AppleShare X
Developer's Kit

AppleShare Client

Draft. Preliminary. April 14, 2000

Technical Publications

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

About This Manual

This manual describes the application programming interface for the AppleShare Client, which consists of functions for creating, parsing, and disposing of AFP Universal Resource Locators (URLs) and functions for creating and disposing of shared volume enumerator references and mounting shared volumes.

Conventions Used in This Manual

The Courier font is used to indicate server control calls, code, and text that you type. Terms that are defined in the glossary appear in boldface at first mention in the text. This guide includes special text elements to highlight important or supplemental information:

Note
Text set off in this manner presents sidelights or interesting points of information. ◆

IMPORTANT
Text set off in this manner—with the word Important—presents important information or instructions. ▲

▲ **WARNING**
Text set off in this manner—with the word Warning—indicates potentially serious problems. ▲

For More Information

The following books provide information that is important for all AppleShare developers:

- *AppleShare X Administrator's Manual*. Apple Computer, Inc.
- *Inside Macintosh*. Apple Computer, Inc.

For information about the programming interface for managing users and groups, see the following publication:

- *AppleShare X Developer's Kit*: *Directory Services.* Apple Computer, Inc.

For additional information on the Apple Filing Protocol (AFP), see the following publications:

- *AppleShare X Developer's Kit*: *Apple Filing Protocol.* Apple Computer, Inc.
- *Inside AppleTalk*, Second Edition. Apple Computer, Inc.

For information on user authentication modules (UAMs), see the following publication:

- *AppleShare X Developer's Kit*: *User Authentication Modules.* Apple Computer, Inc.

For information on controlling an AppleShare file server and handling server events, see the following publication:

- *AppleShare X Developer's Kit*: *Server Control Calls and Server Event Handlng.* Apple Computer, Inc.

For information on AppleShare IP Print Server security mechanisms, see the following publication:

- *AppleShare X Developer's Kit*: *AppleShare IP Print Server Security Protocol.* Apple Computer, Inc.

For information on using the AppleShare IP File Server 6.3 and Macintosh File Sharing, see the following manuals:

- *AppleShare Client User's Manual.* Apple Computer, Inc.
- *Macintosh Networking Reference.* Apple Computer, Inc.

AppleShare Client

This document describes the programming interface for AppleShare Client for Mac OS X. The programming interface provides functions for creating and managing Apple Filing Protocol (AFP) Universal Resource Locators (URLs) and for creating and managing shared volume enumerators.

Note
Your application should not call any of the functions described in this document at deferred task time. ◆

Verifying Library Information

The following functions allow your application to verify that the appropriate version of the AppleShare Client library is available:

- `AFPLibraryPresent` (page 9), which determines whether the AppleShare Client library is available.
- `AFPLibraryVersion` (page 10), which obtains the version of the AppleShare Client library.

The header file for the functions described in this section is `afpClient.h`, located in `/System/Library/Frameworks/AppleShareClientLibrary.framework/Headers`.

AFPLibraryPresent

Determines whether the AFP library is available.

```
Boolean AFPLibraryPresent (void);
```

function result The `AFPLibraryVersion` function returns `TRUE` if the AppleShare Client library is available and `FALSE` if the library is not available.

DISCUSSION

The `AFPLibraryPresent` function determines whether the AppleShare Client library is available.

AFPLibraryVersion

Obtains the version of the AppleShare Client library.

```
UInt32 AFPLibraryVersion (void);
```

function result The `AFPLibraryVersion` function returns an unsigned 32-bit value containing the version number. For Mac OS X, the version number is (TBD).

DISCUSSION

The `AFPLibraryVersion` function obtains the version number of the AppleShare Client library.

Using AFP URLs

This section describes functions that create, mount, verify, parse and dispose of AFP URLs. The functions are:

- `NewAFPURL` (page 11), which creates an AFP URL.
- `AFPMountURL` (page 12), which mounts an AFP URL.
- `IsAFPURL` (page 13), which determines whether a character string is a valid AFP URL.
- `ParseAFPURL` (page 14), which parses an AFP URL into its component parts.

- `DisposeAFPURL` (page 16), which disposes of an AFP URL.

The header file for the functions described in this section is `afpURL.h`, located in `/System/Library/Frameworks/AppleShareClientLibraryCore.framework/Headers`.

NewAFPURL

Creates an AFP URL.

```
char * NewAFPURL (StringPtr protocolName,
                  StringPtr serverNameOrHost
                  StringPtr zoneNameOrNULL,
                  StringPtr uamName
                  StringPtr userName
                  StringPtr password,
                  StringPtr volume,
                  StringPtr path);
```

`protocolName` On input, a value of type `StringPtr` that points to a string containing the transport protocol. Specify `at` for AppleTalk or `ip` for TCP/IP. If `protocolName` is `NULL`, TCP/IP is assumed.

`serverNameOrHost`
On input, a value of type `StringPtr` that points to a string containing the name or address of the computer that hosts the URL that is being created. The name can be a Network Bind Protocol (NBP) name, a Domain Name System (DNS) name, or an Internet Protocol (IP) address.

`zoneNameOrNull`
On input, a value of type `StringPtr` that points to a string containing the AppleTalk zone in which the computer that hosts the URL resides, or `NULL` if the computer does not reside in an AppleTalk zone.

`uamName` On input, a value of type `StringPtr` that points to a string `containing the name of the user authentication module (UAM)` that is to be used to authenticate the user specified by the `userName` parameter.

`userName`	On input, a value of type `StringPtr` that points to a string containing the user name that is to be authenticated.
`password`	On input, a value of type `StringPtr` that points to a string containing the password that is to be used to authenticate the user specified by the `userName` parameter.
`volume`	On input, a value of type `StringPtr` that points to a string containing the volume that is to be mounted if authentication is successful.
`path`	On input, a value of type `StringPtr` that points to a string containing the pathname for a particular directory or file on the volume specified by the `volume` parameter. The path should use the forward slash character (`/`) to delimit the directory and filename components of the path.
function result	The `AFPNewURL` function returns a pointer to the character string that contains the new AFP URL.

DISCUSSION

The `NewAFPURL` function creates an AFP URL that can be used by the Network Browser. A properly formatted AFP URL contains all of the information needed to authenticate a user on a particular server, including the transport protocol, the server name, the zone name (if the transport protocol is AppleTalk), the user authentication module that is to be used to authenticate the user, the user name and his or her password, and the volume that is to be mounted.

AFPMountURL

Mounts an AFP URL.

```
OSStatus AFPMountURL(const char* inURL,
                     const char* inMountPoint,
                     UInt32 inMountFlags,
                     UInt32 inAltFlags);
```

`inURL`	On input, a pointer to a character string containing the URL to mount. The value pointed to by `inURL` must be a fully qualified URL, including the user name and password, and (optionally) the user authentication method to use.
`inMountPoint`	On input, a pointer to a character string containing the mount point path.
`inMountFlags`	On input, a value of type `UInt32` containing mount flags. The mount flags are the same flags that are used for the mount system call.
`inAltFlags`	On input, a value of type `UInt32` alternate mount flags.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPMountURL` function mounts the URL pointed to by `inURL` on the mount point pointed to by `inMountPoint`.

Note
Calling the `AFPMountURL` function does not cause any dialog boxes to be displayed. ◆

IsAFPURL

Verifies an AFP URL.

```
Boolean IsAFPURL (char * url);
```

`url`	On input, a pointer to a character string that contains an AFP URL previously created by calling `NewAFPURL` (page 11).
function result	The `IsAFPURL` function returns `TRUE` if the `url` parameter points to an AFP URL and `FALSE` if it does not.

DISCUSSION

The `ISAFPURL` function verifies that a character string is a properly formatted AFP URL.

ParseAFPURL

Parses an AFP URL.

```
OSStatus ParseAFPURL (char * url,
                      StringPtr protocolName,
                      StringPtr serverNameOrHost
                      StringPtr zoneNameOr,
                      StringPtr uamName
                      StringPtr userName
                      StringPtr password,
                      StringPtr volume,
                      StringPtr path);
```

`url` On input, a pointer to a character string containing an AFP URL previously created by calling `NewAFPURL` (page 11).

`protocolName` On input, a value of type `StringPtr` that points to a string that is long enough to hold a value of type `Str255`. On output, `protocolName` contains the protocol name obtained from the AFP URL specified by the `url` parameter, or is `NULL` if `url` was not created with a protocol name.

`serverNameOrHost`
On input, a value of type `StringPtr` that points to a string that is long enough to hold a value of type `Str255`. On output, `serverNameOrHost` contains the server or host name obtained from the AFP URL specified by the `url` parameter, or is `NULL` if `url` was not created with a server or host name.

`zoneNameOr`
On input, a value of type `StringPtr` that points to a string that is long enough to hold a value of type `Str255`. On output, `zoneNameOr` contains the zone name obtained from the AFP URL specified by the `url` parameter, or is `NULL` if `url` was not created with a zone name.

uamName	On input, a value of type StringPtr that points to a string that is long enough to hold a value of type Str255. On output, uamName contains the UAM name obtained from the AFP URL specified by the url parameter, or NULL if url was not created with the name of a UAM.
userName	On input, a value of type StringPtr that points to a string that is long enough to hold a value of type Str255. On output, userName contains the user name obtained from the AFP URL specified by the url parameter, or NULL if url was not created with a user name.
password	On input, a value of type StringPtr that points to a string that is long enough to hold a value of type Str255. On output, password contains the password obtained from the AFP URL specified by the url parameter, or NULL if url was not created with a password.
volume	On input, a value of type StringPtr that points to a string that is long enough to hold a value of type Str255. On output, volume contains the volume name obtained from the AFP URL specified by the url parameter, or NULL if url was not created with a volume name.
path	On input, a value of type StringPtr that points to a string that is long enough to hold a value of type Str255. On output, path contains the path obtained from the AFP URL specified by the url parameter, or NULL if url was not created with a path.
function result	A result code. For a list of possible result codes, see “Result Codes” (page 33).

DISCUSSION

The ParseAFPURL function obtains the values that were used to create an AFP URL.

DisposeAFPURL

Disposes of an AFP URL.

```
void DisposeAFPURL (char * url);
```

`url` On input, a pointer to a character string that contains an AFP URL previously created by calling `NewAFPURL` (page 11).

function result None.

DISCUSSION

The `DisposeAFPURL` function releases memory associated with an AFP URL. Your application should call `DisposeAFPURL` when an AFP URL is no longer needed.

Using Shared Volumes Enumerators

This section describes the functions that an application uses to work with shared volumes enumerators. The functions are:

- `AFPCreateSharedVolumesEnumerator` (page 17), which creates a shared volumes enumerator reference from a server name and a server zone.
- `AFPCreateSVEFromAddress` (page 20), which creates a shared volumes enumerator reference from an address stored in a `sockaddr` structure.
- `AFPGetIndexedSharedVolume` (page 23), which obtains the name of a shared volume by its index number.
- `AFPSortSharedVolumes` (page 24), which sorts the list of volumes in a shared volume enumerator reference.
- `AFPMountSharedVolume` (page 24), which mounts a shared volume.
- `AFPMountSharedVolumeMP` (page 25), which mounts a shared volume.
- `AFPGetLoginInformation` (page 27), which obtains the log on type (Guest or registered user). If the user is a registered user, `AFPGetLoginInformation` also obtains the user's name and password.

- AFPGetMountAtStartup (page 28), which obtains the startup mounting state of a shared volume.
- AFPSetMountAtStartup (page 28), which sets the startup mounting state of a shared volume.
- AFPChangePassword (page 29), which sets the startup mounting state of a shared volume.
- AFPDeleteSharedVolumesEnumerator (page 30), which disposes of a shared volume enumerator reference created by AFPCreateSharedVolumesEnumerator (page 17) or AFPCreateSVEFromAddress (page 20).

The header file for the functions described in this section is afpClient.h, located in /System/Library/Frameworks/AppleShareClientLibraryCore.framework/Headers.

AFPCreateSharedVolumesEnumerator

Uses a server name and zone to create a shared volumes enumerator reference.

```
OSStatus AFPCreateSharedVolumesEnumerator (StringPtr serverName,
                    StringPtr serverZone,
                    StringPtr uamName,
                    StringPtr userName,
                    StringPtr password,
                    AShareEventUPP callback,
                    void * evtContext,
                    ATFilterUPP filter,
                    void * filterParam,
                    ATNotifyUPP notifier,
                    void * contextPtr;
                    AFPSharedVolumesEnumeratorRef * Ref);
```

serverName — On input, a value of type StringPtr that points to a string containing the name of the AppleShare server for which the shared volume enumerator reference is being created. For TCP/IP connections, serverName should be the DNS name of the server.

`serverZone` On input, a value of type `StringPtr` that points to a string containing the name of the AppleTalk zone in which the server specified by `serverName` resides. For TCP/IP connections, set `serverZone` to `NULL`. If the user logs on to a server using AppleTalk as the transport protocol, the enumerator reference created by `AFPCreateSharedVolumesEnumerator` is updated with the name of the zone in which `serverName` resides.

`uamName` On input, a value of type `StringPtr` that points to a string containing the name of the UAM to use when authenticating the user identified by the `userName` parameter, or `NULL`. If `uamName` is `NULL` and if the user provides a name in the log on dialog box, that is displayed by calling `AFPGetSharedVolumesCount` (page 22) the enumerator reference is updated with the name of the UAM that authenticated the user.

`userName` On input, a value of type `StringPtr` that points a string containing the name of the user to authenticate, or `NULL`. If `userName` is `NULL` and if the user enters a name in the log on dialog box that is displayed by calling `AFPGetSharedVolumesCount` (page 22), the enumerator reference is updated with the name that the user entered.

`password` On input, a value of type `StringPtr` that points to a string containing the password that is to be used to authenticate the user name specified by the `userName` parameter, or `NULL`. If `password` is `NULL` and if the user enters a password in the log on dialog box that is displayed by calling `AFPGetSharedVolumesCount` (page 22), the enumerator reference is updated with the password that the user entered.

`callback` On input, a value of type `AShareEventUPP` that points to an application-defined system event callback routine (page 32) that handles events that occur while `AFPGetSharedVolumesCount` (page 22) or other AppleShare Client library functions display dialog boxes, or `NULL`. If `callback` is `NULL`, the calling application will not receive update events while these dialog boxes are displayed.

`evtContext` On input, an untyped pointer to arbitrary data that `AFPCreateSharedVolumesEnumerator` passes to the application-defined system event callback routine specified by

	callback, or NULL. Your application can use evtContext to associate the invocation of your system event callback routine with any particular enumerator reference.
filter	On input, a value of type ATFilterUPP that points an optional application-defined filter routine (page 32) that can be used to control the volumes that are included in the enumerator reference, or NULL to match all volumes.
filterparam	On input, an untyped pointer to arbitrary data that AFPCreateSharedVolumesEnumerator passes to the filter routine specified by filter, or NULL. Your application can use filterparam to associate the invocation of your filter routine with any particular enumerator reference.
notifier	On input, a value of type ATNotifyUPP that points to an application-defined notification routine (page 31) that is to be called when address resolution for serverName is complete, or NULL if your application does not provide a notification routine.
contextPtr	On input, an untyped pointer to arbitrary data that AFPCreateSharedVolumesEnumerator passes to the notification routine specified by the notifier parameter. Your application can use contextPtr to associate the invocation of your notification routine with any particular enumerator reference.
ref	On input, a value of type AFPSharedVolumesEnumerator. On output, ref points to the enumerator reference created by AFPCreateSharedVolumesEnumerator.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The AFPCreateSharedVolumesEnumerator function creates a shared volumes enumerator reference that can be passed as a parameter to AFPGetSharedVolumesCount, AFPGetIndexedSharedVolume, AFPSortSharedVolumes, and AFPMountSharedVolumes or AFPMountSharedVolumesMP if the enumerator reference was created for a TCP/IP connection.

Passing the enumerator reference to AFPGetSharedVolumesCount (page 22) obtains the number of volumes that the user has permission to mount.

Passing the enumerator reference and an index number to `AFPGetIndexedSharedVolume` (page 23) obtains the name of the volume that is associated with the specified index number.

Passing the enumerator reference to `AFPSortSharedVolumes` (page 24) returns a sorted list of volume names. If your application needs to allow the user to select one or more volumes for mounting, it can display the sorted list in a dialog box.

Passing the enumerator reference and the name of a volume to be mounted to `AFPMountSharedVolume` (page 24) causes the specified volume to be mounted.

Passing the enumerator reference, the name of a volume to be mounted, a password for the volume, and mount flags to `AFPMountSharedVolumeMP` (page 25) causes the specified volume to be mounted.

When you no longer need the enumerator reference, call `AFPDeleteSharedVolumesEnumerator` (page 27) to deallocate the memory that has been allocated to it.

AFPCreateSVEFromAddress

Uses a `sockaddr` structure to create a shared volumes enumerator reference.

```
OSStatus AFPCreateSVEFromAddress(AddressPtr serverAddress,
                    StringPtr uamName,
                    StringPtr userName,
                    StringPtr password,
                    AFPSharedVolumesEnumeratorRef * ref);
```

`serverAddress` On input, a value of type `AddressPtr` that points to a `sockaddr` structure containing the address of the server for which the shared volumes enumerator is to be created.

`uamName` On input, a value of type `StringPtr` that points to a string containing the name of the UAM to use when authenticating the user identified by the `userName` parameter, or `NULL` if authentication is not required.

`userName` On input, a value of type `StringPtr` that points a string containing the name of the user to authenticate, or `NULL` if a user name is not required.

password	On input, a value of type StringPtr that points to a string containing the password that is to be used to authenticate the user name specified by the userName parameter, or NULL if a password is not required.
ref	On input, a pointer to a value of type AFPSharedVolumesEnumeratorRef. On output, ref contains a shared volumes enumerator that pass to AFPMountSharedVolume (page 24).
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The AFPCreateSVEFromAddress function creates a shared volumes enumerator for the server identified by the serverAddress parameter that can be passed as a parameter to AFPGetSharedVolumesCount, AFPGetIndexedSharedVolume, AFPSortSharedVolumes, and AFPMountSharedVolumes or AFPMountSharedVolumesMP.

Passing the enumerator reference to AFPGetSharedVolumesCount (page 22) obtains the number of volumes that the user has permission to mount.

Passing the enumerator reference and an index number to AFPGetIndexedSharedVolume (page 23) obtains the name of the volume that is associated with the specified index number.

Passing the enumerator reference to AFPSortSharedVolumes (page 24) returns a sorted list of volume names. If your application needs to allow the user to select one or more volumes for mounting, it can display the sorted list in a dialog box.

Passing the enumerator reference to AFPMountSharedVolume (page 24) and the name of a volume causes the specified volume to be mounted.

Passing the enumerator reference, the name of a volume to be mounted, a password for the volume, and mount flags to AFPMountSharedVolumeMP (page 25) causes the specified volume to be mounted.

When you no longer need the enumerator reference, call AFPDeleteSharedVolumesEnumerator (page 27) to deallocate the memory that has been allocated to it.

AFPGetSharedVolumesCount

Obtains the number of shared volumes that the user has permission to mount.

```
OSStatus AFPGetSharedVolumesCount (
                    AFPSharedVolumesEnumeratorRef ref,
                    Boolean * allfound,
                    UInt32 * count);
```

`ref` On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20)that represents an AppleShare server.

`allfound` On input, a pointer to a Boolean value. On output, `allfound` points to a value that is `TRUE` if all volumes have been counted and that is `FALSE` if `AFPGetSharedVolumesCount` is still counting. If `allfound` is `FALSE`, call `AFPGetSharedVolumesCount` again until `allfound` is `TRUE`.

`count` On input, a pointer to an unsigned 32-bit integer. On output, `count` points to a value that contains the current count of the number of volumes the user has permission to mount.

function result A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPGetSharedVolumesCount` function returns the number of volumes that a the user has permission to mount. Once an application obtains the number of volumes that the user has permission to mount, it can call `AFPGetIndexedSharedVolume` (page 23) to obtain the name of each volume by its index number.

If the shared volume enumerator reference specified by `ref` does not contain a user name or password, `AFPGetSharedVolumesCount` causes a log on dialog box to be displayed. The log on dialog box allows the user to log in as Guest or as a registered user with an optional password. After the user enters this information, the enumerator reference is updated with log on type (Guest or registered user) and the name and password (if any) the user entered.

AFPGetIndexedSharedVolume

Obtains the name of a shared volume by its index number.

```
OSStatus AFPGetIndexedSharedVolume (AFPSharedVolumesEnumeratorRef ref,
                    OneBasedIndex index,
                    StringPtr volumeName);
```

`ref`	On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20) that represents the AppleShare server that shares the volume whose name is to be obtained.
`index`	On input, a value of type `OneBasedIndex` that specifies the index number. Call `AFPGetSharedVolumesCount` (page 22) to determine the highest valid value of `index`. The lowest value of `index` is 1.
`volumeName`	On input, a value of type `StringPtr`. On output, `volumeName` points to the name of the volume that corresponds to the specified index value.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPGetIndexedSharedVolume` function obtains the name of a volume by its index number. To determine the highest possible index number, call `AFPGetSharedVolumesCount` (page 23).

Once you obtain the name of a volume, you can sort the list of volume names by calling `AFPSortSharedVolumes` (page 24) and you can mount a particular volume by calling `AFPMountSharedVolume` (page 24) or `AFPMountSharedVolumeMP` (page 25) if the enumerator reference was created for a TCP/IP connection.

AFPSortSharedVolumes

Sorts the names of shared volumes.

```
OSStatus AFPSortSharedVolumes (AFPSharedVolumesEnumeratorRef ref);
```

`ref` On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20).

function result A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPSortSharedVolumes` function sorts the list of volumes in a shared volume enumerator reference using the binary presentation of the characters that make up each volume name in ascending order.

AFPMountSharedVolume

Mounts a shared volume.

```
OSStatus AFPMountSharedVolume (AFPSharedVolumesEnumeratorRef ref,
                    Str255 volumeName,
                    short * volumeRefNum,
                    Boolean * isMounted);
```

`ref` On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPMountSharedVolumeMP` (page 25) if the enumerator reference was created for a TCP/IP connection.

`volumeName` On input, a value of type `Str255` that specifies the name of the volume that is to be mounted. Call `AFPGetIndexedSharedVolume` (page 23) to obtain the volume name.

`volumeRefNum`	On input, a pointer to a value of type `short`. On output, `volumeRefNum` points to a unique volume reference number that your application can use to refer to the volume when it sends AFP commands to the server.
`isMounted`	On input, a pointer to a Boolean whose value is `TRUE` if your application wants `AFPMountSharedVolumes` to return an error if the volume is already mounted. Set `isMounted` to `NULL` if you don't want `AFPMountSharedVolumes` to return an error if the volume is already mounted.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPMountSharedVolumes` function mounts the specified shared volume. If a volume is already mounted and if the `isMounted` parameter is `TRUE`, `AFPMountSharedVolumes` returns an error.

AFPMountSharedVolumeMP

Mounts a shared volume.

```
AFPMountSharedVolumeOnMP(AFPSharedVolumesEnumeratorRef  ref,
                    StringPtr inVolumeName,
                    const UInt8* inVolPassword,
                    const char* inMountPoint,
                    UInt32 inMountFlags,
                    UInt32 inAltFlags,
                    Boolean inMakeUnique,
                    UInt32 inMaxPath,
                    char* outMountPath);
```

`ref`	On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSVEFromAddress` (page 20).

inVolumeName	On input, a value of type StringPtr that points to a string containing the name of the volume that is to be mounted. Call AFPGetIndexedSharedVolume (page 23) to obtain the volume name.
inVolPassword	On input, a pointer to a value of type UInt8 containing the volume's password.
inMountPoint	On input, a pointer to a character string containing the mount point path.
inMountFlags	On input, a value of type UInt32 containing mount flags. The mount flags are the same flags that are used for the mount(2) system call.
inMountFlags	On input, a value of type UInt32 containing alternate mount flags. The alternate mount flags are the same flags that are used for the mount(2) system call.
inMakeUnique	On input, a pointer to a Boolean value. If inMakeUnique is TRUE and another file system is already mounted on the path pointed to by inMountPoint, the volume is mounted using a algorithm that creates a unique mount point path name. If inMakeUnique is FALSE, AFPMountSharedVolumeMP fails of a file system is already mounted on the path pointed to by inMountPoint.
inMaxPath	On input, a value of type UInt32 that specifies the maximum length of outMountPath.
outMountPath	On input, a pointer to a character string containing the mount path for the mounted volume.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The AFPMountSharedVolumeMP function mounts the shared volume represented by ref.

AFPGetLoginInformation

Obtains log on information from a shared volume enumerator reference.

```
OSStatus AFPGetLoginInformation (
                    AFPSharedVolumesEnumeratorRef ref,
                    Boolean * isGuest,
                    Str255 userName,
                    Str255 password);
```

`ref`	On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20) that has been used to log a user into an AppleShare server.
`isGuest`	On input, a pointer to a Boolean value. On output, `isGuest` points to a value that is `TRUE` if the user chose to log on as Guest or `FALSE` if the user chose to log on as a registered user.
`userName`	On input, a value of type `Str255`. On output, `userName` contains the name the user typed in the Name text box of the log on dialog box displayed by `AFPGetSharedVolumesCount`.
`password`	On input, a value of type `Str255`. On output, `password` contains the password (if any) the user typed in the Password text box of the log on dialog box displayed by `AFPGetSharedVolumesCount`.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPGetLoginInformation` obtains log on information from an enumerator reference that has been used to log a user on to an AppleShare server.

AFPGetMountAtStartup

Obtains a volume's startup mount information.

```
OSStatus AFPGetMountAtStartup (
                    AFPSharedVolumesEnumeratorRef * ref,
                    StringPtr volumeName);
```

`ref` On input, a pointer to a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20).

`volumeName` On input, a value of type `StringPtr` that points to the name of the volume.

function result A result code whose value is `noErr` if the volume is set to be mounted at startup and whose value is `nsvErr` if the volume is not set to be mounted at startup.

DISCUSSION

The `AFPGetMountAtStartup` function obtains the startup mount information for a volume as set in the specified shared volume enumerator reference.

AFPSetMountAtStartup

Sets a volume's startup mount information.

```
OSStatus AFPSetMountAtStartup (
                    AFPSharedVolumesEnumeratorRef * ref,
                    StringPtr volumeName,
                    Boolean toMount);
```

`ref` On input, a pointer to a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20) that identifies the volume that is to be set.

`volumeName`	On input, a value of type `StringPtr` that points to the name of the volume whose startup mount information is to be set.
`toMount`	On input, a Boolean whose value is `TRUE` if the volume is to be mounted when the computer starts up or `FALSE` if the volume is not to be mounted at startup.
function result	A result code whose value is `noErr` if the volume's startup mounting information was successfully set. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPSetMountAtStartup` function updates the specified shared volume enumerator reference with the latest startup mount information for a volume.

AFPChangePassword

Changes the specified password.

```
AFPChangePassword (AFPSharedVolumesEnumeratorRef * ref,
                   StringPtr oldPassword,
                   StringPtr newPassword);
```

`ref`	On input, a pointer to a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17) or `AFPCreateSVEFromAddress` (page 20).
`oldPassword`	On input, a value of type `StringPtr` that points to a string containing the password that is to be changed.
`newPassword`	On input, a value of type `StringPtr` that points to a string containing the password that is to be set.
function result	A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPChangePassword` function changes the specified password.

AFPDeleteSharedVolumesEnumerator

Disposes of a shared volume enumerator reference.

```
OSStatus AFPDeleteSharedVolumesEnumerator (
                    AFPSharedVolumesEnumeratorRef * ref);
```

`ref` On input, a pointer to a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 17).

function result A result code. For a list of possible result codes, see "Result Codes" (page 33).

DISCUSSION

The `AFPDeleteSharedVolumesEnumerator` function disposes of a shared volume enumerator reference and deallocates memory that has been allocated for it. You should dispose of the enumerator reference as soon as it has fulfilled its purpose of mounting shared volumes.

The enumerator reference maintains an open session with the AppleShare server that is separate from sessions for any of the AppleShare server's volumes that have been mounted. Disposing of the enumerator reference closes this session.

Note
The `AFPDeleteSharedVolumesEnumerator` function deallocates memory, so your application should call it during main event time. ◆

AppleShare Client Application-Defined Routines

This section describes three application-defined routines that your application can provide when it calls `AFPCreateSharedVolumesEnumerator` (page 17):

- A notification callback routine that is called when the host name specified as a parameter to `AFPCreateSharedVolumesEnumerator` is resolved into an IP address.
- A filter callback routine that is called to control the display of volume names.

- A system event callback routine that is called when update events occur while an AppleShare Client function displays a dialog box.

Notification Callback Routine

Your notification callback routine is called when the host name specified as an parameter to `AFPCreateSharedVolumesEnumerator` (page 17) is resolved into an IP address. This is how you would declare your notification callback routine if you were to name it `MyURLNotifyUPP`:

```
OSStatus MyURLNotifyUPP (void* userContext,
                         ATEventCode code,
                         OSStatus result,
                         void *cookie);
```

`userContext` An application-defined value that your application previously passed as the `contextPtr` parameter when it called `AFPCreateSharedVolumesEnumerator` (page 17).

`code` A value of type `ATEventCode` specifying the event that triggered the callback. The value of `code` is `AT_SHAREDVOLUMES_COMPLETE`.

`result` A value of type `OSStatus`. A value of `noErr` indicates that the `AFPCreateSharedVolumesEnumber` successfully created a shared volume enumerator reference.

`cookie` An untyped pointer to arbitrary data. For details about `cookie`, see *Inside Macintosh: Networking with Open Transport*.

result Your notification callback routine should always return `noErr`.

DISCUSSION

Your notification callback routine should use the `userContext` parameter to determine which enumerator reference has been successfully created. Your application can then pass the enumerator reference to `AFPGetSharedVolumesCount` (page 22) to determine the number of volumes that the server identified by the enumerator reference shares.

Filter Callback Routine

Your filter callback routine is called to control the display of volume names. This is how you would declare your filter callback routine if you were to name it `MyFilterUPP`.

```
void MyFilterUPP(StringPtr name,
                 void *data);
```

`name` A value of type `StringPtr` that points to a volume name.

`data` An untyped pointer to arbitrary data that your application passed as the `filterParam` parameter when it called `AFPCreateSharedVolumesEnumerator` (page 17).

result Your filter callback routine should return `TRUE` to display the volume name and `FALSE` to prevent the volume name from being displayed.

DISCUSSION

Your filter callback routine should determine whether the volume identified by `name` should be displayed.

System Event Callback Routine

Your system event callback routine is called to handle update events that may occur while `AFPGetSharedVolumesCount` (page 22) displays the log on dialog box. Here is how you would declare your system event callback routine if you were to name it `MyAShareEventUPP`.

```
void MyAShareEventUPP(EventRecord *theEvent,
                 void *contextPtr);
```

`theEvent` A pointer to an event record that describes the event that occurred.

`event`	An untyped pointer to arbitrary data that your application passed as the `evtContext` parameter when it called `AFPCreateSharedVolumesEnumerator` (page 17). For more information on the `EventRecord` structure see *Inside Macintosh: Overview.*
result	Your system event callback routine should process the system event and return `noErr`.

DISCUSSION

Your system event callback routine may be called to handle events that occur while `AFPGetSharedVolumesCount` (page 22) displays the log on dialog box. Your system event callback routine should process the event and return `noErr`.

Result Codes

The result codes specific to the AppleShare Client API are listed here.

`kATEnumeratorBadIndexErr`	1	The specified index number is in valid.
`kATEnumeratorBadReferenceErr`	2	The specified enumerator reference is invalid.
`kATEnumeratorBadZoneErr`	3	The specified AppleTalk zone could not be found.
`kATEnumeratorBadPortErr`	4	The port number is not correct.
`kATAppleShareNotAvailableErr`	5	The server is not accepting connections.
`kATServerNotFoundErr`	6	The server could not be located.