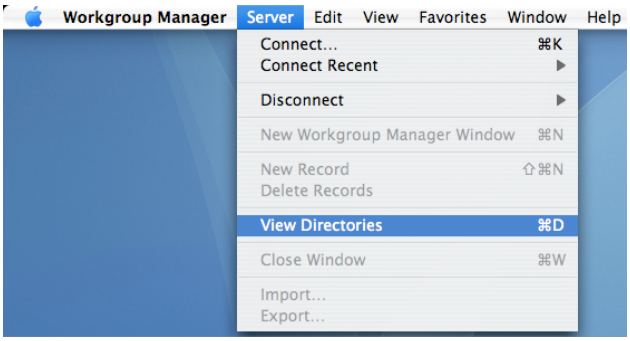


Figure 13. Select "Server > View Directories" to administer the OU

If the local NetInfo node is displayed, click the globe icon, select "Other…" and select the correct node from the LDAPv3 section (Figure 14).

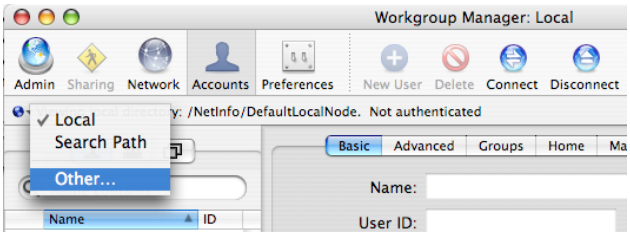


Figure 14. Select "Other…" if the local NetInfo domain is shown

You can now authenticate and use Workgroup Manager as you normally would.

From a Server

If you have a server that has been configured (as outlined in the **Client Configuration** section above), you can connect to the server using the standard Connect dialog in Workgroup Manager. This will allow you to set up share points on the server and add mount records to the server's OU.

Notes & Observations

Multiple Directory Nodes

One drawback with directory nodes in Open Directory is that you really can't have two nodes that point to the same server. If I need to administer two or more OUs, I'll need to either reconfigure Directory Access each time I want to modify a different OU or use two or more separate client systems (each configured to use one OU). The problem is that the node names are based on the address of the server. Even if a different search base is used in each case, each node has the same name. This makes it difficult to differenti-

ate between nodes in Workgroup Manager. It would have been nicer if node names were based on the configuration name in Directory Access.

Trusted Binding
Normally, when performing a trusted bind, principals are created for the computer in both the Password Server and the KDC. Since our university's security office doesn't grant kadmin access to mere mortals, we can't create principals in the KDC.

That said, the other possibility is to use the Password Server alone for trusted binding, but this has its problems as well. Apple's Password Server doesn't support tiered administrative levels. This means that you're either the admin for the entire database, or you're not—there is no in-between. So if you make a local OU administrator a Password Server admin, they could create new password entries, but they also could delete or change other password entries created by other local administrative users. This is a bad thing.

For these reasons, I'm not enabling trusted binding in this configuration.

Department (or Group) supplied Network Home Directories
In the structure I've created, the user records are shared among all the various departments and groups within the organization. Having a large user base (almost 300,000 at IU) makes hosting network home directories a little expensive, so each user's homeDirectory attribute is set to /Users/*username*. This forces the creation of home folders on the local hard drive.

But what if a department had the resources and wanted to host network home directories for its users? Normally, the department would have to modify each user record directly (if the University would allow the modification of a centralized user record at all) or create separate user records for networked home directories. This is time consuming, a nightmare to manage, and would cause trouble if the users in question logged into a system in another department where their home directories from the first department were inaccessible.

Luckily, groups that want to provide this service can do so without modifying the user records. Instead, local admins just need to modify the LDAP mappings in Directory Access. Mapping NFSHomeDirectory and HomeDirectory attributes to static values with variables lets users' records stay fixed while any user with access rights to login to a certain computer can have a network home directory.

Assuming that a mount record exists for the share point in question, A basic mapping for NFSHomeDirectory would look like:

```
#/Network/Servers/nethome.iu.edu/Users/$uid$
```

A basic mapping for HomeDirectory would look like:

#<home_dir><url>afp://nethome.iu.edu/Users</url><path>uid</path></home_dir>

Server-side LDAP mappings

Creating LDAP aliases for all the objects that exist at the root of the tree but not in each OU (accesscontrols, users, presets_users, & some of the items in config) allows you to skip setting up any server-side LDAP mappings. Clients can simply use the *Open Directory Server* LDAP mapping in Directory Access. Although putting the server side mappings in place is not a bad idea. This ensures that everything will still work even if a user has chosen the *From Server* LDAP mapping.

It also makes it easier to deploy customized LDAP mappings. Clients can grab their mappings from the directory instead of having to customize the mappings on each client.

Server-side mapping can also be used by departments or groups to extend the functionality of the LDAP server and customize it without having to modify individual records in LDAP (see **Department (or Group) supplied Network Home Directories** above).

That said, the actual implementation of server-side LDAP mappings is somewhat irregular. In testing, I've found that even though the default server-side mappings included with a standard Mac OS X Server installation (ou=macosxodconfig,cn=config,dc=ods,dc=iu,dc=edu) work as expected, mappings that are saved in other locations don't appear correctly in Directory Access. Although the mappings appear to work correctly, Directory Access displays no mappings at all as shown in Figure 15.

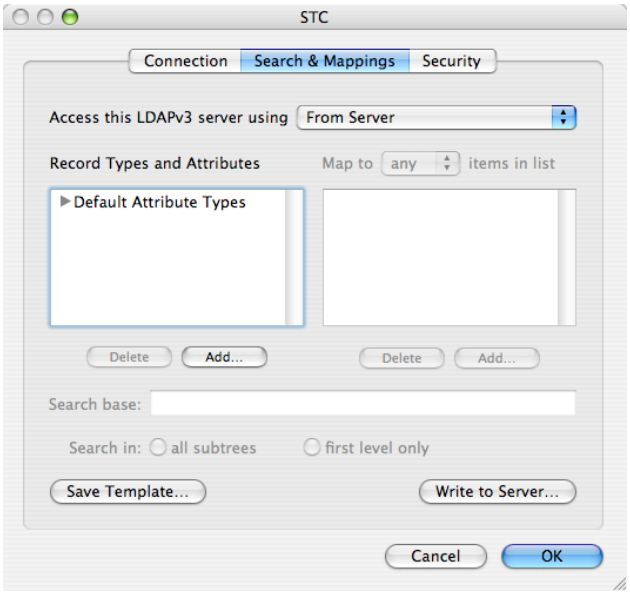


Figure 15. No mappings are displayed if an OU is specified

This is probably just a bug in Directory Access, but I haven't had time to investigate further.

Base context
Since we can now create as many OUs as we need and they can be individually customized to each group's liking, what should we do with the base context (dc=ods,dc=iu,dc=edu)? I can think of two roles that it can serve:

1. Act as generic context for groups that want to have a lab full of Macs that anyone can log in to, but don't need or want any directory based administrative control.

2. A university wide context that could be used by the population at large (mostly students in their dorm rooms) that don't otherwise have a specific OU to use.

It is even easier to configure a client in these cases since all that is needed is the server's DNS name (Figure 16).



Figure 16. Automatic LDAP server setup

In environments where the network is well managed, this could also be deployed network wide via the LDAP URL feature of DHCP (option 95) in which case the need for user or administrator intervention would be reduced even further.

Global policies can then be deployed for these machines using the Guest Computers list.

Acknowledgements

Many thanks to Michael Bartosh of 4AM Media for helping to figure out the naughty bits, testing, and providing feedback on this document. I would have never have been able to complete this project if it wasn't for his help.

Thanks to Nell Owens and Dick Hamstra from Apple Computer for making it possible to get Michael on-site and for providing the development hardware.

Thanks to Joel Rennich of afp548.com for his feedback and for offering to host this document.

Thanks to the developers of phpLDAPadmin for making the best LDAP tool around.

And a big thanks to my wife Sachiko for putting up with me while I was attached to a laptop for most of the summer.