AppleShare IP 6.3
Developer's Kit

AppleShare Registry Library

Technical Publications

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

Simultaneously published in the United States and Canada.

Contents

Listings and Tables

About This Manual

This document describes the programming interface for the AppleShare Registry. The AppleShare Registry is a database that stores information about AppleShare users, groups, and services, as well as the characteristics of the computer on which the Registry resides. The AppleShare Registry programming interface replaces the Users & Groups programming interface provided with AppleShare 3.0 and 4.0.

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 IP Administrator's Manual*. Apple Computer, Inc.
- *Inside Macintosh*. 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 user authentication modules (UAMs), see the following publication:

- *AppleShare IP 6.3 Developer's Kit*: User Authentication Modules. Apple Computer, Inc.

For information on the Print Server security protocol, see the following publication:

- *AppleShare IP 6.3 Developer's Kit*: AppleShare IP Print Server Security Protocol. 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 Handlng. 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 Registry Library

This document describes the programming interface for the AppleShare Registry. The AppleShare Registry is a database that stores information about AppleShare users, groups, and services, as well as the characteristics of the computer on which the Registry resides. The AppleShare Registry programming interface replaces the Users & Groups programming interface provided with AppleShare 3.0 and 4.0.

For this version of the AppleShare Registry, the Registry is synonymous with the Users & Groups Data File. Future versions of the Registry may provide access to data stored in other files. To prepare for future releases of AppleShare, applications that directly access the Users & Groups Data File should use the Registry instead.

Your application accesses the AppleShare Registry by calling functions in the AppleShare Registry Library. A server known as the Registry Agent responds to your application's Registry Library calls by obtaining or setting the requested information.

You can use the AppleShare Registry Library to access the Registry of the computer on which your application is running or the Registry of a remote computer. You can also extend the Registry with information that is specific to your application.

Note
AppleShare 3.0 and 4.0 and AppleShare IP 5.0 and 6.0 use the same format for the Users & Groups Data File. As a result, applications that directly access the Users & Groups Data File and are compatible with AppleShare 3.0 and 4.0 and AppleShare IP 5.0 and 6.0. However, future compatibility is not guaranteed. ◆

About the AppleShare Registry

Each computer that runs an AppleShare server has a unique Registry. That registry is independent of the Registry that may exist on other computers running an AppleShare server on the same network. The Registry stores data used by services running on the local computer and can be accessed from a remote computer. The Registry is not a distributed directory system, but it complements any distributed directory systems by providing storage for data that does not need to be shared globally.

Each computer that has a Registry runs the Registry Agent, which provides local and remote access to the Registry. The Registry Agent is a faceless background application that is launched at system startup. Applications cannot access the Registry until the agent is launched and has initialized itself. Extensions that require local access to the Registry must defer themselves until the agent is launched. Relying on the order in which the system launches extensions is not recommended.

In general, the AppleShare Registry supports synchronous BTree calls using the System 7.x BTree programming interface. It does not, however, support asynchronous BTree calls.

Before you make any calls that access data in the Registry, you should authenticate yourself to the agent by specifying a user that represents you and the user's password. To gain full access to information in the Registry, the user that you specify must be an administrator. Other users do not have any access to the Registry.

Note
Before you authenticate yourself to the Registry Agent, you can obtain status information about any service and the name and type of any object in the Registry. ◆

Objects

The Registry can contain an unlimited number of objects. An object can be one of the four object types listed in Table 1-1.

Table 1-1 Object types

Object	Purpose
`kMachine`	Contains information about the computer on which the Registry resides.
`kUser`	Contains information about a user of the services that run on this computer.
`kGroup`	Contains information about groups of users on this computer.
`kService`	Contains information about the services that are installed on this computer.

Each object in the Registry has a unique object ID that is assigned by the Registry when the object is created. The machine object is always the first object in the Registry, and its ID is always 0.

Attributes

Each object in the Registry consists of an unlimited number of attributes. An attribute stores an element of information. For example, a user object has an attribute for storing the user's password. Your application can add attributes to any object, including objects created by another application.

The value of an attribute is a sequence of raw bytes that does not have an explicit data type. The current BTree limits the size of an attribute value to approximately 200 bytes.

Every object has three predefined attributes; the predefined attributes are listed in Table 1-2. The value of these attributes is assigned when the object is created.

Table 1-2 Predefined attributes

Attribute	Description
`kShortID`	Contains a unique 32-bit value that identifies the object. The value of this attribute is assigned when the object is created. The value is never reused, even if the object is deleted.
`kName`	A name for the object. The name must be unique among all other objects in the Registry. For the machine, user, group, and service objects, this attribute stores the name of the computer, user, group, and service, respectively.
`kType`	The object's type, such as machine, user, group, or service.

Your application can locate objects by using the object's `kShortID` attribute or its `kName` attribute.

Each attribute has an associated signature that can be used, in combination with its name, to identify it. A signature is similar to a creator code in that your application can use the signature to identify the attributes that it creates. The combination of signature and attribute name guarantees that the attributes your application creates are unique.

The signature for the predefined attributes is `kBasic`.

The Machine Object

Each AppleShare Registry contains one machine object. The machine object stores information about the computer on which the Registry resides and provides access to the computer's Gestalt information. The machine object also contains information that was previously stored in the `ULInfo` record of the Users & Groups Data File.

Table 1-3 lists the attributes of the machine object.

Table 1-3 Machine object attributes

Attribute	Description
kGuestProgramLinking	Indicates whether program linking is available to users who log on as Guest.
kNoGuestAccess	Indicates whether users can log on to this computer as Guest.
kProgramLinking	Indicates whether program linking is enabled.
kFileSharingEnabled	Indicates whether file sharing is enabled on this computer.
kNoSavePassword	Indicates whether passwords can be saved in aliases.
kMultihoming	Indicates whether multihoming is enabled.
kUGFileVersion	Contains the version number of Users & Groups Data File.
kServerName	Contains the name of the file server.
kDefaultShutdown	Contains the default shutdown time (in minutes).
kMinPasswordLen	Contains the minimum password length.
kMaxBadLogins	Contains the maximum number of incorrect log-on attempts before a user's account is disabled.
kMaxPwdChgTime	Contains the maximum time that can elapse between password changes.
kUniqueID	Contains a random number that identifies this computer.
kGestalt	Contains Gestalt information about this computer.

The signature for a machine object attribute is `kMachine`.

User Objects

There is a user object for every user in the AppleShare Registry. Table 1-4 lists the attributes of the user object.

Table 1-4 User object attributes

Attribute	Description
`kPasswordAttribute`	Contains a password that can be up to 8 bytes in length.
`kPasswordLen`	Contains the length of the user's password.
`kUserFlags`	Contains user flags, as described in the enumeration that follows this table.
`kUserFailedPasswordAttempts`	Contains the number of failed log-on attempts since the last successful logon.
`kUserPasswordCreationTime`	Contains the time at which the current password was created.
`kUserNumGroups`	Contains the number of groups the user belongs to. In this release, a user can be belong to a maximum of 42 groups.
`kDisableDate`	Contains the date after which the user's account will be disabled.
`kLastLogin`	Contains the date of the user's last log on to any service.
`kUserCommment`	Provides space for storing arbitrary information about the user.
`kUserPhoneRecord`	Provides space storing the user's phone number and location.

The signature for a user object attribute is `kUser`.

Your application can use the following enumeration to interpret the value of the `kUserFlags` attribute:

```
enum {
    bmIACEnabled        = 0x0001,   /* If set, program linking is enabled. */
    bmLoginEnabled      = 0x0100,   /* If set, log on is enabled for this user. */
    bmSuperUser         = 0x0200,   /* If set, this user is an administrator. */
    bmDisableChangePwd  = 0x0400,   /* If set, this user cannot change password. */
    bmGetNewPwd         = 0x1000    /* If set, this user must change password. */
};
```

Group Objects

The Registry uses group objects to aggregate users. A group can have any number of members, but it cannot be a member of another group. Other than the predefined attributes, group objects do not have any attributes; however, your application can add attributes to group objects.

The signature for a group object attribute is `kGroup`.

Service Objects

The Registry has a service object for every server that is installed on the computer. The attributes of a service object contain information about the status of the server and its location on the computer. Each server is responsible for updating the information stored in its service object.

Table 1-5 lists the service object attributes.

Table 1-5 Service object attributes

Attribute	Description
`kShortStatus`	Contains a status word for the running server, as described in the enumeration that follows this table.
`kDetailedStatus`	Contains an application-defined status buffer for the running server.
`kServiceFlags`	Contains application-defined flags.

Table 1-5 Service object attributes (continued)

Attribute	Description
`kServicePSN`	Contains the process number of the running server, as set by the agent when the service starts up.
`kServiceAlias`	Contains an alias to the server's location.
`kServiceType`	Contains the server type, such as FTP, Print, Mail, File, or Web server.

Your application can use the following enumeration to define values for the `kShortStatus` attribute:

```
enum {
    kOAMServiceNotRunning   = 1,    /* Service is not running. */
    kOAMServiceRunning      = 2,    /* Service is running. */
    kOAMServiceStartingUp   = 3,    /* Service is starting up */
    kOAMServiceShutingDown  = 4     /* Service is shutting down */
};
```

The signature for a service object attribute is `kService`.

Using the AppleShare Registry Library

The AppleShare Registry Library uses objects and attributes to provide a wide range of functionality. You can use its routines to

- locate and establish a connection with a local or remote Registry Agent
- authenticate yourself to the Registry Agent
- manage objects in the Registry, including creating, enumerating, and deleting objects
- manage attributes, including listing attribute values and adding and deleting attributes
- manage groups, including checking memberships and adding, listing, and removing members

- receive notifications changes to an object
- manage services, including listing, starting, and stopping services, and getting status information from servers
- authenticate users in the Registry and set their passwords

Accessing the AppleShare Registry Library

The AppleShare Registry Library is a Code Fragment Manager library that allows you to connect to a local or remote Registry Agent. Connections with remote Registry Agents are made over AppleTalk.

The AppleShare Registry Library has built-in support for the Thread Manager, and it is recommended that you use it, especially if your application works with a Registry Agent that's running on a remote computer. To use the library's built-in Thread Manager support, your application should call `OAMBecomeServiceThread` to give at least one of your threads to the AppleShare Registry Library for use in processing network events. This function does not return until the AppleShare Registry Library closes, your application calls `OAMDeinitalize`, or your application exits.

AppleShare Registry Library functions are always synchronous relative to the thread on which they are made. That is, control does not return to the thread until the function completes. However, other threads continue to run while one thread is blocked making a call to the AppleShare Registry Library.

The Registry Agent can notify you when a change is made to an object in the Registry. These notifications are always sent at thread-safe time, usually in the context of the thread that called `OAMBecomeServiceThread`. You can call other Registry functions from a notification routine, but other notifications may be blocked until your notification routine returns.

Note
Applications that run at interrupt time cannot use the built-in support for the Thread Manager. Instead, these applications must provide their own thread support. ◆

Locating Registry Agents

All operations on a Registry take place over a connection with the Registry Agent that's running on the computer where that Registry resides. If you want

to connect to a remote Registry Agent, you must first locate the computer by calling `OAMFindServer`.

The `MyListRegistryAgents` routine shown in Listing 1-1 calls `OAMFindServer` to locate computers that are running agents. When called with wildcard values for `name` and `zone`, the routine looks for all Registry Agents running in all Appletalk zones.

Listing 1-1 Obtaining a list of available Registry Agents

```
void MyListRegistryAgents(unsigned char *  name, unsigned char * zone)
{
    int                 numInBuffer= 0;
    int                 numFound= 0;
    OAMStatus           err = noErr;
    char                buffer[2048];
    OAMBufferDescriptor lookup_bd;

    OAMServerLocator*loc = NULL;
    int loc_size = offsetof(OAMServerLocator,protSpecific) + sizeof(Str32);

    OAMServerSpec serverSpec;

    int serverIndex;

    /* set up the buffer descriptor */
    lookup_bd.buffer= buffer;
    lookup_bd.bufferLen= sizeof(buffer);
    lookup_bd.actCount= 0;

    /* build the AppleTalk OAMServerLocator */
    loc = (OAMServerLocator *) NewPtr(loc_size);
    memset(loc, 0, loc_size);

    loc->specSize = loc_size;
    loc->protType = kAppleTalk;
    memcpy (loc->name, name, sizeof(Str32));
    memcpy (loc->protSpecific, zone, sizeof(Str32));
```

```
err = OAMFindServer(loc, &lookup_bd, &numInBuffer, &numFound, NULL);
printf("Found %d AppleShare Registry Agent(s).\n", numFound);

for (serverIndex = 1; serverIndex<=numInBuffer; serverIndex++)
{
    err = OAMFindServerExtract(&lookup_bd, serverIndex, &serverSpec);
        p2cstr(serverSpec.name);
        printf("%s\n", serverSpec.name);
}
```

First, the `MyListRegistryAgents` routine builds the buffer descriptor structure in which `OAMFindServer` will return the search results. Then it builds an `OAMServerLocator` structure and copies into it the name and zone that are the target of the search.

After `MyListRegistryAgents` calls `OAMFindServer`, it calls `OAMFindServerExtract` to parse the buffer descriptor structure and print the name of each computer that is running the agent.

Connecting to a Registry Agent

To connect to a Registry Agent, you call `OAMOpenSession`. If you're connecting to the agent running on the local computer, you call `OAMOpenSession` with a null pointer as input. If you're connecting to the agent running on a remote computer, you call `OAMOpenSession` with a pointer to the `OAMServerSpec` structure that you obtained by calling `OAMFindServer`.

Listing 1-2 shows how to establish a connection to the local Registry Agent.

Listing 1-2 Connecting to a local Registry Agent

```
OAMSessionID MyOpenLocalAgentSession()
{
    OAMStatus       err = noErr;
    OAMSessionID    sessID = 0;

    err = OAMOpenSession(NULL, &sessID, NULL);

    return sessID;
}
```

Listing 1-3 shows how to connect to the remote Registry Agent.

Listing 1-3 Connecting to a remote Registry Agent

```
OAMSessionID MyOpenSession(Str31 serverName, Str31 zoneName)
{
    OAMStatus           err         = noErr;
    OAMSessionID        sessID      = 0;
    int                 numInBuffer = 0;
    int                 numFound    = 0;
    OAMServerLocator    *loc;
    OAMServerSpec       server;
    UInt32              loc_size= offsetof(OAMServerLocator,protSpecific) +
                        sizeof(Str32);

    /* set up the server locator */
    loc = (OAMServerLocator *)new char [loc_size];

    /* build the AppleTalk OAMServerLocator */
    memset(loc, 0, loc_size);
    loc->specSize = loc_size;
    loc->protType = kAppleTalk;
    memcpy (loc->name, serverName, sizeof(Str32));
    memcpy (loc->protSpecific, zoneName, sizeof(Str32));

    /* set up buffer descriptor */
    char                buffer[512]= {};
    OAMBufferDescriptor bd = { buffer, sizeof(buffer), 0};

    /* set up the server spec */
    memset(&server, 0, sizeof(OAMServerSpec));

    err = OAMFindServer(loc, &bd, &numInBuffer, &numFound, NULL);
    if(!err && numFound != 0) {
        err = OAMFindServerExtract(&bd, 1, &server);
        err = OAMOpenSession(&server, &sessID, NULL);
        printf("OAMOpenSession: %d\n", err);
        if (err == noErr) {
            printf("sessionID = %d\n", sessID);
        }
```

```
    } else
        printf("Server Not Found\n");
}
```

Authenticating the Session

Once you've connected to an agent, you can obtain status information about any service and the name and type of any object in that agent's Registry. To obtain any other information or to make changes to the Registry, you must authenticate the session.

To authenticate session, you call `OAMAuthenticateSession` and specify the user object that represents you and that user's password. The user that you specify must be an administrator.

The `MyAuthenticate` routine shown in Listing 1-4 provides sample authentication code.

Listing 1-4 Authenticating your identity for a session

```
OAMStatus MyAuthenticate(OAMSessionID sess, StringPtr name, StringPtr password)
{
    OAMStatuserr    = noErr;
    OAMObjectSpec   obj;
    OAMKey          key;

    memset(&key, 0, sizeof(key));
    memcpy(&key.keyBuffer, password, 8);
    key.keyBufferLen = 8;
    BuildObjectSpecByNameType(&obj, name, kUser);
    err = OAMAuthenticateSession(sess, &obj, &key, NULL);

    return err;
}

void BuildObjectSpecByNameType(OAMObjectSpec *obj, StringPtr name, OAMType type)
{
    memset(obj, 0, sizeof(OAMObjectSpec));
    obj->specType = kOAMObjectSpecByNameType;
    obj->objectType = type;
```

```
    short len = *name + 1;
    memcpy(obj->u.name, name, len);
}
```

First, the `MyAuthenticate` routine shown in Listing 1-4 builds an `OAMKey` structure and copies the password of the user in to it. Next, it calls `BuildObjectSpecifyNameType` to build an `OAMObjectSpec` structure containing the name of the user. Then `MyAuthenticate` calls `OAMAuthenticateSession` to authenticate the session.

The authentication will fail if the user you specify is not an administrator or if the user's ability to log on has been disabled.

Obtaining Information About Objects

Once you've connected to a Registry Agent and authenticated your session, you can access any object and any of the object's attributes.

The `MyGetMachineName` routine shown in Listing 1-5 gets the name of the computer on which the Registry Agent is running. The name of the computer is stored in the `ServerName` attribute of the machine object.

Listing 1-5 Getting the server name from the Registry

```
OAMStatus MyGetMachineName(OAMSessionID sess, StringPtr machineName)
{
    OAMStatus               err = noErr;
    OAMObjectSpec           machineObj;
    OAMAttributeDescriptor  myAttrDesc[2] = {};

    memset(&myAttrDesc, 0, sizeof(myAttrDesc));
    myAttrDesc[0].attributeSignature = kMachine;
    myAttrDesc[0].attributeType = kServerName;
    myAttrDesc[0].bufferDescriptor.buffer = machineName;
    myAttrDesc[0].bufferDescriptor.bufferLen = sizeof(Str31);
    myAttrDesc[0].bufferDescriptor.actCount = sizeof(Str31);
    /* terminate the array of attribute descriptors with a null */
    myAttrDesc[1].attributeSignature = NULL;
```

```
    BuildObjectSpecByShortID(&machineObj, kMachineShortID);
    err = OAMGetAttribute(sess, &machineObj,  myAttrDesc, NULL);

    return err;
}

void BuildObjectSpecByShortID(OAMObjectSpec *obj, OAMShortObjectSpec id)
{
    memset(obj, 0, sizeof(OAMObjectSpec));
    obj->specType = kOAMObjectSpecByShortID;
    obj->u.shortID = id;
}
```

First, the `MyGetMachineName` routine builds an array of two `OAMAttributeDescriptor` structures. The first `OAMAttributeDescriptor` structure is built so that it specifies an attribute signature of `kMachine` and an attribute type of `kServerName`. It also sets up the `bufferDescriptor` field of the `OAMAttributeDescriptor` structure so that it is long enough to hold the name that `OAMGetAttribute` will return. The second `OAMAttributeDescriptor` structure is set to null to terminate the array.

Next, the `MyGetMachineName` routine calls `BuildObjectSpecByShortID` to set the `specType` member of the `OAMObjectSpec` structure to `kOAMObjectSpecByShortID`, indicating that the object is to be found by its object ID. The `BuildObjectSpecByShortID` routine sets the value of `u.shortID` to `id`, whose value is 0.

Then the `MyGetMachineName` routine calls `OAMGetAttributes`. Because the value of `u.shortID` is 0, `OAMGetAttributes` returns the `kServerName` attribute (which contains the computer's name) for the machine object (which always has an object ID of 0).

Listing Objects in the Registry

To obtain the attributes for more than one object in the Registry, call the `OAMIterate` function. The `OAMIterate` function allows you to access objects without providing their names or IDs. You can iterate all objects in the Registry, the members of groups, or a user's group memberships. In addition to calling `OAMIterate`, you call `OAMIterParseNextObject` to extract information returned by `OAMIterate`.

The `MyListAllUserName` routine shown in Listing 1-6 calls `OAMIterate`, `OAMParseNextObject`, and `OAMParseGetNextAttribute` to list the names of all the user objects in the Registry and their attributes.

Listing 1-6 Listing the names of all user objects in the Registry

```
void MyListAllUserNames(OAMSessionID sess)
{
    OAMStatus               err = noErr;
    OAMIterationSpec        iter;
    UInt32                  maxToGet= 15;
    OAMType                 typeList[2] = {kUser,0};
    char                    bd_buffer[4096] = {};
    char                    iter_buffer[512] = {};
    OAMBufferDescriptor     bd  = {};
    OAMAttributeDescriptor  iter_attr[2] = {};
    Str31                   userName = {};
    int                     userCount = 0;

    /* set up the buffer descriptor */
    bd.buffer       = bd_buffer;
    bd.bufferLen    = sizeof(bd_buffer);
    bd.actCount     = 0;

    /* set up the iteration spec */
    memset(&iter, 0 , sizeof(iter));
    iter.iterationType = kOAMIterObjects;
    iter.typeList= typeList;
    iter.iterationFlags = kOAMIterByIndex;
    iter.u.startingIndex = 1;
    iter.maxToGet = maxToGet;

/* set up the iter_attr */
    iter_attr[0].attributeSignature = kBasic;
    iter_attr[0].attributeType = kName;
    iter_attr[0].bufferDescriptor.buffer = userName;
    iter_attr[0].bufferDescriptor.bufferLen = sizeof(iter_buffer);
    iter_attr[0].bufferDescriptor.actCount = 0;
```

```
/* null terminate the array of attribute descriptors */
    iter_attr[1].attributeSignature = NULL;

    do
    {
        err = OAMIterate(sess, &iter, iter_attr, &bd, NULL);

        if(!err) {
            OAMObjectSpec obj = {};
            OAMParseInfo parseInfo;
            int attrIndex;
            memset(&parseInfo, 0, sizeof(OAMParseInfo));
            err = OAMParseAttributeBuffer(&bd, iter_attr, &parseInfo);

            while(OAMParseGetNextObject(&parseInfo, &obj) != kOAMParseDone) {
                printf("id = %d type = %s\n", obj.u.shortID, (char*)&obj.objectType);
                for ( attrIndex = 0; ; attrIndex++)
                {
                    if (OAMParseGetNextAttribute(&parseInfo, &iter_attr[attrIndex]) !=
                        kOAMParseDone)
                        DisplayAttr(&iter_attr[attrIndex]);
                    else
                        break;
                    }
                    userCount++;
                }
                iter.u.startingIndex += maxToGet;
            }
        } while (!err && iter.more );
    printf("userCount(%d)\n",userCount);
}
```

The `MyListAllUserName` routine does the following:

1. Creates the iterator, specifying its type and a buffer in which to store the results.
2. Calls `OAMIterate`, which fills the provided buffer with information provided by the agent.
3. Calls `OAMIterParseNextObject` to extract information from the buffer.

4. Calls `OAMParseGetNextAttribute` to extract the value of each attribute and calls `DisplayAttr` (not shown) to display the value.
5. Continues calling `OAMParseGetNextAttribute` until there are no more attributes to display.
6. Continues to call `OAMIterParseNextObject` until it returns an error indicating that the buffer is empty.
7. Continues to call `OAMIterate` until it returns `OAMIterIsDone` to indicate that there are no more objects to iterate.

Creating Objects in the Registry

You use the `OAMCreateObject` function to create an object in the Registry. To create an object, you must supply a type and a unique name. You can also specify a list of attributes to be set for the new object.

The `MyCreateUser` routine shown in Listing 1-7 creates a new user and sets the user's password attribute.

Listing 1-7 Creating a new user

```
void MyCreateUser(OAMSessionID sessionID, unsigned char *name)
{
    OAMStatuserr            = noErr;
    OAMObjectSpec           userObj;
    OAMAttributeDescriptor  myAttrDesc[2] = {};
    char password[8]        = "temp\0\0\0\0";

    memset(&myAttrDesc, 0, sizeof(myAttrDesc));
    myAttrDesc[0].attributeSignature = kUser;
    myAttrDesc[0].attributeType = kPasswordAttribute;
    myAttrDesc[0].bufferDescriptor.buffer = password;
    myAttrDesc[0].bufferDescriptor.bufferLen = 8;
    myAttrDesc[0].bufferDescriptor.actCount = 0;
    /* terminate the array of attribute descriptors with a null *
    myAttrDesc[1].attributeSignature = NULL;
```

```
    printf("CreateUser(%#s)\n", name);
    BuildObjectSpecByNameType(&userObj,(StringPtr) name, kUser);
    err = OAMCreateObject(sessionID, &userObj, NULL, NULL);
}
```

Setting Attribute Values

To set an attribute value, call `OAMSetAttribute`. This function is very similar to the `OAMGetAttribute` function. The difference is that instead of copying the attribute value into a specified buffer, the attribute value is copied into the Registry from the specified buffer. You can specify a list of attributes to set in one call.

The `MyChangeUserName` routine shown in Listing 1-8 changes an object's name attribute. After this routine returns, the object can be found by specifying its new name.

Listing 1-8 Changing a user's name

```
void MyChangeUserName(OAMSessionID sess, OAMObjectSpec *userObj, unsigned char *name)
{
    OAMStatuserr = noErr;
    OAMAttributeDescriptor myAttrDesc[2] = {};

    memset(&myAttrDesc, 0, sizeof(myAttrDesc));

    myAttrDesc[0].attributeSignature = kBasic;
    myAttrDesc[0].attributeType = kName;
    myAttrDesc[0].bufferDescriptor.buffer = name;
    myAttrDesc[0].bufferDescriptor.bufferLen = sizeof(Str31);
    myAttrDesc[0].bufferDescriptor.actCount = 0;
    /* null terminate array of attribute descriptors */
    myAttrDesc[1].attributeSignature = NULL;

    err = OAMSetAttribute(sess, userObj,  myAttrDesc, NULL);
}
```

Receiving Notification of Changes in the Registry

You can request that the Registry Agent notify you when data in the Registry changes, thereby eliminating the need to poll constantly for changes.

To receive notifications from the Registry Agent, you must first set a notification procedure for your session. You can set only one notification procedure for a session.

Your notification procedure is called in the context of the thread that called `OAMBecomeSupportThread`. The notification procedure is called at the normal execution time for that thread. If you do not yield time to that thread, you will not receive notifications.

Your notification procedure is called with information about the event that occurred and information that identifies your session.

The `MySetNotifyProc` routine shown in Listing 1-9 installs the notification procedure named `MyNotifyProc` that prints out the notification information the receives when any user of the Registry Agent creates an object.

Listing 1-9 Requesting notification for the creation of objects

```
void MyNotifyProc( OAMNotification *no )
{
    printf("vers %d sess %X objectID %d notifyID %d userData %X type %X \n",
        no->version, no->sess, no->objectID, no->notifyID, no->userData, no->u.type);
}

void MySetNotifyProc(OAMSessionID sessionID)
{
    OAMStatus err = noErr;
    OAMNotifyOption notifyNewObj = { kOAMNotifyNewObject, 0 };
    OAMBooleanOption setSelf = { true };
    OAMNotificationSpec spec[3];

    err = OAMSetNotificationProc(sessionID, gNotifyProc, NULL);

    /* set up the NotifySpec */
    memset(&spec, 0, sizeof(spec));
    /* NOTIFY NEW OBJECT */
```

```
    spec[0].type = kOAMNotifyStartOption;
    spec[0].bufferDescriptor.buffer = &notifyNewObj;
    spec[0].bufferDescriptor.bufferLen = sizeof(OAMNotifyOption);

    /* NOTIFICATION (SETSELFSEND == TRUE) BY DEFAULT */

    spec[1].type = kOAMNotifySelfOption;
    spec[1].bufferDescriptor.buffer = &setSelf;
    spec[1].bufferDescriptor.bufferLen = sizeof(OAMBooleanOption);
    /* terminator */
    spec[2].type = 0;

    err = OAMRequestNotification(sessionID, spec, NULL);
}
```

First, the `MySetNotifyProc` routine calls `OAMSetNotificationProc` to install its notification procedure, `MyNotifyProc`.

Next, `MySetNotifyProc` builds an array of `OAMNotificationSpec` records. The first `OAMNotificationSpec` structure in the array specifies that the agent is to start notifying the application whenever new objects are created. The second `OAMNotificationSpec` structure specifies that the application wants to receive notifications of objects that it creates. The first member of the third `OAMNotificationSpec` structure is set to zero to null-terminate the array.

Then `MySetNotifyProc` calls `OAMRequestNotification` with the session ID and the array of `OAMNotificationSpec` structure arguments. When `MySetNotifyProc` returns, the application's `MyNotifyProc` routine will be called to receive notifications whenever any user of the agent creates a new object.

AppleShare Registry Constants and Data Types

The Attribute Descriptor Structure

You use an attribute descriptor structure to define the attributes for an object that is to be created, to get and set the values of attributes, to delete attributes, and to get the value of attributes that match an iteration criteria.

The `OAMAttributeDescriptor` data type defines an attribute descriptor structure.

```
struct OAMAttributeDescriptor {
    OAMType             attributeSignature; /* aka creator */
    OAMType             attributeType;
    OAMBufferDescriptor bufferDescriptor;
    OSStatus            status;
    UInt32              offset;
    UInt32              actSize;
};

typedef struct OAMAttributeDescriptor OAMAttributeDescriptor;
```

Field descriptions

`attributeSignature` The attribute's signature, such as `kBasic`.

`attributeType` The attribute's type, such as `kName`.

`bufferDescriptor` An `OAMBufferDescriptor` structure (page 1-33) used to store information about the object that is to be created, the attributes that are to be set or deleted, or the attributes that are to be obtained.

`status` On output, the result of the operation.

`offset` Reserved.

`actSize` On output, the number of bytes of data returned.

The Authenticate Information Structure

You use an authenticate information structure when you call `OAMAuthenticateObject` to authenticate an object, such as a user. The `OAMAuthenticateInfo` data type defines an authentication information structure.

```
struct OAMAuthenticateInfo {
    OAMObjectSpec*  objectSpec;
    UInt16          flags;
    UInt16          stage;  /* initially set to zero */
    OAMStatus       objStatus;
    UInt8           uam[64];
    UInt8           reserved[64];
};
```

```
typedef struct OAMAuthenticateInfo OAMAuthenticateInfo;
```

Field descriptions

`objectSpec`	A pointer to an `OAMObjectSpec` structure that identifies the object that is to be authenticated.
`flags`	An unsigned short that contains authentication flags.
`stage`	An unsigned short that identifies the authentication stage.
`objStatus`	An `OSStatus` that indicates the status of the authentication. For example, `objStatus` may indicate that the user's password has expired even though the user has been authenticated.
`uam`	An array of 64 unsigned bytes that specify the user authentication method.
`reserved`	Reserved.

The Buffer Descriptor Structure

You use a buffer descriptor structure to store information about objects and attributes that you are going to pass to the Registry Agent or receive from the Registry Agent. A buffer descriptor structure is a member of the `OAMAttributeDescriptor` structure and the `OAMNotificationSpec`.

You specify a buffer descriptor structure when you call the following AppleShare Registry Library functions:

- `OAMInitialize` to initialize the AppleShare Registry Library
- `OAMFindServer` to locate Registry Agents
- `OAMFindServerExtract` to extract Registry Agents
- `OAMIterate` to iterate objects in the Registry
- `OAMParseAttributeBuffer` to parse iterated objects
- `OAMAuthenticateObject` to authenticate a user
- `OAMChangeObjectKey` to change a user's password

The `OAMBufferDescriptor` data type defines a buffer description structure.

```
struct OAMBufferDescriptor {
    void *buffer;       /* the buffer in which data is to be stored */
    UInt32 bufferLen;   /* the length of the buffer */
    UInt32 actCount;    /* actual count */
};

typedef struct OAMBufferDescriptor OAMBufferDescriptor;
```

Field descriptions

`buffer`	A pointer to a buffer.
`bufferLen`	The length of the buffer pointed to by `buffer`.
`actCount`	The number of valid bytes in the buffer.

The Iteration Specification Structure

You use an iteration specification structure when you call `OAMIterate` to iterate objects in the AppleShare Registry. The `OAMIterationSpec` data type defines an iteration specification structure.

```
struct OAMIterationSpec {
    OAMType             iterationType;
    OAMFlags            iterationFlags;
    OAMType*            typeList;
    OAMObjectSpec       *iterArgument;
    UInt32              maxToGet;
    union {
        OAMObjectSpec   *startingObject;
        UInt32          startingIndex;
    } u;
    /* set by call */
    UInt32              totalSize;
    UInt32              lastPosition;
    Boolean             more;
};

typedef struct OAMIterationSpec OAMIterationSpec;
```

Field descriptions

`iterationType`	The iteration type. A value of 1 indicates an iteration of objects (`kOAMIterObjects`), a value of 2 indicates an iteration of group members (`kOAMIterMembers`), and a value of 3 indicates an iteration by objects that are members of groups (`kOAMIterMemberships`).
`iterationFlags`	The iteration flags. A value of 1 indicates an iteration by index (`kOAMIterByIndex`), and a value of 2 indicates an iteration by object (`kOAMIterByObject`). The value of `iterationFlags` determines the value of the union `u` that follows.
`typeList`	A pointer to an array whose elements specify the object types (`kMachine`, `kUser`, `kGroup`, and `kService`) that you want to iterate.
`iterArgument`	A pointer to an `OAMObjectSpec` structure that identifies the object at which to begin the iteration.
`maxToGet`	The maximum number of objects to return in one call to `OAMIterate`.
`startingObject`	A pointer to an `OAMObjectSpec` structure that identifies the object at which to begin the iteration.
`startingIndex`	If the value of `iterationFlags` is 1, the position at which to begin the iteration.
`totalSize`	The total size of the `OAMIterationSpec` structure.
`lastPosition`	On output, a value that indicates the last position iterated. To continue the iteration, set `startingObject` to the value of `lastPosition` returned by the previous call to `OAMIterate`.
`more`	On output, a value that indicates whether there are more objects to be iterated.

The Key Structure

You use a key structure to authenticate a session by calling `OAMAuthenticateSession`. The `OAMKey` data type defines a key structure.

```
struct OAMKey {
    OAMType keyType; /* 0->cleartext password */
    UInt32  keyBufferLen;
    UInt8   keyBuffer[8];
};

typedef struct OAMKey OAMKey;
```

Field descriptions

`keyType`	The key type. For this version, `keyType` must be 0 for clear text.
`keyBufferLen`	The length of `keyBuffer`.
`keyBuffer`	The buffer containing the null-padded password.

The Notification Specification Structure

You use a notification specification structure when you call `OAMRequestNotification` to specify the types of changes for which you want to receive notifications. The `OAMNotificationSpec` data type defines a notification specification structure.

```
struct OAMNotificationSpec {
    OAMType             type;
    OAMBufferDescriptor bufferDescriptor;
};

typedef struct OAMNotificationSpec OAMNotificationSpec;
```

Field descriptions

`type`	The type of notification. The value of `type` can be `kOAMNotifyStartOption` to start receiving notifications for the changes specified in `bufferDescriptor`, `kOAMNotifyStopOption` to stop receiving notifications for the changes listed in `bufferDescriptor`, or `kOAMNotifySelfOption` to stop or start receiving notifications for the changes listed in `bufferDescriptor` that are caused by your application.

`bufferDescriptor` An `OAMBufferDescriptor` structure that specifies the types of changes for which you want to receive notifications. The changes are a comma-separated, null-terminated list of one or more of the constants listed in Table 1-6.

Table 1-6 Changes that trigger notifications

`kOAMNotifyNewObject`	A new object is created.
`kOAMNotifyDeleteObject`	An object is deleted.
`kOAMNotifyRenameObject`	The name of an object changes.
`kOAMNotifyAttributeSet`	The value of an attribute changes.
`kOAMNotifyMemberAdd`	A member is added to a group.
`kOAMNotifyMemberRemove`	A member is removed from a group.
`kOAMNotifyLogin`	An object has been authenticated.
`kOAMNotifyAll`	Send notification when any of these changes occur.

The Notification Structure

You receive a notification structure when the agent calls your notification procedure. The `OAMNotification` data type defines a notification specification structure.

```
struct OAMNotification {
    UInt16              version; /* 0 */
    OAMSessionID        sess;
    OAMShortObjectSpec  objectID;
    UInt32              notifyID;
    UInt32              userData;
    union {
        OAMType type;
        OAMShortObjectSpec member;
    } u;
};

typedef struct OAMNotification OAMNotification;
```

Field descriptions

`version`	The version number. For this release of the AppleShare Registry Library, the version number is 0.
`sess`	The session ID for this notification.
`objectID`	The object ID of the object that changed.

`notifyID`	The change that occurred. The value of `notifyID` can be one of the following:
	`kOAMNotifyNewObject`
	`kOAMDeleteObject`
	`kOAMRenameObject`
	`kOAMNotifyAttributeSet`
	`kOAMMemberAdd`
	`kOAMMemberRemove`
	`kOAMNotifyLogin`

`userData`	Storage that can be used for any purpose.
`type`	The type of object that caused the notification to occur (`kMachine`, `kUser`, `kGroup`, or `kService`).
`member`	If `notifyID` is `kOAMMemberAdd` or `kOAMMemberRemove`, `member` contains the object ID of the affected member.

The Object Specification Structure

The object specification structure is a member of the `OAMIterationSpec` structure (page 1-34) and the `OAMAuthenticateInfo` structure (page 1-32).

You specify a buffer descriptor structure when you call the following AppleShare Registry Library functions:

- `OAMAuthenticateSession` to authenticate a session
- `OAMCreateObject` and `OAMDeleteObject` to create and delete objects in the AppleShare Registry
- `OAMGetAttribute`, `OAMSetAttribute`, and `OAMDeleteAttribute` to manage an object's attributes
- `OAMIsGroupMember`, `OAMAddGroupMember`, and `OAMRemoveGroupMember` to manage groups

- OAMParseNextObject to retrieve information about an object from a buffer
- OAMStartService and OAMStopService to start and stop a service

The OAMObjectSpec data type defines an object specification structure.

```
struct OAMObjectSpec {
    OAMType objectType;
    OAMType specType;
    union {
        OAMShortObjectSpec shortID;
        Str31 name;
        UInt8 otherTypes[4];
    } u;
};

typedef struct OAMObjectSpec OAMObjectSpec;
```

Field descriptions

objectType	The object's type (kMachine, kUser, kGroup, or kService.)
specType	The specification type. If spectype is 0, get object ID 0 (always the machine object). If spectype is 1, get the object by its ID number as specified in shortID. If spectype is 2, get the object by is name, as specified in name.
shortID	An unsigned, 32-bit value that identifies the object its ID number.
name	A 31-byte string value that identifies the object by its name.
otherTypes	Reserved.

The Parse Information Structure

You provide a parse information structure when you call OAMParseAttributeBuffer, OAMParseGetNextObject, and OAMParseGetNextAttribute. These functions use the parse information structure that you supply to maintain the current position in a buffer of object information obtained by calling OAMIterate.

The `OAMParseInfo` data type defines a parse information structure.

```
struct OAMParseInfo {
    UInt32 reserved[32];
};

typedef struct OAMParseInfo OAMParseInfo;
```

Field descriptions

`reserved`	Private information.

The Server Locator Structure

You use a server locator structure when you call `OAMFindServer` to locate Registry Agents. The OAM`ServerLocator` data type defines a server locator structure.

```
struct OAMServerLocator {
    UInt16 specSize;        /* total size of this structure */
    UInt32 protType;        /* 0 for AppleTalk */
    UInt32 reserved1;       /* reserved */
    UInt32 reserved2;       /* reserved */
    Str63 name;             /* The name of the server */
    UInt8 protSpecific[4];  /* Variable-size, protocol-specific part */
};

typedef struct OAMServerLocator OAMServerLocator;
```

Field descriptions

`specSize`	The total size of this `OAMServerLocator` structure.
`protType`	The protocol that will be used to locate the server. For this version of the AppleShare Registry Library, AppleTalk is the only supported protocol. To specify AppleTalk, use the constant `kAppleTalk`, which is defined as 0.
`reserved1`	Reserved.
`reserved2`	Reserved.
`name`	The name of the server that is to be located. The name can be a wildcard character.

protSpecific	Protocol-specific information required to locate an agent using the protocol specified by protType. For AppleTalk, the protocol-specific information is AppleTalk zone information.

The Server Specification Structure

You provide a server specification structure when you call OAMFindServerExtract to extract information for connecting to an agent from a buffer returned by the OAMFindServer function. The OAMFindServerExtract function returns the requested information in an OAMServerSpec structure which you can pass to OAMOpenSession in order to open a session with the server that the structure specifies.

The OAMServerSpec data type defines a server specification structure.

```
struct OAMServerSpec {
    UInt16 specSize;        /* total size of this server spec struct */
    UInt32 protType;        /* 0 for AppleTalk */
    UInt32 reserved1;       /* reserved */
    UInt32 reserved2;       /* reserved */
    Str63 name;             /* the name of the server */
    UInt8 protSpecific[4];  /* variable-size, protocol-specific part */
};

typedef struct OAMServerSpec OAMServerSpec;
```

Field descriptions

specSize	The total size of this server specification structure.
protType	The protocol that is to be used. For this version of the AppleShare Registry Library, the only supported protocol is AppleTalk. To specify AppleTalk, use the constant kAppleTalk, which is defined as 0.
reserved1	Reserved.
reserved2	Reserved.
name	The name of the server.

`protSpecific` Protocol specific information required to locate an agent using the protocol specified by `protType`. For AppleTalk, the protocol specific information is AppleTalk zone information.

AppleShare Registry Functions

Initializing the AppleShare Registry Library

OAMInitialize

Initializes the AppleShare Registry Library.

```
OAMStatus OAMInitialize (
                    UInt32 numSessions,
                    UInt32 maxNumCalls,
                    OAMBufferDescriptor* buffer_i,
                    OAMOption* options_i);
```

`numSessions` On input, the maximum number of sessions that your application will open.

`maxNumCalls` On input, the maximum number of outstanding requests that your application will make. For this version, the maximum number of outstanding requests is 1.

`buffer_i` Reserved. For this version, the value of `buffer_i` should be `NULL`.

`options_i` If your application runs at interrupt time, it should call `OAMGetSize` to obtain the appropriate memory size, allocate the memory, and pass a pointer to it when your application calls `OAMInitialize`. In all other cases, the value of `options_i` should be `NULL`.

function result A value indicating whether the AppleShare Registry Library is successfully initialized. If `OAMInitialize` does not return successfully, you should not call any other AppleShare Registry Library functions. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrServerNotFound`, and `kOAMErrMaximumSessions`.

DISCUSSION

The `OAMInitialize` function must be called at non-interrupt time before you call any other AppleShare Registry Library functions.

OAMDeInitialize

Deinitializes the AppleShare Registry Library.

```
OAMStatus OAMDeInitialize (void);
```

function result A value indicating whether the AppleShare Registry Library was successfully deinitialized.

DISCUSSION

The `OAMDeInitialize` function must be called at non-interrupt time to close any outstanding sessions and cancel any pending calls to AppleShare Registry Library functions. You cannot make any calls to the AppleShare Registry Library after you call `OAMDeInitalize`. After you call `OAMDeInitialize`, any calls that you have made to `OAMBecomeSupportThread` return.

OAMBecomeSupportThread

Gives a thread of execution to the AppleShare Registry Library.

```
OAMStatus OAMBecomeSupportThread (void);
```

DISCUSSION

The `OAMBecomeSupportThread` function gives a thread of your application's execution to the AppleShare Registry Library, which uses it to process requests that your application makes. Your application should call `OAMBecomeSupportThread` if it communicates with a remote agent and needs to process notifications in a timely, predictable manner.

If your application calls `OAMBecomeSupportThread`, it must make the call after its calls `OAMInitialize` and before it calls any other AppleShare Registry Library functions. The `OAMBecomeSupportThread` does not return until your application calls `OAMDeInitialize` or exits.

OAMGetSize

For applications that run at interrupt time, obtains the appropriate size of memory for a subsequent call to `OAMOpenSession`.

```
UInt32 OAMGetSize (
                    UInt32 numSessions,
                    UInt32 maxNumCalls);
```

`numSessions` On input, the maximum number of sessions that your application will open.

`maxNumCalls` On input, the maximum number of outstanding requests that your application will make. For this version, the maximum number of outstanding requests is 1.

function result A value representing the optimal buffer size in bytes for sessions that your application subsequently opens. Your application can pass the returned value to `OAMOpenSession`. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrServerNotFound`, and `kOAMErrMaximumSessions`.

DISCUSSION

The `OAMGetSize` function returns an integer value that represents the amount of memory that your application will need based on the value of `numSessions` and `maxNumCalls`. Only those applications that run at interrupt time need to call `OAMGetSize`. After calling `OAMGetSize`, your application should call `OAMInitialize` and pass to it the value returned by `OAMGetSize`.

Connecting to Agents

OAMGetLocalServerStatus

Obtains status information about an agent.

```
OAMStatus GetLocalServerStatus (UInt32 *serverStatus);
```

`serverStatus` On input, a pointer to an unsigned 32-bit value in which `OAMGetLocalServerStatus` is to return the status.

function result A value indicating whether `OAMGetLocalServerStatus` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrServerNotFound`, and `kOAMErrMaximumSessions`.

DISCUSSION

The `GetLocalServerStatus` function returns status information about an agent. On output, the possible values of `serverStatus` are

```
enum {
    kOAMServiceNotRunning   = 1,    /* service is not running. */
    kOAMServiceRunning      = 2,    /* service is running. */
    kOAMServiceStartingUp   = 3,    /* service is starting up */
    kOAMServiceShutingDown  = 4     /* service is shutting down */
};
```

OAMFindServer

Locates AppleShare Registry agents.

```
OAMStatus OAMFindServer (
                    OAMServerLocator *locator_i,
                    OAMBufferDescriptor *buffer_o,
                    UInt32 *numInBuffer_o,
                    UInt32 *numFound_o,
                    OAMOption* options_i);
```

`locator_i` On input, a pointer to an `OAMServerLocator` structure (page 40) that describes the agents that are being sought.

`buffer_o` On output, a pointer to an `OAMBufferDescriptor` structure (page 33) that contains information about the agents that were found.

`numInBuffer_o` On output, a pointer to an integer value containing the number of agents that are described by the buffer pointed to by `buffer_o`.

`numFound_o` On output, the total number of agents found. If `numFound_o` is greater than `numInBuffer_o`, `buffer_o` does not contain the complete list of available agents. You should increase the size of `buffer_o` and call `OAMFindServer` again.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMFindServer` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrServerNotFound`, and `kOAMErrMaximumSessions`.

DISCUSSION

The `OAMFindServer` function returns in `buffer_o` a list of agents. If you only want to receive the total number of available agents, call `OAMFindServer` with `buffer_o` set to null.

OAMFindServerExtract

Extracts information about a server and generates an `OAMServerSpec` structure.

```
OAMStatus OAMFindServerExtract (
                    OAMBufferDescriptor *buffer_i,
                    UInt32 index_i,
                    OAMServerSpec* spec_o);
```

`buffer_i` On input, a pointer to an `OAMBufferDescriptor` structure (page 33) containing a list of agents returned by previously calling `OAMFindServer`.

`index_i` On input, a value representing the agent information to be extracted from `buffer_i`. Starting at 1, increment `index_i` for each call to `OAMFindServerExtract`.

`spec_o` On output, a pointer to an `OAMServerSpec` structure (page 41) describing the server with which your application intends to open a session.

function result A value indicating whether `OAMFindServerExtract` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrServerNotFound`, and `kOAMErrMaximumSessions`.

DISCUSSION

The `OAMFindServerExtract` function extracts information about a server from a buffer returned by `OAMFindServer` and uses the information to generate an `OAMServerSpec` structure. Your application can then use the `OAMServerSpec` structure for a server to open a session with that server.

OAMOpenSession

Opens a session with an AppleShare Registry Agent.

```
OAMStatus OAMOpenSession (
                    OAMServerSpec* server_i,
                    OAMSessionID* sessID_o,
                    OAMOption* options_i);
```

`server_i` On input, a pointer to the `OAMServerSpec` structure (page 41) that describes the AppleShare Registry Agent with which your application wants to open a session. To create the `OAMServerSpec` structure, you application should call `OAMFindServer` and `OAMFindServerExtract`.

`sessID_o` On output, a pointer to the session ID for the opened session.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMOpenSession` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrServerNotFound`, and `kOAMErrMaximumSessions`. If your application is opening a session with the local agent, it should also check for these error conditions: `kOAMErrServerNotInstalled`, `kOAMErrServerNotReady`, `kOAMErrNoMachineName`.

DISCUSSION

The `OAMOpenSession` function opens a session with the server specified in the `OAMServerSpec` structure. You can open multiple sessions with the same agent, but doing so is not recommended, for performance reasons.

OAMAuthenticateSession

Authenticates an open session.

```
OAMStatus OAMAuthenticateSession (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* user,
                    OAMKey* key,
                    OAMOption *options_i);
```

`sessID_i` On input, the session ID that identifies the session that is to be authenticated.

`user` On input, a pointer to an `OAMObjectSpec` structure (page 38) that identifies the user to be authenticated for the opened session.

`key` On input, a pointer to an `OAMKey` structure (page 35) that contains the user's password.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMAuthenticateSession` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, `kOAMErrAuthenticationError`, `kOAMErrAuthenticationInProgress`, `kOAMErrLoginDisabled`, `kOAMErrAuthenticationServerError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMAuthenticateSession` function authenticates users. The specified user must have administrator privileges on the computer running the agent with which your application has opened a session.

OAMCloseSession

Closes a session.

```
OAMStatus OAMCloseSession (
                    OAMSessionID sessID_i,
                    OAMOption *options_i);
```

`sessID_i` On input, the session ID that identifies the session that is to be closed.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMCloseSession` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, and `kOAMErrSessionIDError`.

DISCUSSION

The `OAMCloseSession` function closes the specified open session.

Managing Objects

OAMCreateObject

Creates an object in the AppleShare Registry database.

```
OAMStatus OAMCreateObject (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* object,
                    OAMAttributeDescriptor *attr,
                    OAMOption *options_i);
```

`sessID_i` On input, the session ID that identifies the session for which the object is to be created.

`object`	On input, a pointer to the `OAMObjectSpec` structure (page 38) that contains information about the object that is to be created.
`attr`	On input, a pointer to an `OAMAttributeDescriptor` structure (page 31) that contains information about the attributes of the object that is to be created.
`options_i`	Reserved. For this version, `options_i` should be a null pointer.
function result	A value indicating whether `OAMCreateObject` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, and `kOAMErrDuplicateObject`.

DISCUSSION

The `OAMCreateObject` function creates an object in the AppleShare Registry database on the computer that is running the agent for the specified session.

OAMDeleteObject

Deletes an object from the AppleShare Registry database.

```
OAMStatus OAMDeleteObject (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* object,
                    OAMOption* options_i);
```

`sessID_i`	On input, the ID of the session for which an object is to be deleted.
`object`	On input, a pointer to the `OAMObjectSpec` structure (page 39) that identifies the object that is to be deleted.
`options_i`	Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMDeleteObject` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, and `kOAMErrObjectNotFound`.

DISCUSSION

The `OAMDeleteObject` function deletes the specified object from the AppleShare Registry database of the computer that is running the agent for the specified session.

OAMIterate

Lists objects in the AppleShare Registry that match an iteration criteria.

```
OAMStatus OAMIterate (
                    OAMSessionID sessID_i,
                    OAMIterationSpec* iterSpec,
                    OAMAttributeDescriptor *desc,
                    OAMBufferDescriptor *buffer,
                    OAMOption *options_i);
```

`sessID_i` On input, the ID of the session for which the iteration is to be performed.

`iterSpec` On input, a pointer to the `OAMIterationSpec` structure (page 34) that specifies the types of objects that are to be iterated and the iteration method (by object ID or by index).

`desc` On input, a pointer to an `OAMAttributeDescriptor` structure (page 31) that describes the object attributes that comprise the iteration criteria.

`buffer` On input, a pointer to an `OAMBufferDescriptor` structure (page 33) into which `OAMIterate` is to place the output of the iteration. On output, `buffer_o` contains the iteration output.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMIterate` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMIterate` function obtains a buffer that contains objects that match the iteration criteria specified by `iterSpec`. Once you've obtained the buffer, pass it to `OAMParseAttributeBuffer`, which prepares the buffer for parsing. Then call `OAMParseGetNextObject` to extract the information about each object from the buffer.

OAMParseAttributeBuffer

Prepares for parsing a buffer that contains iteration output.

```
OAMStatus OAMParseAttributeBuffer (
                    OAMBufferDescriptor *buffer_i,
                    OAMAttributeDescriptor *desc,
                    OAMParseInfo* parseInfo_o);
```

`buffer_i` On input, a pointer to an `OAMBufferDescriptor` structure (page 33) returned by `OAMIterate`.

`desc` On input, a pointer to an `OAMAttributeDescriptor` structure (page 31) that stores attribute values.

`parseInfo_o` On output, a pointer to a `parseInfo` structure (page 1-39) that contains information that is required by the `OAMParseGetNextObject` function.

function result A value indicating whether `OAMParseAttributeBuffer` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMParseAttributeBuffer` prepares a buffer returned by `OAMIterate` for parsing. Once `OAMParseAttributeBuffer` prepares the buffer, you can call `OAMParseGetNextObject` to extract the information about each object from the prepared buffer.

OAMParseGetNextObject

Retrieves information about an object from a buffer that has been prepared for parsing by previously calling `OAMParseAttributeBuffer`.

```
OAMStatus OAMParseGetNextObject (
                    OAMParseInfo* parseInfo_o,
                    OAMObjectSpec* object);
```

`parseInfo_o` On output, a pointer to the `OAMParseInfo` structure (page 1-39) that contains state information about the current position in the buffer for subsequent calls to `OAMParseGetNextObject` and `OAMParseGetNextAttribute`.

`object` On input, a pointer to the `OAMObjectSpec` structure (page 38) in which `OAMParseGetNextObject` is to place information about the next object in the buffer. On output, `object` contains the next object in the buffer prepared for parsing by `OAMParseAttributeBuffer`.

function result A value indicating whether `OAMParseGetNextObject` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMParseGetNextObject` function retrieves information about objects in the AppleShare Registry from a buffer that has been prepared for parsing by previously calling `OAMParseAttributeBuffer`.

OAMParseGetNextAttribute

Retrieves information about the next attribute for an object from a buffer that has been prepared for parsing by `OAMParseAttributeBuffer`.

```
OAMStatus OAMParseGetNextAttribute (
                    OAMParseInfo* parseInfo_o,
                    OAMAttributeDescriptor* attr);
```

`parseInfo_o` On output, a pointer to the `OAMParseInfo` structure (page 1-39) that contains state information about the current position in the buffer for subsequent calls to `OAMParseGetNextObject` and `OAMParseGetNextAttribute`.

`attr` On input, a pointer to the `OAMAttributeDescriptor` structure (page 31) in which `OAMParseGetNextAttribute` is to place information about the next attribute. On output, `attr` contains the requested attribute information.

function result A value indicating whether `OAMParseGetNextAttribute` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, and `kOAMErrUAMNotFound`.

DISCUSSION

Retrieves information about the next attribute from a buffer that has been prepared by previously calling `OAMParseAttributeBuffer`.

Managing Attributes

OAMGetAttribute

Gets the value of an object's attributes.

```
OAMStatus OAMGetAttribute (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* object,
                    OAMAttributeDescriptor *attr,
                    OAMOption* options_i,
```

`sessID_i` On input, the ID of the session for which an object's attributes are to be obtained.

`object` On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the object whose attributes are to be obtained.

`attr` On input, a pointer to null-terminated array of `OAMAttributeDescriptor` structures (page 31). On output, the array contains the value of the specified attributes for the specified object.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMGetAttribute` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, `kOAMErrObjectNotFound`, and `kOAMErrRequestTooLarge`. The `status` member of `attr` may contain one of the following errors: `kOAMErrMaximumAttributes`, `kOAMErrAttributeNotFound`, or `kOAMErrAttributeBufferTooSmall`.

DISCUSSION

The `OAMGetAttribute` function obtains the value of the specified attributes for the specified object from the AppleShare Registry of the computer that is running the agent for the specified session. Your application should request the value of no more than 20 attributes.

OAMSetAttribute

Sets the value of an object's attributes.

```
OAMStatus OAMSetAttribute (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* object,
                    OAMAttributeDescriptor *attr,
                    OAMOption* options_i);
```

`sessID_i` On input, the ID of the session for which an object's attributes are to be set.

`object` On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the object whose attributes are to be set.

`attr` On input, a pointer to a null-terminated array of `OAMAttributeDescriptor` structures (page 31) that contains the attribute values that are to be set.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMSetAttribute` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrObjectNotFound`, and `kOAMErrSessionIDError`. The `status` member of `attr` may contain one of the following errors: `kOAMErrMaximumAttributes`, `kOAMErrAttributeNotFound`, `kOAMAttributeReadOnly`, or `kOAMErrAttributeBufferTooLarge`.

DISCUSSION

The `OAMSetAttribute` function sets the value of the specified attributes for the specified object in the AppleShare Registry of the computer running the agent for the specified session. Your application should try to set no more than 20 attributes in a single call.

OAMDeleteAttribute

Deletes the specified attributes from the specified object.

```
OAMStatus OAMDeleteAttribute (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* object,
                    OAMAttributeDescriptor attr,
                    OAMOption* options_i);
```

`sessID_i` On input, the ID of the session for which an object's attributes are to be deleted.

`object` On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the object whose attributes are to be deleted.

`attr` On input, a pointer to a null-terminated array of `OAMAttributeDescriptor` structures that identifies the attributes that are to be deleted.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMDeleteAttribute` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrObjectNotFound`, and `kOAMErrSessionIDError`. The `status` member of `attr` may contain one of the following errors: `kOAMErrMaximumAttributes`, `kOAMErrAttributeNotFound`, `kOAMErrAttributeReadOnly`, or `kOAMErrAttributeReadWriteOnly`.

DISCUSSION

The `OAMDeleteAttribute` function deletes the specified attributes from the specified object in the AppleShare Registry of the computer running the agent for the specified session. Your application should try to delete no more than 20 attributes in a single call.

Managing Group Membership

OAMIsGroupMember

Determines whether a user object is a member of a group object.

```
OAMStatus OAMIsGroupMember (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* group,
                    OAMObjectSpec* member,
                    Boolean *isMember,
                    OAMOption *options_i);
```

`sessID_i`	On input, the ID of the session for which a user's group membership is to be confirmed.
`group`	On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the group for which a user's membership is to be confirmed.
`member`	On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the user whose group membership is to be confirmed.
`isMember`	On output, a pointer to a Boolean value indicating whether the user is a member of the group.
`options_i`	Reserved. For this version, `options_i` should be a null pointer.
function result	A value indicating whether `OAMIsGroupMember` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, `kOAMErrContainerObjectNotFound`, and `kOAMErrMemberObjectNotFound`.

DISCUSSION

The `OAMIsGroupMember` obtains information about whether the specified user is a member of the specified group.

OAMAddGroupMember

Adds a member to a group.

```
OAMStatus OAMAddGroupMember (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* group,
                    OAMObjectSpec* newmember,
                    OAMOption *options_i);
```

`sessID_i` On input, the ID of the session for which a membership is to be added.

`group` On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the group to which a membership is to be added.

`newMember` On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the member that is to be added.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMAddGroupMember` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, `kOAMErrContainerObjectNotFound`, `kOAMErrMemberObjectNotFound`, and `kOAMErrMaximumMemberObjects`.

DISCUSSION

The `OAMAddGroupMember` function adds the specified member to the specified group. You cannot add a group to a group. For this release, a user can be a member of no more than 42 groups.

OAMRemoveGroupMember

Removes a member from a group.

```
OAMStatus OAMRemoveGroupMember (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* group,
                    OAMObjectSpec* oldMember,
                    OAMOption *options_i);
```

`sessID_i`	On input, the ID of the session for which a membership is to be removed.
`group`	On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the group from which a membership is to be removed.
`oldMember`	On input, a pointer to the `OAMObjectSpec` structure (page 38) that identifies the member that is to be removed.
`options_i`	Reserved. For this version, `options_i` should be a null pointer.
function result	A value indicating whether `OAMRemoveGroupMember` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrSessionIDError`, `kOAMErrContainerObjectNotFound`, and `kOAMErrMemberObjectNotFound`.

DISCUSSION

The `OAMRemoveGroupMember` function removes the specified member from the specified group.

Receiving Notifications

OAMSetNotificationProc

Establishes the notification procedure for the current session.

```
OAMStatus OAMSetNotificationProc (
                    OAMSessionID sessID_i,
                    OAMNotificationUPP procPtr,
                    OAMOption *options_i);
```

`sessID_i` On input, the ID of the session for which a notification procedure is to be established.

`procPtr` On input, a pointer to your application's notification procedure. For an example, see "Receiving Notification of Changes in the Registry" (page 30).

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMSetNotificationProc` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMSetNotificationProc` function establishes the notification procedure for the current session. Later, your application can call `OAMSetNotificationProc` again to establish a different notification procedure (but you do not have to call `OAMRequestNotification` again). Only one notification procedure can be installed at any one time.

OAMRequestNotification

Registers notification types.

```
OAMStatus OAMRequestNotification (
                    OAMSessionID sessID_i,
                    OAMNotificationSpec *notificationSpec,
                    OAMOption *options_i);
```

`sessID_i` On input, the ID of the session for which a notification procedure is to be established.

`notificationSpec`
On input, a pointer to a `OAMNotificationSpec` structure (page 36) that specifies the notification types for which the application wants to receive notifications.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMRequestNotification` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMRequestNotification` function registers the notification types that you want to receive. Later, you can call `OAMRequestNotification` again to change the types of notifications, and you can call `OAMRequestNotification` to stop receiving notifications. By default, `OAMRequestNotification` sends notifications for changes that you make to objects in the AppleShare Registry.

Before you call `OAMRequestNotification`, you should call `OAMSetNotificationProc` to install the notification procedure that you use to receive notifications.

Managing Services

OAMStartService

Starts the specified server.

```
OAMStatus OAMStartService (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* service,
                    OAMOption* options_i);
```

`sessID_i` On input, the ID of the session for which a server is to be started.

`service` On input, pointer to an `OAMObjectSpec` structure (page 38) that identifies the server that is to be started.

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMStartService` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMStartService` function starts the server specified by `service`.

OAMStopService

Stops the specified server.

```
OAMStatus OAMStopService (
                    OAMSessionID sessID_i,
                    OAMObjectSpec* service,
                    OAMOption* options_i);
```

`sessID_i`	On input, the ID of the session for which a service is to be started.
`service`	On input, pointer to an `OAMObjectSpec` structure (page 38) that identifies the server that is to be stopped.
`options_i`	Reserved. For this version, `options_i` should be a null pointer.
function result	A value indicating whether `OAMStopService` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMStopService` function stops the server specified by `service`.

Authenticating Objects

OAMAuthenticateObject

Authenticates the specified object.

```
OAMStatus OAMAuthenticateObject (
                    OAMSessionID sessID_i,
                    OAMAuthenticateInfo* authInfo,
                    OAMBufferDescriptor* input,
                    OAMBufferDescriptor* output,
                    OAMOption* options_i);
```

`sessID_i` On input, the ID of the session for which a service is to be started.

`authInfo` On input, a pointer to an `OAMAuthenticateInfo` structure (page 1-32) that specifies the user who is to be authenticated.

`input` On input, a pointer to an `OAMBufferDescriptor` structure (page 33) containing a password.

`output` On output, pointer to an `OAMBufferDescriptor` structure (page 33) containing the output of the user authentication method (if any).

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMAuthenticateObject` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMAuthenticateObject` function authenticates the specified object.

OAMChangeObjectKey

Changes a user's password.

```
OAMStatus OAMChangeObjectKey (
                    OAMSessionID sessID_i,
                    OAMAuthenticateInfo* authInfo,
                    OAMBufferDescriptor* input,
                    OAMBufferDescriptor* output,
                    OAMOption* options_i);
```

`sessID_i` On input, the ID of the session for which a user's password is to be changed.

`authInfo` On input, a pointer to an `OAMAuthenticateInfo` structure (page 1-32) that specifies the user whose password is to be changed.

`input` On input, a pointer to an `OAMBufferDescriptor` structure (page 33) containing a password.

`output` On output, pointer to an `OAMBufferDescriptor` structure (page 33) containing the output of the user authentication method (if any).

`options_i` Reserved. For this version, `options_i` should be a null pointer.

function result A value indicating whether `OAMChangeObjectKey` returned successfully. Your application should check for these error conditions: `kOAMErrInitializationError`, `kOAMErrParameterError`, `kOAMErrNetworkError`, `kOAMErrAuthenticationError`, `kOAMErrLoginDisabled`, `kOAMErrSessionIDError`, and `kOAMErrUAMNotFound`.

DISCUSSION

The `OAMChangeObjectKey` function changes the password for the specified user.

AppleShare Registry Library Result Codes

The result codes specific to the AppleShare Registry Library are listed here.

noErr	0	No error
kOAMErrInitializationError	–29300	The AppleShare Registry Library has not been initialized.
kOAMErrParameterError	–29301	A parameter is invalid.
kOAMErrGeneralError	–29302	An internal error occurred.
kOAMErrObjectNotFound	–29310	The specified object or object type does not exist in the Registry.
kOAMErrContainerObjectNotFound	–29311	The specified group object does not exist in the Registry.
kOAMErrMemberObjectNotFound	–29312	The specified group member does not exist in the Registry.
kOAMErrDuplicateObject	–29320	The specified object already exists in the Registry.
kOAMErrMaximumObjects	–29330	The user object already contains the maximum number of group members.
kOAMErrMaximumMemberObjects	–29331	The group object already has the maximum number of members.
kOAMErrAttributeNotFound	–29340	The specified attribute does not exist in the Registry.
kOAMErrAttributeReadOnly	–29341	The specified attribute allows only read access. Its value is maintained by the Registry.
kOAMErrAttributeReadWriteOnly	–29342	The specified attribute is a required attribute that cannot be deleted.
kOAMErrAttributeBufferTooSmall	–29343	The specified buffer is too small to store the data that has been returned by an AppleShare Registry Library function.
kOAMErrAttributeBufferTooLarge	–29344	The specified buffer is too large to store the data that has been passed to an AppleShare Registry Library function.

kOAMErrMaximumAttributes	–29345	More than 20 attributes have been specified.
kOAMErrBufferTooSmall	–29350	The specified buffer is too small to store the data that has been returned by an AppleShare Registry Library function.
kOAMErrBufferTooLarge	–29351	The specified buffer is too large to store the data that has been passed to an AppleShare Registry Library function.
kOAMErrAuthenticationError	–29360	An authentication error. For example, the specified password is incorrect or the user is not an administrator.
kOAMErrAuthenticationInProgress	–29361	The call to OAMAuthenticate was successful, but additional calls to OAMAuthenticate must be made to complete the authentication process.
kOAMErrLoginDisabled	–29362	Log-on privileges for the user that was used to authenticate this session have been disabled.
kOAMErrAuthenticationServerError	–29363	The server failed a key challenge from the client.
kOAMErrUAMNotFound	–29364	The requested user authentication module does not exist.
kOAMErrAdminDisabled	–29365	Administrative privileges for the user object used to authenticate this session have been disabled.
kOAMErrAuthenticationAdminErr	–29366	Authentication error.
kOAMErrPasswordNeedsChange	–29370	Authentication was successful, but the password of the user object used to authenticate this session must be changed before it can be used again.
kOAMErrPasswordExpired	–29371	Authentication failed. The user's password has expired.
kOAMErrPasswordMinimumLen	–29372	Authentication succeeded, but the password is shorter than the minimum allowed.

kOAMErrSamePassword	–29373	The password specified in a call to OAMChangeObjectKey is the same as the current password.
kOAMErrPasswordChangeDisabled	–29374	The user object specified in a call to OAMChangeObjectKey is not allowed to change the password attribute.
kOAMErrServerNotFound	–29380	The specified agent was not found on the network.
kOAMErrServerNotInstalled	–29381	The AppleShare Registry Agent is not installed on this machine.
kOAMErrServerNotReady	–29382	The agent is starting up. Reissue the call after a short delay.
kOAMErrNoMachineName	–29383	The machine name is not available to the local agent.
kOAMErrRequestTooLarge	–29384	The call returned more than the maximum amount of allowable data. Adjust parameters to return less data.
kOAMErrNetworkError	–29385	The connection to the Registry has been lost because of a network failure or the termination of an agent. Establish another session when the network is restored and the agent is available.
kOAMErrSessionIDError	–29386	The session ID is invalid.
kOAMErrMaximumSessions	–29387	Your application tried to open more sessions than it specified when it called OAMInitialize.

Appendixes

Interface Files

This appendix describes the constants and data types defined for AppleShare Registry, Web & File server and Mail server attributes.

AppleShare Registry Attributes

This section describes the constants and data types defined in AppleShareRegistry.h for the Apple Registry object types and attributes.

Object Types

The following object types are defined for the AppleShare Registry:

```
enum {
    kOAMAnyObjectType   = '****',
    kOAMMachine         = 'mach',
    kOAMUser            = 'user',
    kOAMGroup           = 'grop',
    kOAMService         = 'serv'
};
```

Object Attributes

All objects have the following attributes:

```
enum {
    kOAMShortID         = 'shid', /* UInt32 */
    kOAMName            = 'name', /* Str31 */
    kOAMInternetName    = 'inam', /* Str31 */
    kOAMType            = 'type'  /* OSType */
};
```

Machine Object Attributes

In addition to the attributes defined in “Object Attributes” on page A-73, machine objects have the following attributes:

```
enum {
    kOAMGuestProgramLinking = 'gstl', /* UInt8 */
    kOAMNoGuestAccess       = 'gsta', /* SInt16 */
    kOAMProgramLinking      = 'prgl', /* SInt16 */
    kOAMBoot                = 'boot', /* SInt16 */
    kOAMFileSharingEnabled  = 'fshr', /* SInt16 */
    kOAMApple2Mode          = 'apl2', /* SInt16 */
    kOAMNoSavePassword      = 'spwd', /* SInt16 */
    kOAMMultihoming         = 'mlth', /* SInt16 */
    kOAMGuestInited         = 'gsti', /* SInt16 */
    kOAMUGInited            = 'ugin', /* SInt16 */
    kOAMUGFileVersion       = 'vers', /* SInt16 */
    kOAMServerName          = 'name', /* Str32 */
    kOAMDefaultShutdown     = 'smin', /* SInt16 */
    kOAMMinPasswordLen      = 'plen', /* UInt8 */
    kOAMMaxBadLogins        = 'bmax', /* SInt16 */
    kOAMMaxPwdChgTime       = 'pwdc', /* SInt16 */
    kOAMLoginDisabledTime   = 'disT', /* SInt32 */
    kOAMUniqueID            = 'uniq', /* SInt32 */
    kOAMMoreSFlags          = 'msfl', /* long */
    kOAMOtherFlags          = 'oflg', /* long */
    kOAMOTPort              = 'port', /* SInt16 */
};
```

User Object Attributes

In addition to the attributes defined in “Object Attributes” on page A-73, user objects have the following attributes:

```
enum {
    kOAMPasswordAttribute          = 'pwd '),  /* must be 8 bytes (zero
                                                  padded) */
    kOAMPasswordLen                = 'pwdL',   /* UInt8 */
    kOAMUserFlags                  = 'flgs',   /* UInt16 */
    kOAMUserFailedPasswordAttempts = 'blog',   /* UInt16 */
    kOAMUserPasswordCreationTime   = 'pwdD',   /* UInt32, set by
```

```
                                                    Agent) */
    kOAMUserNumGroups               = 'nmem',   /* UInt16 (read only) */
    kOAMPrimaryGroup                = 'pgrp',   /* UGRec (id and name) */
    kOAMDisableDate                 = 'disD',   /* SInt32 */
    kOAMLastLogin                   = 'logD',   /* SInt32 */
    kOAMUserComment                 = 'cmnt',   /* Str197 */
    kOAMUserPhoneRecord             = '976 '    /* OAMUserPhoneRecord */
};
```

Group Object Attributes

The attributes defined for group objects are object attributes as described in “Object Attributes” on page A-73.

Service Object Attributes

In addition to the attributes defined in “Object Attributes” on page A-73, service objects have the following attributes:

```
enum {
    kOAMShortStatus     = 'shst', /* UInt32 */;
    kOAMServiceFlags    = 'flag', /* OAMFlags */;
    kOAMServicePSN      = 'psn ', /* ProcessSerialNumber */
    kOAMServiceFSSpec   = 'fssp', /* FSSpec */
    kOAMServiceType     = 'styp'  /* OSType */
};
```

Web & File Server Service Object Attributes

This section describes the constants and data types defined in AppleShareFileServerRegistry.h for the Web & File server service object attributes.

```
enum {
    kFSCreatorSig   =   'ipwf'
};
```

```
enum {
    kFSServerInfoType       = 'srvr', /* ServerInfo structure*/
    kFSServerGreetingType   = 'gret', /* ServerGreeting structure */
    kFSCacheInfoType        = 'cach', /* CacheInfo structure */
    kFSIdleUserInfoType     = 'idle', /* UserInfo structure */
    kFSAdminInfoType        = 'admm', /* AdminInfo structure */
    kFSHTTPInfoType         = 'http', /* HTTPServiceInfo structure */
    kFSHTTPFolderType       = 'hfol', /* HTTPFolderRec structure */
    kFSHTTPFileType         = 'hfil', /* HTTPFileRec structure */
    kFSHTTPPluginType       = 'hplg', /* HTTPPluginsRec structure */
    kFSMultiWebDirSig       = 'webd', /* Multi-domain signature */
    kFSHTTPWebDirCountType  = 'domC', /* HTTPWebDirCountRec structure */
    kFSHTTPWebDirType       = 'OOOO', /* HTTPWebDirREc structure */
    kFSFTPInfoType          = 'ftp ', /* FTPServiceInfo structure */
    kFSAFPInfoType          = 'afp ', /* AFPServiceInfo structure */
    kFSSMBInfoType          = 'smb ', /* SMBServiceInfo structure */
    kFSMimeType             = 'mime', /* MimeTypeChanged structure */
    kFSIPFilterType         = 'filt', /* IPFilterInfo structure */
    kFSMaxConnection        = 'maxc'  /* ServerMaxConnection structure */
};

/* Service specific advanced options */
enum {
    kFSAFPTCPSig    = 'afpt', /* AFP over TCP port signature */
    kFSSMBInfoSig   = 'smb ', /* SMB over TCP port signature */
    kFSHTTPInfoSig  = 'http', /* HTTP port signature */
    kFSFTPInfoSig   = 'ftp '  /* FTP port signature */
};

enum {
    kFSTransferTimeout  = 'trto', /* transfer timeout for kFSFTPInfoSig (SInt32) */
    kFSKeepAliveTimeout = 'kato', /* keep alive timeout for kFSHTTPInfoSig (SInt32) */
    kFSCGITimeout       = 'cgto', /* CGI timeout for kFSHTTPInfoSig (SInt32) */
    kFSLogSize          = 'logs'  /* Log size for kFSHTTPInfoSig (UInt32) */
};
```

General Server Preferences

```
enum {
    kFSServerInfoVersion= 1
};

struct ServerInfo {
    SInt16  versionNumber;      /* Version number for this record */
    SInt16  fileServerVersion;  /* Version# for file server */
    SInt16  userActivityLimit;  /* User activity limit in % */
    SInt16  maxLogin;           /* Maximum client connections */
    SInt16  maxGuestAccess;     /* Maximum number of guest and anonymous logins */
    SInt16  shutdownMinutes;    /* Minutes until shutdown */
};
typedef struct ServerInfo ServerInfo;
```

Login Greeting Server Preferences

```
enum {
    kFSServerGreetingVersion= 1
};

struct ServerGreeting {
    SInt16  versionNumber;  /* Version number for this record */
    Str199  greetingMsg;    /* Login greeting */
};
typedef struct ServerGreeting ServerGreeting;
```

Cache Server Preferences

```
enum {
    kFSCacheInfoVersion= 1
};
```

```
struct CacheInfo {
    SInt16  versionNumber;  /* version number for this record */
    SInt32  cacheForOthers; /* memory to be reserved for other applications */
};
typedef struct CacheInfo CacheInfo;
```

Idle User Server Preferences

```
/* idleFlag bits. */
enum {
    kFSIdleDisconEnabledMask    = 0x0001,   /* Allow to disconnect idle users */
    kFSDisconExeptOpenFilesMask = 0x0002,   /* If true, disconnect except open file
                                               users */
    kFSDisconSuperUserMask      = 0x0004,   /* If true, isconnect idel super users */
    kFSDisconNormalUserMask     = 0x0008,   /* If true, disconnect idle normal users */
    kFSDisconGuestUserMask      = 0x0010    /* If true, disconnect idle guest users *
};

enum {
    kFSIdleUserInfoVersion  = 1
};

struct IdleUserInfo {
    SInt16  versionNumber;  /* Version number for this record */
    SInt16  idleFlag;       /* See above */
    SInt16  idleMinute;     /* Nax number of minutes for idle users */
    Str199  disconnectMsg;  /* Disconnect message */
};
typedef struct IdleUserInfo IdleUserInfo;
```

Administrator Information Server Preferences

```
enum {
    kFSAdminInfoVersion = 1
};
```

```
struct AdminInfo {
    SInt16  versionNumber;  /* Version number for this record */
    Str31   namePhone;      /* Name and phone number */
    Str31   organization;   /* Organization */
};
typedef struct AdminInfo AdminInfo;
```

HTTP Server Preferences

```
/* Flag bits. */
enum {
    kFSMultiDomainEnabled   = 0x0001,   /* Allow multi-domain support */
    kFSDirListingEnabled    = 0x0002,   /* Allow directory listing */
    kFSAutoShareWebFolder   = 0x0004    /* If on, web folder share point at start up;
                                           server will clear */
};

enum {
    kFSHTTPServiceInfoVersion   = 2
};

enum {
    kFSHTTPEnable       = 1,    /* Enable HTTP */
    kFSHTTPLogEnable    = 1,    /* Enable HTTP Log */
    kFSHTTPGuestEnable  = 1     /* Guest turned on for HTTP */

};

struct HTTPServiceInfo {
    SInt16  versionNumber;  /* Version number for this record */
    SInt16  enabled;        /* Enable HTTP, 1-enable, 0-not enable */
    SInt16  status;
    SInt16  logEnabled;     /* Enable HTTP Log, 1-enable, 0-not enable */
    SInt16  maxConnect;     /* HTTP maximum client connections */
    SInt16  guestEnabled;   /* Whether guest allowed for HTTP */
    SInt32  flag;           /* New field for ASIP6 *
};
typedef struct HTTPServiceInfo HTTPServiceInfo;
```

```
struct HTTPFolderRec {
    SInt16      vRefNum;        /* HTML folder path vRefNum */
    Str27       volumeName;     /* HTML folder path volume name */
    UInt32      volCreateDate;  /* HTML folder path volume creation date */
    SInt32      dirID;          /* HTML folder path DirID */
};
typedef struct HTTPFolderRec HTTPFolderRec;

struct HTTPFileRec {
    Str255  partialPath; /* Partial path to HTML file starting from the Web folder */
};
typedef struct HTTPFileRec HTTPFileRec;

enum {
    kFSPlugInEnable         = 1,    /* Enable Plug-ins*/
    kFSPlugInLoggingEnable  = 1     /* Enable Plug-in logging*/};

struct HTTPPlugInsRec {
    SInt16  pluginEnable;       /* Plug-ins,  1-enable, 0-not enable */
    SInt16  loggingEnable;      /* Plug-in logging, 1-enable, 0-not enable */
    SInt32  memSize;            /* Plug-in memory allocation */
    FSSpec  preProcessorSpec;   /* Preprocessor plugin spec */
    FSSpec  postProcessorSpec;  /* Postprocessor plugin spec */
    FSSpec  errorSpec;          /* Error plugin spec */
};
typedef struct HTTPPlugInsRec HTTPPlugInsRec;

struct VolSpec {
    Str32   vName;              /* Pascal string because FSSpec uses pascal string*/
    UInt8   filler;
    SInt16  vRefNum;
    UInt32  vCreateDate;
};
typedef struct VolSpec VolSpec;

enum {
    kFSHTTPWebDirCountRecVersion = 1
};
```

```
struct HTTPWebDirCountRec {
    UInt16  count;          /* How many HTTPWebDirRec there are */
    SInt16  version;        /* Version of the HTTPWebDirRec */
};
typedef struct HTTPWebDirCountRecHTTPWebDirCountRec;

enum {
    kFSWebFolderEnabled     = 1
};

enum {
    kFSAddressTypeDNS       = 0,
    kFSAddressTypeIPAddress = 1
};

struct HTTPWebDirRec {
    UInt16  enabled;        /* 1 == this web folder enabled, 0 == disabled */
    SInt16  addressType;    /* Indicates how to decode addressText: 0 == DNS name,
                               1 = IP Address (in ASCII) */
    Str63   addressText;    /* Domain name or IP address (in ASCII) of this web folder
                               (pascal str) See addressType field */
    UInt16  portNumber;     /* Port number for this web folder's IP address
                               (InetPort) */
    VolSpec volSpec;        /* Volume specifier */
    SInt32  dirID;          /* Dir ID of this web folder */
};
typedef struct HTTPWebDirRec HTTPWebDirRec;
```

FTP Server Preferences

```
enum {
    kFSFTPServiceInfoVersion    = 1
};

struct FTPServiceInfo {
    SInt16  versionNumber;      /* version number for this record */
    SInt16  ftpEnable;          /* Enable FTP, 1-enable, 0-not enable */
    SInt16  status;
```

```
    SInt16  ftpAnonymousEnable; /* Enable Anonymous Login, 1-enable, 0-not enable */
};
typedef struct FTPServiceInfo FTPServiceInfo;
```

AFP Server Preferences

```
enum {
    kFSAFPServiceInfoVersion    = 1
};

struct AFPServiceInfo {
    SInt16  versionNumber;      /* version number for this record */
    SInt16  afpOverTCPIPEnable; /* Enable AFP over TCPIP, 1-enable, 0-not enable */
    SInt16  afpOverTCPIPStatus;
    SInt16  afpOverATalkEnable; /* Enable AFP over AppleTalk, 1-enable, 0-not enable */
};
typedef struct AFPServiceInfo AFPServiceInfo;
```

SMB Server Preferences

```
enum {
    kFSSMBServiceInfoVersion    = 1
};

enum {
    kFSSMBMaxNetBIOSnameLength  = 15,
    kFSSMBMaxWorkGroupLength    = 15,
    kFSSMBMaxCommentLength      = 43,
    kFSSMBWINSAddressLength     = 62
};

struct SMBServiceInfo {
    SInt16  versionNumber;  /* version number for this record */
    SInt16  smbEnable;      /* Enable SMB, 1-enable, 0-not enable */
    SInt16  status;
    SInt16  guestEnabled;   /* Whether guest allowed for SMB */
    Str31   netBIOSname;    /* Server name. Limited to 15 bytes */
    Str31   workGroup;      /* NETBIOS group name.  Limited to 15 bytes */
```

```
    Str63   comment;        /* ASCII comment for the server. Limited to 43 bytes */
    SInt16  winsEnabled;    /* Enable WINS, 1-enable, 0-not enable */
    Str63   winsAddress;    /* WINS IP address */
};
typedef struct SMBServiceInfo SMBServiceInfo;
```

MIME Type Change Notifications

```
enum {
    kFSMimeTypeChangedVersion   = 1
};

struct MimeTypeChanged {
    SInt16  version;    /* version number for this record */
    SInt32  counter;    /* no meaning, just for notification */
};
typedef struct MimeTypeChanged MimeTypeChanged;
```

IP Filtering

```
enum {
    kFSIPFilterAllow    = 1
};

/* Structure of individual filters */

struct IPFilterStruct {
    SInt16  allow;      /* 0 for disallow, 1 for allow */
    UInt32  highIPBytes;
    UInt32  lowIPBytes;
};
typedef struct IPFilterStruct IPFilterStruct;

enum {
    kFSMaxIPFilters = 10
};
```

```
struct IPFilterInfo {
    SInt16          versionNumber;  /* Version number for this record */
    SInt16          numFilters;     /* Number of IP filters */
    IPFilterStruct  filters[10];
};
typedef struct IPFilterInfo IPFilterInfo;
```

Maximum Connection Information

Tags for this record are type `kFSCreatorSig` and attribute `kFSMaxConnection`. This value is used if the maximum connection number encoded in the serial number is 250.

```
enum {
    kFSServerMaxConnectionInfoVersion   = 1
};

struct ServerMaxConnectionInfo {
    SInt16  version;
    SInt32  flag;   /* Reserved for future use */
    SInt32  maxConnection;
};
typedef struct ServerMaxConnectionInfo ServerMaxConnectionInfo;
```

Mail Server User Attributes

This section describes the constants and data types defined in AppleShareMailServerRegistry.h for the Mail server user attributes.

Signature and Type

The following enumeration defines the signature and type for Mail server user attributes:

```
enum {
    kMUMailServerSignature  = 'mail'),
    kMU60Attributes         = 'mU60')
};
```

Constants

```
enum {
    kMUMaxSMTPForwardLength = 255,
    kMUFingerprintLength    = 16
};
```

MU60Attributes Structure

The AppleShare IP Mail server uses the `MU60Attributes` structure to store information about a user's mail configuration. The maximum size of the `MU60Attributes` structure is 500 bytes.

The `MU60Attributes` structure is defined as follows:

```
typedef struct {
    UInt32      version;
    UInt32      mailUserFlags;
    Str32Field  atalkForwardName;
    Str32Field  atalkForwardServer;
    Str32Field  atalkForwardZone;
    char        smtpForward[256];
    char        fingerprint[16];
    UInt32      notifyIPAddress;
}; typedef struct MU60Attributes MU60Attributes;
```

Field descriptions

`version` The version of AppleShare IP that this structure supports. For AppleShare IP 6.0, the value of version is `kASRMailDefsVersion6`.

`mailUserFlags` Bit values that describe the user's mail configuration. See "ASDMailUserFlags" on page A-86 for details.

`atalkForwardName`	Contains the name of the user to whom mail is to be forwarded if forwarding over AppleTalk is configured for this user.
`atalkForwardServer`	Contains the name of the server to which mail is to be forwarded if forwarding over AppleTalk is configured for this user.
`atalkForwardZone`	Contains the AppleTalk zone of the server to which mail is to be forwarded if forwarding over AppleTalk is configured for this user.
`smtpForward`	Contains the name of the SMTP server to which mail is to be forwarded if forwarding over TCP/IP is configured for this user. The maximum length of `smtpForward` is 256 bytes.
`fingerprint`	Contains a 16-byte value that uniquely identifies this user; initialize to zero when you create the attribute.

▲ **WARNING**

If your application rewrites a user's mail attributes, it should read the current value of the `fingerprint` field before it writes the mail attributes. Failure to maintain the integrity of the `fingerprint` field will corrupt the user's mail attributes and require an administrator to intervene in order to recover the user's mail.

`NotifyIPAddress`	If `kMUUseSpecificIPAddr` is set in `mailUserFlags`, contains the IP address to which the user wants notification of new mail to be sent.

ASDMailUserFlags

The `ASDMailUserFlags` enumeration defines the bit values for the `mailUserFlags` of the `MU60Attributes` structure as follows:

```
enum {
    KMUUserEnabledMask      =   0x0000000F,
    kMUAPOPRequired         =   0x00000004,

    kMUForwardingMask       =   0x000000F0.
    kMUNoForwarding         =   0x00000010,
    kMUForwardSMTP          =   0x00000020,
```

```
    kMUForwardATalk         =   0x00000040,

    kMUIMAPOPFlagsMask      =   0x00000F00,
    kMUPOPEnabled           =   0x00000100,
    kMUIMAPEnabled          =   0x00000200,

    kMUNotificationMask     =   0x0000F000,
    kMUNotificationON       =   0x00001000,
    kMUUseLastIPAddr        =   0x00002000,
    kMUUseSpecificIPAddr    =   0x00004000,

    kMUSharedBoxFlagMask    =   0x000F0000,
    kMUSeparatePOPAndIMAP   =   0x00010000,
    kMUShowPOPInIMAP        =   0x00020000,

    // Legacy, do not use beyond 5.0.x versions.
    kMUMailEnabled          =   0x00000001,
    kMULoginEnabled         =   0x00000002
};
```

`KMUUserEnabledMask` Mask for setting user enable flags.

`kMUAPOPRequired` If set, indicates that the user must use the Authentication Post Office Protocol (APOP) to authenticate the connection.

`kMUForwardingMask` Mask for setting forwarding options. One of `kMUNoForwarding`, `kMUForwardSMTP`, or `kMUForwardATalk` must be set.

`kMUNoForwarding` If set, indicates that mail is enabled for this user on this mail server.

`kMUForwardSMTP` If set, indicates that mail for this user is to be forwarded over TCP/IP to an SMTP mail server.

`kMUForwardATalk` If set, indicates that mail for this user to be forwarded over AppleTalk to an SMTP server.

`kMUIMAPPOPFlagsMask` Mask for setting mail protocol. One or both of these constants can be set: `kMUPOPEnabled` and `kMUIMAPEnabled`.

`kMUIPOPEnabled` If set, indicates that the user can use the Post Office Protocol version 3 (POP3) or the PASS protocol to connect to this mail server.

`kMUIMAPEnabled` If set, indicates that the user can use the Interactive Mail Access Protocol (IMAP) or the PASS protocol to connect to this mail server.

`kMUNotificationMask`
Mask for setting mail notification. If `kMUNotificationOn` is set, `kMUUseLastIPAddr` or `kMUUseSpecificIPAddr` must be set.

`kMUNotificationON` If set, indicates that the user wants to be notified when the mail server receives new mail for this user.

`kMUUseLastIPAddr` If set, indicates that the notification of new mail should be sent to the user's most recent IP address.

`kMUUseSpecificIPAddr`
If set, indicates that the notification of new mail should be sent to the IP address stored in the `notifyIPAddress` field of the `MU60Attributes` structure.

`kMUSharedBoxFlagMask`
Mask for determining how inboxes are to be displayed if the user can connect using POP and IMAP. `kUMSeparatePOPAndIMAP` can be set, or `kMUSeparatePOPAndIMAP` and `kUMShowPOPInIMAP` can be set.

`kMUSeparatePOPAndIMAP`
If set, indicates that the user's POP inbox is not to be displayed when the user uses IMAP to check his or her mail.

`kMUShowPOPAndIMAP`
If set, indicates that the user's POP inbox is to be displayed when the user uses IMAP to check his or her mail.

`kMUMailEnabled` Obsolete in the AppleShare IP 6.0 Mail server.

`kMULoginEnabled` Obsolete in the AppleShare IP 6.0 Mail server.

Index

G

I

K

L

M

N

O

P

R

S

T

U

V