Autofs: Automatically Mounting Network File Shares in Mac OS X

Technical White Paper
June 2009

Contents

Page 3 **Overview**
Background
Applying Changes
Automount Timeout and Other Autofs Options
auto_master

Page 5 **Direct, Indirect, and Executable Maps**
Direct Maps
Indirect Maps
Executable Maps
Wildcards and Submounts
Mount Options

Page 9 **Special Maps**
fstab
Static Automounts
Directory Utility
Server Admin
Hosts Automounts
File System Options

Page 15 **Maps in Directory Services**
Autofs Maps and Automounts via LDAP
Advanced LDAP Configuration
Configuring Mac OS X to Access Maps in LDAP
Setting Up Autofs to Use Directory Services

Page 19 **NIS Schema Example**

Page 23 **Autofs in Mac OS X**
Dots Versus Underscores
automountInformation Format
Wildcards
Finder Integration
Kerberized NFS
Other Ways to Mount
User Home Directories
Deep Mounts

Page 26 **Conclusion**

Page 27 **Glossary**

Page 28 **Appendix: Test Source Code**

Overview

Mac OS X v10.5 Leopard offers a new way of automatically mounting network file shares: autofs. Mounting network file shares is a useful technique when a directory or file share on one server needs to be accessible from other servers or client machines on the network. Autofs uses standard autofs maps already familiar to Solaris and Linux administrators, and the new automounting capabilities in Leopard offer a robust and expanded feature set for automatically mounting file shares in Mac OS X systems. The autofs file system in Mac OS X is multithreaded, keeps track of which paths are located on remote Apple Filing Protocol (AFP), Server Message Block (SMB), or Network File System (NFS) file servers—even across symlinks—and automatically mounts the appropriate server. The Finder and other applications do not have to wait for one mount to be completed before requesting another, resulting in increased responsiveness. Administrators can specify automount paths for an entire organization using the same standard automounter maps supported by Linux or Solaris to provide greater compatibility with other UNIX-based systems.

This how-to guide for system administrators covers the features of autofs, describes how to set it up, and offers tips on how to integrate it into directory services.

Background

Autofs solves the problem of mounting and unmounting volumes to client machines. It addresses a multitude of scenarios:

- A number of exports or file shares need to be mapped to different places in the local file system (configured in autofs using direct maps).
- A number of exports or file shares from different servers or the same server need to be mounted inside the same directory (configured in autofs using indirect maps).
- A list of mounts exists in a format that autofs cannot understand (configured in autofs by calling a script for the mount whenever a local directory is accessed—the script passes back the path to be mounted using an executable map).
- Autofs needs to mount maps in fstab format. fstab format entries can come from /etc/fstab, local directory services, or Open Directory. Autofs can either mount them in the same place (the -fstab maps) or in the location specified by the fstab entry using -static maps.
- Autofs needs to automatically discover NFS servers with exports and create mounts as administrators navigate to folders with the same name as the export using the -hosts map.

All maps are stored in directory services (either NIS or LDAP) and discovered through directory service calls. This allows administrators to make changes in one place and automatically propagate the maps to client systems.

Applying Changes

Any changes made to the autofs configuration and maps can be applied immediately by running automount. If the -v flag is specified, it will run with verbose output. If the -c flag is specified, the cache will be cleared. Following is the command for running automount:

```
sudo automount -vc
```

Autofs will attempt to unmount exports or shares that have been removed and create triggers and mounts for any automounts added to the configuration. Note that the -v option is only useful for verifying whether automount understood any changes.

Automount Timeout and Other Autofs Options

One of the advantages of using autofs is that it will mount exports or shares when the mount point is accessed, or "triggered," and unmount them when they have not been used for a specified time. In Mac OS X, this is set by default for 60 minutes. Administrators can set the duration before an unused export or share is unmounted by editing the /etc/autofs.conf file and changing the AUTOMOUNT_TIMEOUT option. The /etc/autofs.conf file also has settings related to logging, tracing, and general mount options.

auto_master

The /etc/auto_master file defines which maps are available. Maps are a collection of automounts that are related by mount point organized in the same file or location in directory services. An automount defines a place in the local file system within the map that points to an export or a file server on the network.

Since automounts can be defined in a number of different ways, autofs can manage automounts using its own defined map files, from fstab to directory services. The auto_master file defines the locations and directories designated to be automounted. It also defines where the configuration of the maps comes from for each directory.

Direct, Indirect, and Executable Maps

Direct Maps

Direct maps allow an administrator to mount an export or share from a file server to a specific directory in Mac OS X. For example, if there are binary utilities on an NFS server exported from /shared/utilities on server suse.baranaba.com and the utilities need to be mounted on /opt/bin, a direct map can accomplish this as follows.

First, edit the /etc/auto_master file and add:

```
/-        auto_utilities
```

The first item, "/-", is a special mount point that tells autofs to use the mount point specified inside the mount map file (auto_utilities). Since the filename auto_utilities does not start with a forward slash, autofs will look for it in /etc. If the filename starts with a forward slash, autofs uses the absolute path. Create a file called /etc/auto_utilities and add this entry:

```
/opt/bin    suse.baranaba.com:/shared/utilities
```

This tells autofs to mount the export /shared/utilites from the server suse.baranaba.com to /opt/bin.

If the mount command was applied in the terminal, the map will appear on auto_utilities on /opt/bin:

```
map auto_utilities on /opt/bin (autofs, automounted)
```

The /opt/bin is now a trigger to read the map /etc/auto_utilities and mount the exports accordingly once the trigger is accessed. Another mount command after the directory is accessed will produce the following mount:

suse.baranaba.com:/shared on /opt/bin (nfs, nodev, nosuid, automounted)

Normally, direct maps are placed into a single map file, and autofs mounts and unmounts them as needed. If there are other folders in /opt, they will show up alongside bin. This is an important distinction from indirect maps.

Indirect Maps

Indirect maps allow administrators to mount specific exports or shares from different file servers into specific directories in Mac OS X. When autofs takes over the parent folder, the map specifies the directories, exports, and shares that are mounted in the directories within the parent folder. The exports or shares can be on different file servers in different locations, but each mount will appear as a folder within the specified mount point. This allows administrators to collect resources from many different file servers and give the user a single place to access them.

For example, imagine an administrator wants to put all of an organization's graphics libraries into a single folder. However, because the libraries are spread across multiple file servers, the files cannot simply be copied to each client machine. The clip art is on server1.example.com in a folder /Shared/clipart, the proofs are at server2.example.com in a folder called /sharing/documents/04proofs, and the stock photos are in server3.example.com in a folder called /sf2008.

To collect them into a folder in /Volumes/Resources on each client, the administrator would need to add a line to /etc/auto_master to add a new map:

```
/Volumes/Resources resource_map
```

Then create the file /etc/resource_map with the trigger folder and the export or share that would get mounted on top of it:

```
Clipart     server1.example.com:/Shared/clipart

Proofs      server2.example.com:/sharing/documents/04proofs

Stock       server3.example.com:/sf2008
```

Users would then have easy access to clip art, proofs, and stock photos without knowing that they are on different servers in different folders. The mounts would not be mounted until they were accessed. The structure would appear as follows:

```
/Volumes/Resources

/Volumes/Resources/Clipart

/Volumes/Resources/Proofs

/Volumes/Resources/Stock
```

Note that autofs controls the parent folder, and it is not possible to add non-autofs mount points to it. In the previous example, if the administrator wanted to create a local folder called Artwork in the same folder as Clipart, Proofs, and Stock, it would not be possible to create a folder within /Volumes/Resources.

Executable Maps

Executable maps allow administrators to specify a map file that does not contain a list of file servers and exports and shares to mount, but instead a script that is executed when the trigger folder is accessed. When the trigger folder is accessed, the script is called without any arguments. The script then returns the names of the mount points within the trigger folder. When a user navigates to the mount points, the name of the mount point is passed to the script, and the script returns the file server and path to the export or share. Autofs then mounts the export or share to the mount point, and the user has access to the files.

To turn a map file into an executable map, simply set the execute bit with chmod on the map. For example, create an executable map by adding the following line to /etc/auto_master:

```
/Volumes/dyno              auto_run
```

Create the file /etc/auto_run and add the following script to it:

```
#!/bin/bash

if [ $# = 0 ]; then # List keys

    echo my_stuff

    exit

fi

echo suse.baranaba.com:/opt
```

Change the execute bit on the map:

```
chmod +x /etc/auto_run
```

When the folder /Volumes/dyno is first accessed, autofs calls the script without any arguments, and it returns a mount point named my_stuff and then exits. When the folder /Volumes/dyno/my_stuff is accessed, the script is called with the name my_stuff, and the script returns suse.baranaba.com:/opt, which is mounted on /Volumes/dyno/my_stuff.

Executable maps are useful for scripting how maps are created and for bridging systems with different map information.

Wildcards and Submounts

Wildcard characters enable administrators to reduce the number of lines in a map file, which provides a useful shortcut. Wildcards match the name of the folder being navigated to by the user and use the folder name in the mount name. For example, with an indirect map defined in auto_master as:

```
/opt        auto_wildcard_example
```

A simple wildcard automount in /etc/auto_wildcard_example would be:

```
*   suse.baranaba.com:/&
```

The asterisk (*) is any folder that is accessed in /opt, and the ampersand (&) is the name of the folder accessed. The asterisk can only be specified in the mount point part of the automount, and the ampersand can only be specified in the remote system name or remote subdirectory field. If the directory /opt/abc is accessed, the map will substitute the abc for the asterisk and the ampersand, and the remote export suse.baranaba.com:/abc will be mounted on /opt/abc.

An ampersand can be used on its own, without an asterisk. For example:

```
bin suse.baranaba.com:/shared/&
```

This would specify that when the /opt/bin folder is accessed, suse.baranaba.com:/shared/bin would be mounted on /opt/bin.

It is possible to do multiple substitutions, for example, to map the following mount points with exports:

```
/opt/artwork       artwork.baranaba.com:/artwork
/opt/graphics      graphics.baranaba.com:/graphics
/opt/specs         specs.baranaba.com:/specs
```

An effective strategy would be to use an indirect map that maps the three directories to the three exports. Since the host names and the exports match the mounted directory name, it is possible to simply specify the map as follows:

```
*   &.baranaba.com:/&
```

It is also possible to specify submounts to define autofs file systems within autofs file systems. This allows one file to define the top-level structure and then delegate the second-level structure to other files (and, presumably, other administrators). For example, imagine that there are three departments: shipping, billing, and receiving. Each department wants to control its own map file to define the automounts under its department. Since the top level needs to be organized, define the auto_master map as:

```
/opt        auto_sub
```

and the auto_sub mount map as:

```
shipping    -fstype=autofs auto_shipping
billing     -fstype=autofs auto_billing
receiving   -fstype=autofs /var/auto_receiving
```

Each department could then define its own maps (/etc/auto_shipping for shipping, /etc/auto_billing for billing, and /var/auto_receiving for receiving). The top-level structure would be /opt/shipping, /opt/billing, and /opt/receiving, and as those directories are accessed, the department-level maps would be referenced.

It is also useful to pass variables between submount maps. Since a submount map only knows about its mount points, a variable could pass the name of the parent folder to use in forming the export or share. For example, if the parent map is defined as:

```
shipping    -fstype=autofs,-Dparent=/opt/&        auto_shipping
```

Inside auto_shipping, use ${parent} as a variable:

```
forms       suse.baranaba.com:${parent}/forms
```

or

```
forms       suse.baranaba.com:${parent}/&
```

There are some other prepopulated variables as well to use in the maps:

CPU = `uname -p`

HOST = `uname -n`

OSNAME = `uname -s`

OSREL = `uname -r`

OSVERS = `uname -v`

They can be used as follows:

```
/opt/bin suse.baranaba.com:/opt/bin/${OSNAME}
```

Mount Options

If mount options in the auto_master file for a map are specified, then these options will apply to each automount in the map unless other options are specified. For example, if the /etc/auto_master has the following entry:

```
/opt        auto_reference -ro
```

All automounts in the auto_reference map will be mounted read only unless an automount has other options.

Special Maps

To support earlier versions of Mac OS X, there are special maps that allow autofs to read fstab, a legacy system, usually file based, that lists mount points and file servers. Special maps also allow autofs to discover which folders are being exported by a specific file server and provide access to them.

fstab

Most administrators think of fstab as a file that lists static information about file systems to be mounted. While most administrators interact with it by creating a file called /etc/fstab, autofs receives information from fstab using the getfsent(3) API. This is important because Mac OS X provides fstab information through /etc/fstab from fstab entries in local directory services (configured in Directory Utility under Mounts) and in Open Directory (configured as automounts via Server Admin).

Mac OS X also adds an option to fstab entries called net. When this option is specified, the mount point in the fstab entry is ignored, and the trigger and resulting mount reside in the path specified in /etc/auto_master, usually /Network/Servers. This enables administrators to have the same fstab for multiple systems and allows Mac OS X to mount the export or share in a different location than the one specified in the fstab entry.

Autofs will also mount entries from fstab that are not specified with the net option. These entries are mounted at the mount point specified in the fstab entry (see “Static Automounts”).

For example, create an entry in /etc/fstab to mount a shared folder called shared in /mnt/suse/shared:

```
suse.baranaba.com:/shared /mnt/suse/shared nfs net 0 0
```

In the /etc/auto_master file, enable autofs to use fstab entries by using the special map type of -fstab:

```
/Network/Servers   -fstab
```

This will cause autofs to call the getfsent to get all of the fstab entries that have the net option, create a trigger at /Network/Servers/suse.baranaba.com/shared, and mount the share when it is accessed. Since autofs controls the parent folder and creates triggers for all the fstab entries that have the -net option, the -fstab special map is an indirect map.

Static Automounts

Static automounts are similar to fstab automounts. They both use getfsent(3) to read the fstab entries. However, while the -fstab option will only mount fstab entries with the -net option, static mounts will mount the entries in fstab that do not have the -net option. Note that static automounts are direct mounts.

For example, if the same fstab entry was specified without the -net option:

```
suse.baranaba.com:/shared /mnt/suse/shared nfs none 0 0
```

In /etc/auto_master, this is called a static mount type and is specified using the -static map type:

```
/-  -static
```

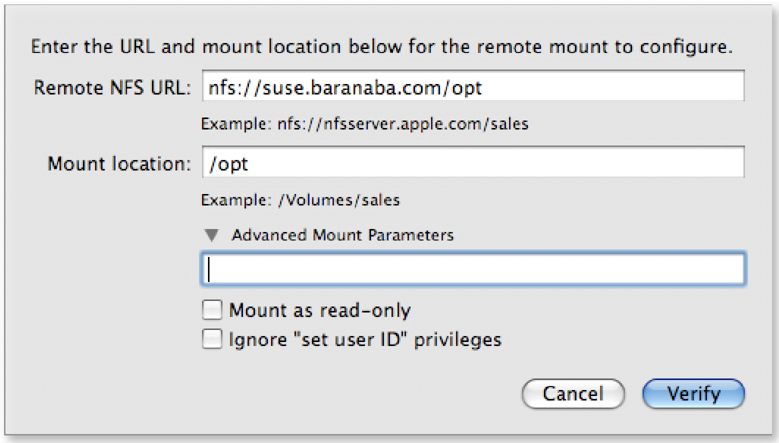
Note that the path is specified as "/-" since static map types are specified in the fstab entries. These are direct maps since each entry in fstab without the -net option gets mounted directly to the specified place in the local file system.

When static maps are set up by automount, the mount point is the trigger that causes the export or share to mount when it is first accessed.

Note that if a native autofs map and a legacy (-fstab or -static) map both mount a share or export to the same mount point, autofs will detect this and only mount the native map.

Directory Utility

Mac OS X also adds entries to fstab in Directory Utility. You can create a mount by following these steps:


Enter the URL and mount location below for the remote mount to configure.

Remote NFS URL: nfs://suse.baranaba.com/opt

Example: nfs://nfsserver.apple.com/sales

Mount location: /opt

Example: /Volumes/sales

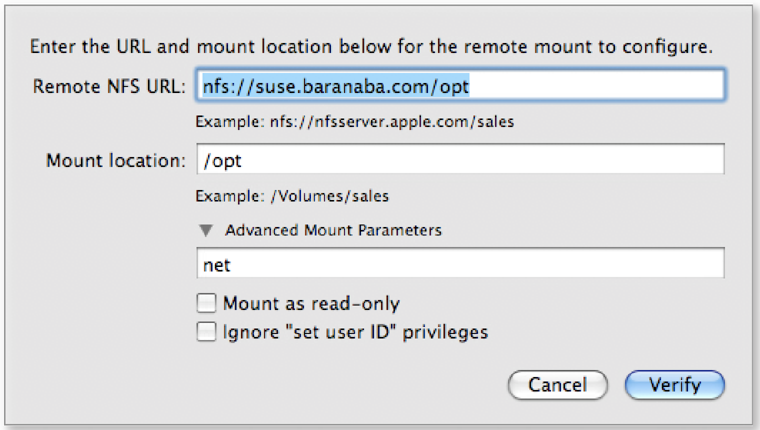
Advanced Mount Parameters

☐ Mount as read-only
☐ Ignore "set user ID" privileges

The NFS mount would be mounted in /opt as a static mount, after enabling static mounting in /etc/auto_master as:

```
/Network/Servers   -static
```

Here is how it looks with the net option mounted:



Enter the URL and mount location below for the remote mount to configure.

Remote NFS URL: nfs://suse.baranaba.com/opt

Example: nfs://nfsserver.apple.com/sales

Mount location: /opt

Example: /Volumes/sales

Advanced Mount Parameters

net

☐ Mount as read-only

☐ Ignore "set user ID" privileges

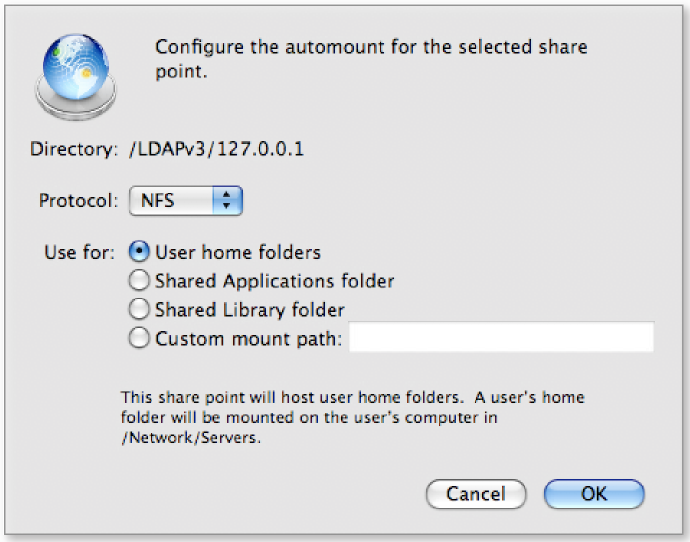
Even though the mount point is specified as /opt, the export would be mounted on /Network/Servers/suse.baranaba.com/opt, after specifying an auto_master entry as:

```
/Network/Servers   -fstab
```

The -fstab option tells autofs to look in fstab for all entries with the net option and mount them in /Network/Servers with the mount point as the name server and the subdirectory as the mount path.

Server Admin

The following steps outline how to specify an automount in Server Admin.



Configure the automount for the selected share point.

Directory: /LDAPv3/127.0.0.1

Protocol: NFS

Use for: [✓ User home folders] Shared Applications folder Shared Library folder Custom mount path:

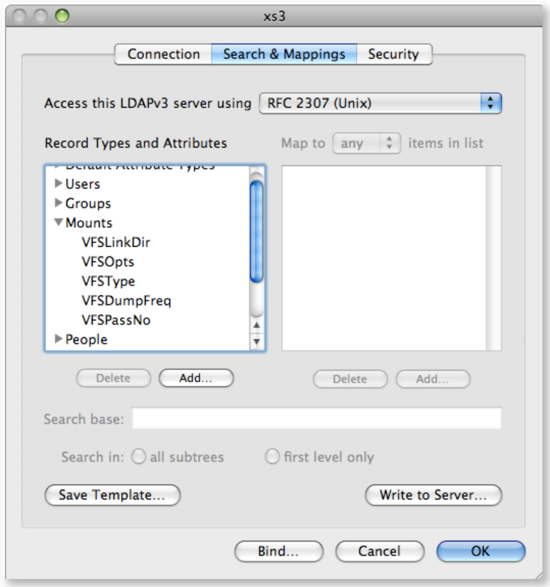
This share point will host user home folders. A user's home folder will be mounted on the user's computer in /Network/Servers.

The following table summarizes the options and the mount point when each of the options in Server Admin is selected.

Shared Applications Folder

Radio Button	-net Option?	Mount Type	Where Mounted
User home folder	Yes	-fstab	/Network/Servers/<server>/<path>
Shared Applications folder	No	-static	/Network/Applications
Shared Library folder	No	-static	/Network/Library
Customer mount path	No	-static	Path specified

For Mac OS X to have access to fstab entries in directory services, Search & Mappings is configured to map to the corresponding attributes in Open Directory (or another third-party LDAP server).



Access this LDAPv3 server using: RFC 2307 (Unix)

Record Types and Attributes
- Default Attribute Types
- Users
- Groups
- Mounts
  - VFSLinkDir
  - VFSOpts
  - VFSType
  - VFSDumpFreq
  - VFSPassNo
- People

Search base:

Search in: ○ all subtrees ○ first level only

This allows autofs to manage the exports or shares from fstab and is enabled by -fstab and -static options in auto_master. Note that it is not necessary to configure autofs for autofs to use the fstab entries stored in directory services. Specifying -static or -fstab will tell autofs to use getfsent(3) to access fstab entries.

To see the fstab entries, an entry was made in the /etc/fstab file, a mount was made in Directory Utility, and an automount was created in Server Admin. getfsent(3) was called three times in a c program (see fstabprint.c in the appendix), and the fstab entries were printed out:

```
-------------------------

fs_spec is suse.baranaba.com:/opt

fs_vfstype is nfs

fs_mntops is

fs_type is

fs_file is /opt

-------------------------

fs_spec is suse.baranaba.com:/shared

fs_vfstype is nfs

fs_mntops is net
```

```
fs_type is rw
fs_file is /mnt/1234
------------------------
fs_spec is xs3.baranaba.com:/Users
fs_vfstype is nfs
fs_mntops is
fs_type is
fs_file is /Network/Library
```

The first entry shows the fstab entry in /etc/fstab. The second entry shows the mount from Directory Utility (note the net option). The third entry is from Open Directory (note the absence of the net option since it is a shared Library folder).

Hosts Automounts

The hosts type of automount allows autofs to run showmount "-e" on each entry in the hosts database (usually in /etc/hosts) to discover shares exported by those servers. If auto_master contains:

```
/net                      -hosts
```

and the /etc/hosts file contains:

```
##
# Host Database
#
192.168.1.3 suse.baranaba.com
```

and suse.baranaba.com is exporting these shares:

```
# showmount -e suse.baranaba.com
Exports list on suse.baranaba.com:
/shared     192.168.1.0/255.255.255.0
/bin        192.168.1.0/255.255.255.0
/opt        192.168.1.0/255.255.255.0
/home       192.168.1.0/255.255.255.0
```

then the /net directory will contain:

```
bash-3.2# ls -R /net
suse.baranaba.com


/net/suse.baranaba.com:
bin home    opt     shared
```

Since Mac OS X contains entries for localhost and broadcast in the /etc/hosts file, entries for these will appear in /net. However, since these do not have shares exported, no shares will be triggered when they are accessed in /net. They can safely be ignored.

The hosts automount map is normally specified with the nobrowse option, so shares or exports that have not already been mounted will not appear (either with the ls terminal command or the Finder). Since this option is not currently supported, an entry for each host will appear in the /etc/hosts file.

File System Options

In the previous examples, all the file systems are NFS. However, an SMB or AFP automount can be specified by using the -fstype option. For SMB, the format would be:

```
smbshare  -fstype=smbfs ://username:password@winserver.example.
com/share
```

For AFP, it would be:

```
adserver  -fstype=afp afp://username:password@afpserver.example.
com/share
```

Note that the user name and password are only required if the share does not allow anonymous access, or if you want to mount it with escalated privileges. Since the user name and password are contained within the map file, make sure the map file is read and writable only by root (600).

Autofs currently does not access Kerberos tickets from either the computer account's or the user's Kerberos credentials, so the user name and password must be specified in the mount map to use authentication.

Note that shares or exports that are mounted with other mechanisms, such as via the Finder with Connect To Server or with command-line utilities such as mount_smbfs or mount_afp, are not managed by autofs.

Maps in Directory Services

Up to this point, all of the autofs maps and automounts have been stored on a local disk in /etc/auto_master and related /etc/<map name> files. To configure a large number of clients, it is necessary to either update the configuration files on all the clients or provide the maps via Open Directory in the legacy fstab attributes. However, autofs can read both the auto_master and maps directly from directory services.

Autofs Maps and Automounts via LDAP

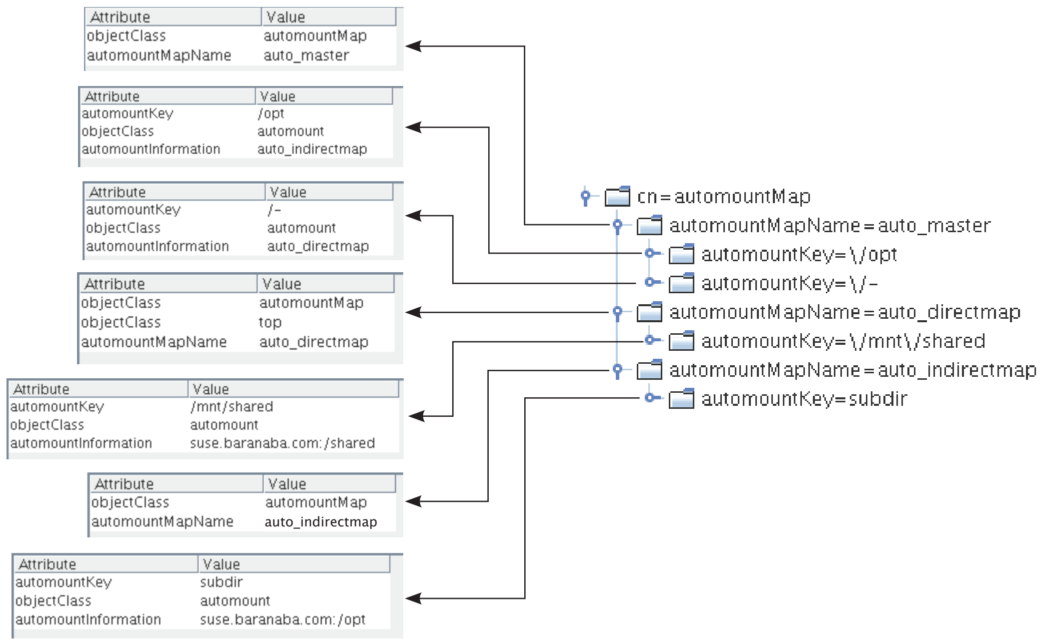
To access maps and automounts in LDAP, they need to be organized in a particular way within the LDAP-based directory service. While Mac OS X provides a template for mapping to RFC 2307bis-based schema, this is not required.

If you have automount maps accessible via an LDAP directory that uses the RFC 2307bis schema, the configuration is simple. Configure the LDAP server in Directory Utility, and the maps should work correctly. The RFC 2307bis schema is the most common schema for Solaris-type autofs mount maps.

Advanced LDAP Configuration

In addition to using the common RFC 2307bis schema, it is possible to use Directory Utility to map to any objectClasses and attributes that contain the map and automount information. The maps and automounts must be in a format similar to how the automounts and maps are organized in the local files. When autofs starts up and is configured to look at directory services for maps and automounts, the first thing it does is search for the master map (called auto_master in the stock configuration on Mac OS X, but it can be called anything). The master map is a specific objectClass (automountMap by default). Once the master map is discovered, it searches for all child objects of that master map by objectClass (automount by default). These child objects have the same role as the lines in /etc/auto_master and define the mount points and maps. Now that autofs knows the names of the maps, it searches directory services by the map name in the same way it discovered the master map. The child objects of each of these maps are the automount entries for that map (in the same way that the lines in each map file define an automount). The following diagram shows an auto_master, a direct map (auto_directmap), and an indirect map (auto_indirectmap).

The information is arranged hierarchically. The auto_master object (analogous to the /etc/auto_master file) is of objectClass automountMap, and each child object of it (analogous to the automount lines inside a map file) is of objectClass automount. The child objects of the auto_master object define the mount point and the name of the automount map. These automount maps are objects in the LDAP directory and have child objects of objectClass automount that define the automounts for the parent automount map.


cn=automountMap
automountMapName=auto_master
automountKey=\/opt
automountKey=\/-
automountMapName=auto_directmap
automountKey=\/mnt\/shared
automountMapName=auto_indirectmap
automountKey=subdir

Attribute	Value
objectClass	automountMap
automountMapName	auto_master

Attribute	Value
automountKey	/opt
objectClass	automount
automountInformation	auto_indirectmap

Attribute	Value
automountKey	/-
objectClass	automount
automountInformation	auto_directmap

Attribute	Value
objectClass	automountMap
objectClass	top
automountMapName	auto_directmap

Attribute	Value
automountKey	/mnt/shared
objectClass	automount
automountInformation	suse.baranaba.com:/shared

Attribute	Value
objectClass	automountMap
automountMapName	auto_indirectmap

Attribute	Value
automountKey	subdir
objectClass	automount
automountInformation	suse.baranaba.com:/opt

Configuring Mac OS X to Access Maps in LDAP

Now that the maps and automounts are in the LDAP-based directory service, the next step is to map the Mac OS X record types and attributes to the objectClasses in the LDAP-based directory service. Directory Utility provides a template that is compliant with RFC 2307bis, a nonratified RFC. Use of this type of mapping is not required but is a common mapping for autofs.

To configure Mac OS X to use automount maps in LDAP:

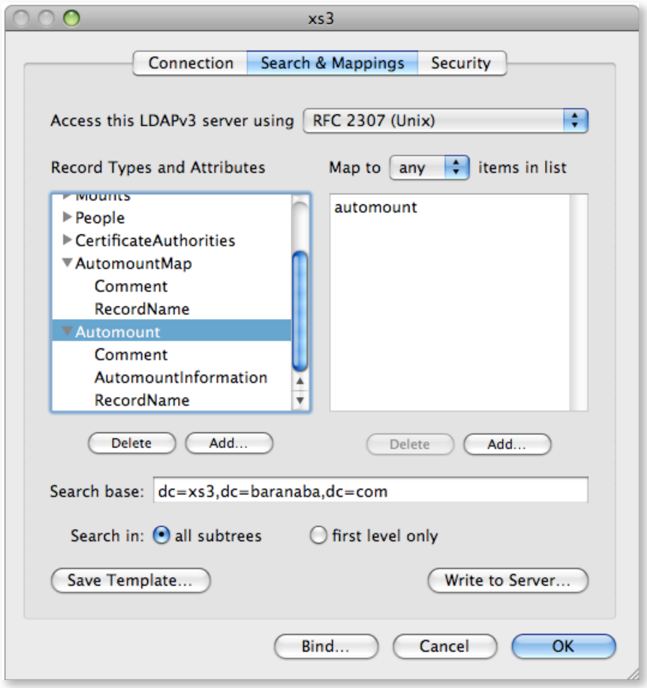
1. Open Directory Utility in /Applications/Utilities.
2. Choose Show Advanced Settings if they are not already displayed.
3. Select the Services button and select the LDAP v3 plug-in (if necessary, unlock the preference to select the plug-in).
4. Select the pencil icon to edit the settings.
5. Choose New to configure a new LDAP v3 server.
6. Enter a DNS name for the LDAP v3 server and click Continue.
7. Select the RFC2307 template and click Continue.
8. Click OK to finish binding. The new service should now be in the list of LDAP v3 servers bound to directory services.
9. Choose Edit… to edit the mappings.
10. Select the Search & Mappings tab.

The left side of Search & Mappings defines record types and attributes. The top level of the Record Types and Attributes section is the Record Type in directory services that gets mapped to a search base and objectClass in LDAP. For example, the AutomountMap record in directory services is mapped to automountMap objectClass in LDAP.

Once a map is discovered, the attributes under AutomountMap (Comment and RecordName) are mapped to the attributes on the right in LDAP (automountMapName and description), as shown in the following table. RecordName needs to be populated with the map name (for example, auto_master for the master map, or auto_directmap for a map). Comments are optional.

autofs	Directory Services	LDAP Attribute
Map name	RecordName	automountMapName

The automounts are called Automount in directory services and, by default, are mapped to objectClass automount:



Access this LDAPv3 server using: RFC 2307 (Unix)

Record Types and Attributes

- Mounts
- People
- CertificateAuthorities
- AutomountMap
  - Comment
  - RecordName
- Automount
  - Comment
  - AutomountInformation
  - RecordName

Map to any items in list

automount

Search base: dc=xs3,dc=baranaba,dc=com

Search in: [✓ all subtrees] first level only

Each automount contains a key and a location. Following is an example of how to specify an automount in a map file on disk:

homes suse.baranaba.com:/shared/homes

Directory services refers to the key as RecordName and the location as AutomountInformation. By default, RecordName is mapped to automountKey, and AutomountInformation is mapped to automountInformation in LDAP, as shown in the following table.

autofs	Directory Services	LDAP Attribute
Key	RecordName	automountKey
Location	AutomountInformation	automountInformation

These mappings are configured under the Automount Record Type in Directory Utility.

Setting Up Autofs to Use Directory Services

To configure autofs to use directory services for automount maps and automounts, specify "+" at the beginning of a line in the auto_master file or in a map file. For example, to grab entries from directory services that are a type of AutomountMap and have a RecordName of auto_master, specify the following in the /etc/auto_master configuration file:

```
+auto.master
```

This inserts the configuration from directory services at that point in the auto_master configuration. If the name of the auto master for the maps is auto.master (as is common in some setups), specify it as:

```
+auto.master
```

By default, this is at the top of the /etc/auto_master file. Note that if there are duplicate maps with the same RecordName, the ones specified first in the auto_master file are used. By specifying +auto_master first in /etc/auto_master, directory services maps can override any locally defined maps and automounts.

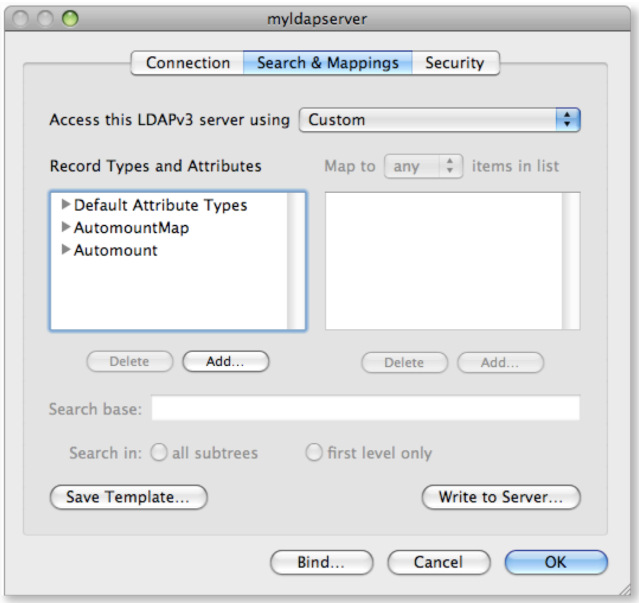
For example, if an LDAP-based directory had a direct and an indirect map, there would be three records in directory services of type AutomountMap: one called auto_master that refers to the other two automount maps. If the mappings in directory services are set up correctly, the automount map RecordNames can be viewed by using dscl:

```
> dscl localhost -ls /Search/AutomountMap

auto_directmap

auto_master

auto_indirectmap
```

NIS Schema Example

A very common setup for LDAP-based automount maps is to use the NIS schema. Since there is a template for the RFC 2307bis and not the NIS automount schema, a few modifications to the objectClasses and mappings are required. The following steps outline how the modifications can be made.

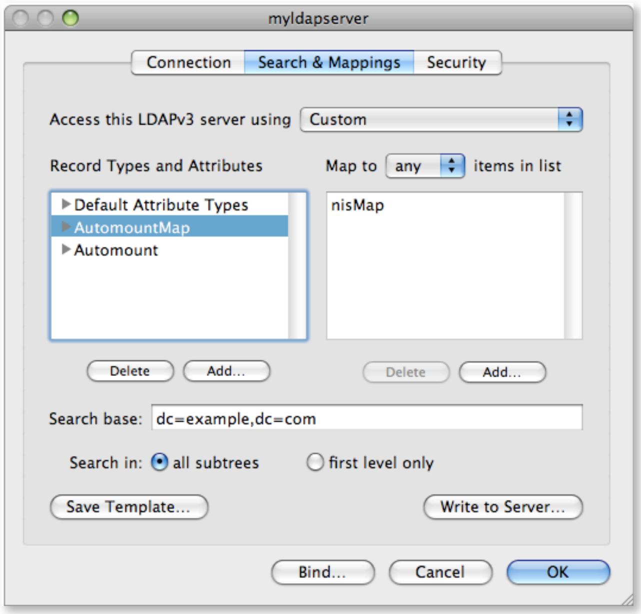
1. Open Directory Utility and select Advanced Settings.
2. Choose Services, select LDAPv3, and click the pencil to edit.
3. Select New, then Manual.
4. Name the configuration, enter the IP address of your LDAP v3 server that holds the automount maps, and select RFC 2307 (UNIX) as the LDAP Mappings.
5. Enter the Search Base Suffix for the LDAP server and click OK.
6. Select Edit.
7. Select the Search and Mappings tab and remove all records except for Default Attribute Types, AutomountMap, and Automount:



Access this LDAPv3 server using: Custom

Record Types and Attributes
- Default Attribute Types
- AutomountMap
- Automount

8. Select AutomountMap and change the objectClass from automountMap to nisMap:



Access this LDAPv3 server using: Custom

Record Types and Attributes — Map to any items in list

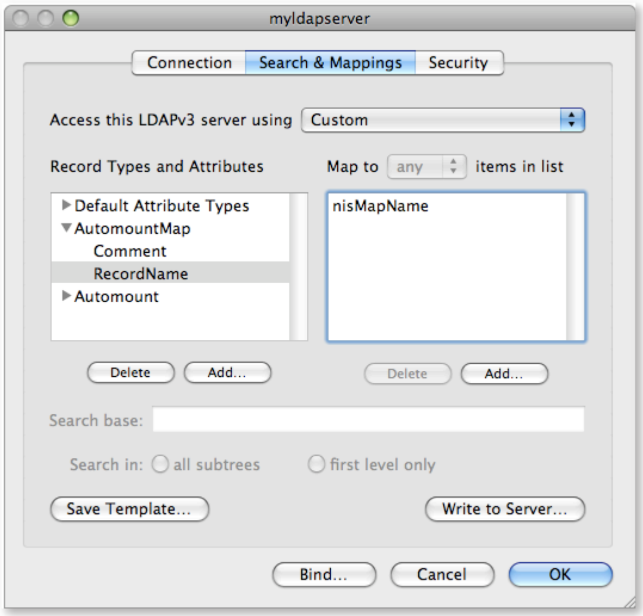
- Default Attribute Types
- AutomountMap
- Automount

nisMap

Search base: dc=example,dc=com

Search in: [✓ all subtrees] first level only

9. Select the RecordName attribute and map to nisMapName:



Access this LDAPv3 server using: Custom

Record Types and Attributes — Map to any items in list

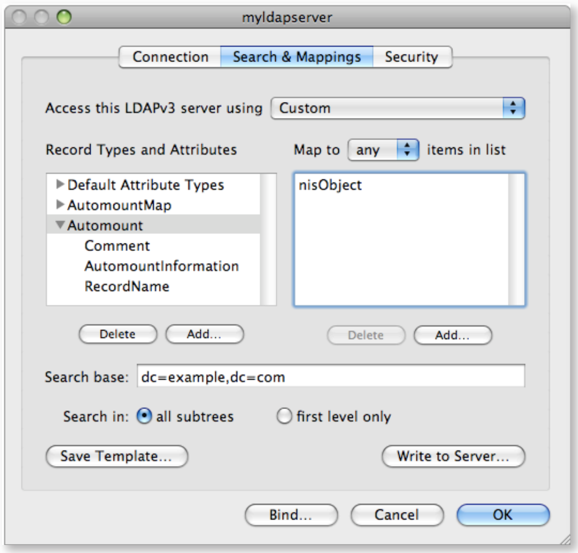
- Default Attribute Types
- AutomountMap
  - Comment
  - RecordName
- Automount

nisMapName

Search base:

Search in: all subtrees first level only

10. Select the Automount record type and map to nisObject:



Access this LDAPv3 server using: Custom

Record Types and Attributes
- Default Attribute Types
- AutomountMap
- Automount
  - Comment
  - AutomountInformation
  - RecordName

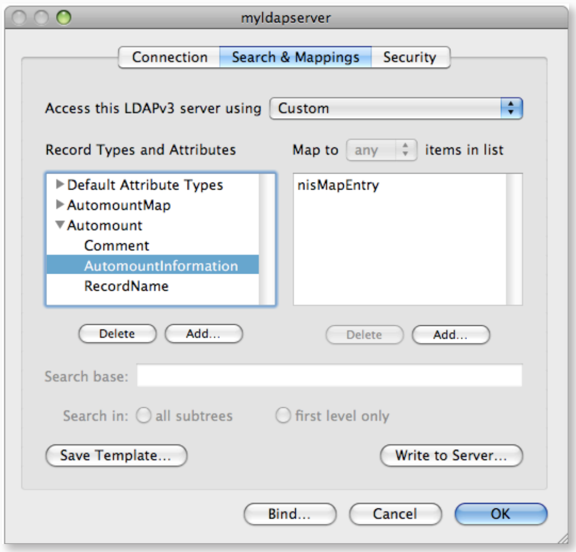
Map to: any items in list

nisObject

Search base: dc=example,dc=com

Search in: [✓ all subtrees] first level only

11. Select the AutomountInformation attribute under Automount and map to nisMapEntry:



Access this LDAPv3 server using: Custom

Record Types and Attributes
- Default Attribute Types
- AutomountMap
- Automount
  - Comment
  - AutomountInformation
  - RecordName

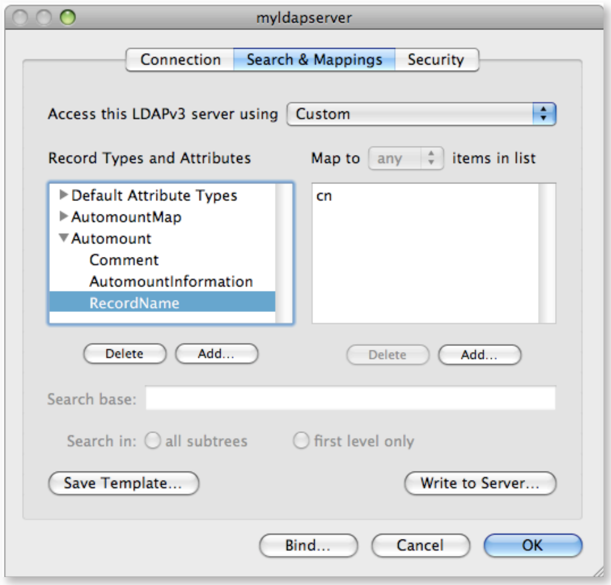
Map to: any items in list

nisMapEntry

Search base:

Search in: all subtrees first level only

12. Select the RecordName attribute under Automount and map to cn:



Access this LDAPv3 server using Custom

Record Types and Attributes

Map to any items in list

- Default Attribute Types
- AutomountMap
- Automount
  - Comment
  - AutomountInformation
  - RecordName

cn

Search base:

Search in: all subtrees first level only

13. Finally, verify that the maps are read correctly by running:

```
sudo automount -vc
```

Autofs in Mac OS X

Autofs in Mac OS X should work out of the box with autofs mount maps hosted in directory services. However, some older versions of autofs may have some differences that you should be aware of.

Dots Versus Underscores

If the names of your automount maps contain dots instead of underscores (for instance, auto.master), be sure to specify +auto.master instead of auto_master in /etc/auto_master.

automountInformation Format

Autofs automount objects in LDAP are expected to have an AutomountInformation with the map name (auto_static) and not the full DN (automountMapName=auto_static, dc=example,dc=com).

Wildcards

Earlier versions of autofs may have automount maps that contain "/" for wildcards. To use wildcards in autofs on Leopard, the wildcard must be specified as "*".

Finder Integration

Automounts do not show up directly in the sidebar of Mac OS X where other exports or shares are listed. Any -fstab entries (those specified with -net in fstab) can be accessed by selecting All under Shared in the sidebar and selecting Servers. This will show the automounts that are defined via the -fstab option.

The Finder can also access other folders that have automounts on them by navigating to them: Go > Go To Folder....

If the permissions get out of sync in the Finder and automounted directories, try flushing the directory services cache and the group membership cache and restarting the Finder:

```
sudo dscacheutil -flushcache
sudo dsmemberutil -flushcache
killall Finder
```

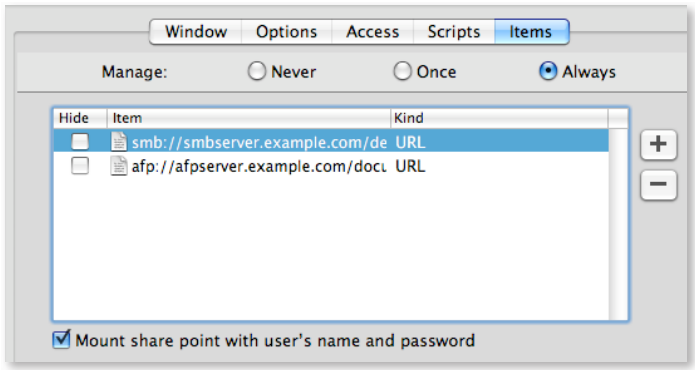
Kerberized NFS

Autofs supports automounting of Kerberized NFS exports. With Kerberized NFS, the connection itself is not authenticated, and every NFS operation has a set of credentials associated with it. Autofs will mount the Kerberized NFS export. When a user accesses the mount, that user will receive an NFS service ticket (assuming the user has a valid Kerberos TGT). Every transaction will then apply that user's credentials.

A Kerberized NFS mount can have multiple connections from multiple users, each using the correct user's credentials for each transaction. This allows administrators to support multiple users, each authenticated with their own credentials to the same mount point. This is very different from AFP and SMB mounts, which autofs handles by mounting shares as guests, and all operations sent over the connection are accomplished as guests on the server.

Other Ways to Mount

Autofs does not mount an export or share as the user who accesses the trigger. It also does not use Kerberos to authenticate mounts for SMB and AFP. Therefore, it is not appropriate to use automounting for all exports or shares. The most common way to mount AFP or SMB shares for specific users is to use Managed Client for Mac OS X (MCX) to specify a login preference that mounts the share point with the user's credentials. If the user is participating in Single Sign On, the exports or shares will be mounted with Kerberos authentication if supported by the file server.



Manage: Never Once [✓ Always]

Hide	Item	Kind
☐	smb://smbserver.example.com/de	URL
☐	afp://afpserver.example.com/doc	URL

[✓] Mount share point with user's name and password

User Home Directories

A common way to set up user network home directories is to create automounts in Mac OS X Server (as shown in "Server Admin"). Since autofs does not mount AFP or SMB shares using Kerberos or individual user credentials, the automounted home shares are unmounted by Login Window and remounted using the user's credentials from the Login Window. This allows the shares to be mounted with the correct credentials and permissions.

If a user logs in to Mac OS X via SSH and not via the console, Login Window is not available to unmount the automounted home directory and remount it as the user. If the user's home directory is on AFP, the user can run the mnthome command to unmount the automount and remount it using the user's Kerberos credentials.

Deep Mounts

It is possible to mount items underneath an SMB share, and not only limited to the top-level share. For example, 2000 project folders all shared via SMB could be accessed by:

```
smb://server.example.com/projects
```

If one of the projects was named "widgets" and was a folder within the project's share point, only that folder could be mounted just as if it were a share as shown in this indirect map:

```
widgets  -fstype=smbfs ://server.example.com/projects/widgets
```

Also, if the NFS server allows it, items can be mounted underneath an NFS export point.

Conclusion

Autofs provides a robust automounting system for Mac OS X. It allows organizations to leverage their current autofs automount maps used on Linux or Solaris to Mac OS X systems. Since the autofs file system on Mac OS X is multithreaded and dynamic, users will experience increased responsiveness, and administrators will notice fewer connections as unused mounts are dynamically disconnected. Autofs on Mac OS X provides streamlined integration into UNIX-based networks and gives users seamless access to network resources provided by autofs maps.

Glossary

Autofs: A robust automounter included with Mac OS X and Mac OS X Server. It allows administrators to configure file systems so that users can reach the files transparently.

Automount: Automatically mounts file systems when they are in use and then unmounts them after a period of inactivity.

Exports: Network File Systems that are shared out from other systems on the network.

fstab: A legacy system, usually file-based, that lists mount points and file servers.

Autofs maps: A collection of automounts related by mount point and organized in the same file or location in directory services. Autofs uses several types of maps, such as the master map, direct maps, indirect maps, and executable maps.

Master map: A master list specifying all the maps that autofs should check.

Direct autofs maps: Make a direct association between a mount point on the client and a directory on the server.

Indirect maps: Use a substitution value of a key to establish the association between a mount point on the client and a directory on the server.

Executable maps: Allow administrators to specify a map file that does not contain a list of file servers and exports or shares to mount, but rather contains a script that is executed when a trigger folder is accessed.

Mount point: The full pathname of a directory where a remote file system is mounted.

Trigger: A location on the local file system that causes the automount daemon to automatically mount a remote file system to the mount point located at the trigger.

Appendix: Test Source Code

fstab.c:

```
#include <fstab.h>
#include <stdio.h>

int main(int argv, char ** argc) {
         struct fstab *tab;
         while ((tab=getfsent())) {
                  printf("-------------------------\n");
                  printf("fs_spec is %s\n",tab->fs_spec);
                  printf("fs_vfstype is %s\n",tab->fs_vfstype);
                  printf("fs_mntops is %s\n",tab->fs_mntops);
                  printf("fs_type is %s\n",tab->fs_type);
                  printf("fs_file is %s\n",tab->fs_file);
         }
}
```