AppleShare IP 6.3
Developer's Kit

AppleShare Client

Technical Publications

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

Simultaneously published in the United States and Canada.

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 text that you type or see displayed. This manual 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 sources provide additional information that may be of interest to AppleShare developers:

- *AppleShare IP Administrator's Manual.* Apple Computer, Inc.

- *Inside Macintosh*. Apple Computer, Inc.

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

- *AppleShare IP 6.3 Developer's Kit: AppleShare Registry Library.* Apple Computer, Inc.

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

- *AppleShare IP 6.3 Developer's Kit: AppleTalk Filing Protocol.* Apple Computer, Inc.
- *AppleShare IP 6.3 Developer's Kit: AppleTalk Filing Protocol Version 2.1 and 2.2*. Apple Computer, Inc.
- *Inside AppleTalk*, Second Edition. Apple Computer, Inc.

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

- *AppleShare IP 6.3 Developer's Kit: Server Control Calls and Server Event Handling*. Apple Computer, Inc.

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

- *AppleShare IP 6.3 Developer's Kit: AppleShare IP Print Server Security Protocol.* Apple Computer, Inc.

For information on using an AppleShare IP 6.3 file server 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 the AppleShare Client for versions of the Mac OS greater than Mac OS 8.6. The programming interface provides functions for creating and managing AppleTalk Filing Protocol (AFP) Universal Resource Locators (URLs) that can be used by applications such as the Network Browser. It also provides functions for mounting shared volumes.

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

The header file for the AppleShare Client programming interface is `afpClient.h`.

Verifying Library Information

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

- `AFPLibraryPresent` (page 1-7), which determines whether the AppleShare Client library is available.
- `AFPLibraryVersion` (page 1-8), which obtains the version of the AppleShare Client library.

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 9.0, the version number is 1.

DISCUSSION

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

Working with AFP URLs

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

- `NewAFPURL` (page 1-9), which creates an AFP URL.
- `DisposeAFPURL` (page 1-12), which disposes of an AFP URL.
- `IsAFPURL` (page 1-12), which determines whether a character string is a valid AFP URL.
- `ParseAFPURL` (page 1-10), which parses an AFP URL into its component parts.

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.

ParseAFPURL

Parses an AFP URL.

```
OSStatus ParseAFPURL (char * url,
                      StringPtr protocolName,
                      StringPtr serverNameOrHost
                      StringPtr zoneNameOrNULL,
                      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 1-9).

`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.

`zoneNameOrNull`
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, `zoneNameOrNull` 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 27).

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 1-9).
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.

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 1-9).

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.

Mounting Volumes

This section describes the functions that an application uses to mount AppleShare volumes. The functions described in this section have no dependency on any particular version of AppleShare.

The functions are:

- `AFPCreateSharedVolumesEnumerator` (page 1-14), which creates a shared volume enumerator reference containing the names of the volumes that a user has permission to mount.
- `AFPGetIndexedSharedVolume` (page 1-18), which obtains the name of a shared volume by its index number.
- `AFPSortSharedVolumes` (page 1-19), which sorts the list of volumes in a shared volume enumerator reference.
- `AFPMountSharedVolume` (page 1-20), which mounts a shared volume.
- `AFPGetLoginInformation` (page 1-21), 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 1-22), which obtains the startup mounting state of a shared volume.
- `AFPSetMountAtStartup` (page 1-22), which sets the startup mounting state of a shared volume.
- `AFPDeleteSharedVolumesEnumerator` (page 1-23), which disposes of a shared volume enumerator reference created by `AFPCreateSharedVolumesEnumerator`.

AFPCreateSharedVolumesEnumerator

Creates an enumerator reference for volumes on an AppleShare server.

```
OSStatus AFPCreateSharedVolumesEnumerator (
                    StringPtr serverName,
                    StringPtr serverZone,
                    StringPtr uamName,
                    StringPtr userName,
                    StringPtr password,
                    AShareEventUPP callback,
                    void *evtContext,
                    ATFilterProc filter,
                    void *filterParam,
                    ATNotifyProcPtr 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 17) 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 17), 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 17), 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 1-26) that handles events that occur while `AFPGetSharedVolumesCount` (page 1-17) 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 `ATFilterProc` that identifies an optional application-defined filter routine (page 1-25) 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 ATNotifyProcPtr that identifies an application-defined notification routine (page 1-24) 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 27).

DISCUSSION

The AFPCreateSharedVolumesEnumerator function creates a shared volume enumerator reference that can be passed as a parameter to AFPGetSharedVolumesCount, AFPGetIndexedSharedVolume, AFPSortSharedVolumes, and AFPMountSharedVolumes.

Passing the enumerator reference to AFPGetSharedVolumesCount (page 17) obtains the number of volumes that the user has permission to mount.

Passing the enumerator reference and an index number to AFPGetIndexedSharedVolume (page 18) obtains the name of the volume that is associated with the specified index number.

Passing the enumerator reference to AFPSortSharedVolumes (page 19) 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 AFPMountSharedVolumes (page 20) and the name of a volume causes the specified volume to be mounted.

IMPORTANT

Whenever `AFPCreateSharedVolumeEnumerator` (page 1-14) returns without error, it creates an enumerator reference that maintains a separate session with the AppleShare server in addition to sessions for any of that server's volumes. As soon as you know that the enumerator reference is no longer needed, you should dispose of the enumerator reference by calling `AFPDeleteSharedVolumesEnumerator` (page 1-21), closes this unnecessary session and releases system resources. ▲

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 1-14) 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 27).

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 1-18) 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 1-14) 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 1-17) 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 27).

DISCUSSION

The `AFPGetIndexedSharedVolume` function obtains the name of a volume by its index number. To determine the highest possible index number, call `AFPGetSharedVolumesCount` (page 1-18).

Once you obtain the name of a volume, you can sort the list of volume names by calling `AFPSortSharedVolumes` (page 19) and you can mount a particular volume by calling `AFPMountSharedVolumes` (page 20).

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 1-14).

function result A result code. For a list of possible result codes, see "Result Codes" (page 27).

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.

AFPMountSharedVolumes

Mounts a shared volume.

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

`ref` On input, a value of type `AFPSharedVolumesEnumeratorRef` created by previously calling `AFPCreateSharedVolumeEnumerator` (page 1-14).

`volumeName` On input, a value of type `Str255` that specifies the name of the volume that is to be mounted. Call `AFPGetIndexedSharedVolume` (page 18) 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 AppleTalk Filing Protocol commands to the server.

`isMounted` On input, a pointer to a Boolean. On output, `isMounted` is set to `TRUE` if the volume was already mounted prior to calling `AFPMountSharedVolumes`, or is set to `FALSE` if the volume was not already mounted.

function result A result code. For a list of possible result codes, see "Result Codes" (page 27).

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.

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 1-14) 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 27).

DISCUSSION

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

If a session with the server was active when the `ref` parameter was created, the `userName` and `password` parameters are empty when `AFPGetLoginInformation` returns. If the user selected a custom user authentication method (UAM) when using `ref` to log on, the `password` parameter is empty when `AGPGetLoginInformation` returns.

Note
You must call `AFPGetSharedVolumesCount` before calling `AFPGetLoginInformation`. ◆

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 1-14).

`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 1-14) 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 27).

DISCUSSION

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

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 1-14).
function result	A result code. For a list of possible result codes, see "Result Codes" (page 27).

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.

Whenever `AFPCreateSharedVolumeEnumerator` (page 1-14) returns without error, it creates an enumerator reference that maintains a separate session with the AppleShare server in addition to sessions for any of that server's volumes. Disposing of the enumerator reference closes this unnecessary session and releases system resources.

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`:

- 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` is resolved into an IP address.

This is how you would declare your notification callback routine if you were to name it `MyURLNotifyProc`:

```
OSStatus ATNotifyProcPtr (
                    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 14).

`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 17) 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 `MyFilterProc`.

```
void MyFilterProc(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 14).
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 17) 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 14). 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 17) 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.