AppleSearch Client Developer's Guide

Apple Computer, Inc.

20525 Mariani Avenue
Cupertino, CA 95014-6299
(408) 996-1010

Simultaneously published in the United States and Canada.

Contents

Preface

AppleSearch is full-text search-and-retrieval workgroup software that allows users to retrieve customized information from unstructured document files (for instance, correspondence, electronic mail, and reports) that reside on an AppleSearch server. This document is intended for developers who want to write their own custom AppleSearch client user interface to the AppleSearch server or who want to integrate certain features of the AppleSearch client software into their own existing applications.

What you need to know

Before you use this guide you should be familiar with the AppleSearch Client and Server applications. You should be familiar with key features of AppleSearch Client such as reporters, article retrieval, scheduled searches, and updates. You should also be familiar with Apple events and know how to program in the Macintosh environment. For information on these subjects, see the references in "Suggested Reading," later in this Preface.

What this guide contains

Here is a description of what you'll find in this guide:

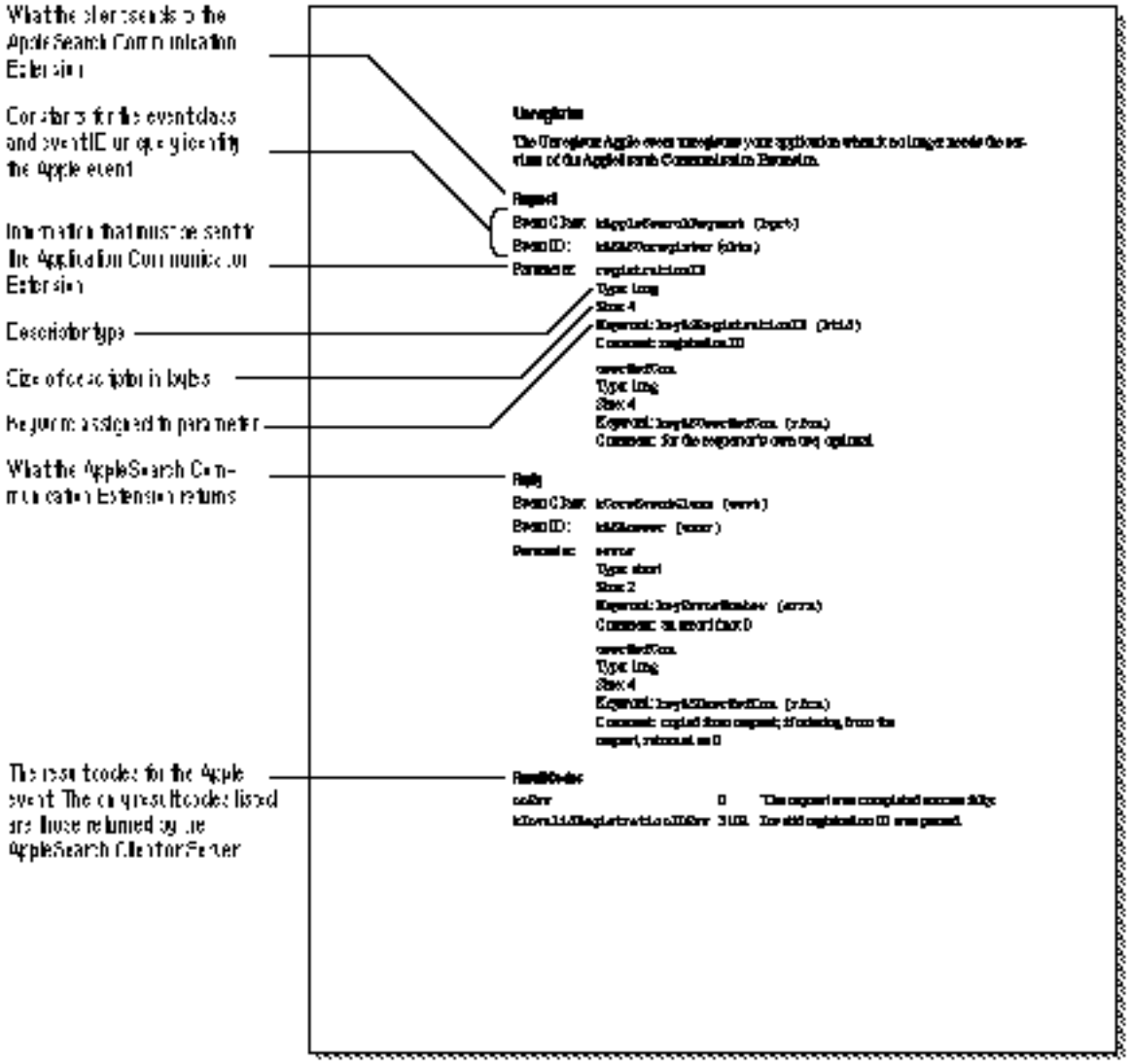
- Chapter 1, "Architectural Overview," presents an overview of the AppleSearch client/server architecture and describes key concepts of the AppleSearch Client application programming interface (API).
- Chapter 2, "Apple Events Application Programming Interface," describes how to use the Apple events version of the API and then gives a detailed specification of each Apple event.
- Chapter 3, "C Library Application Programming Interface," describes how to use the C API and then gives a detailed description of each C function call.
- Appendix A, "Constants and Errors," presents a summary of AppleSearch API errors and related constants.
- Appendix B, "Apple Events Summary," presents a summary of the Apple events API.
- Appendix C, "C Library Summary," presents a summary of the C Library API.

Conventions used in this guide

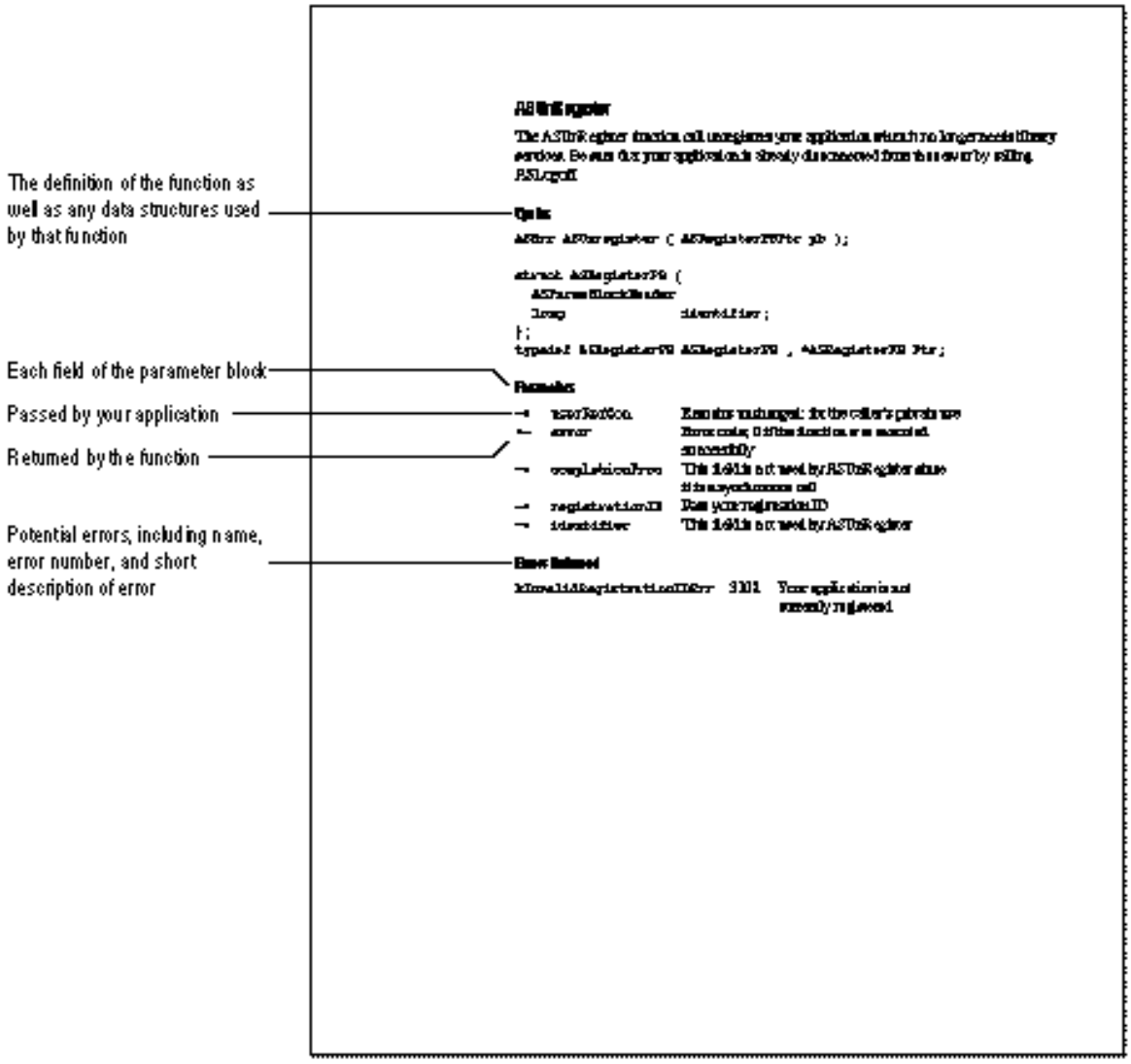
The Courier font is used for event classes, event IDs, parameters, keywords, and result code messages. This font is also used to indicate code elements that must be typed into your program.

Format of API descriptions

The description of each Apple event API contains the finformation shown in the following figure:



The description of each C Library API contains the finformation shown in the following figure:



About the AppleSearch Client Developer's Kit

This manual, the *AppleSearch Client Developer's Guide*, is just one component in the AppleSearch Client Developer's Kit. The kit also includes the following items:

- AppleSearch client software
- a C Library object file
- header files for the C Library and Apple events

- an Apple events tool that allows you to log Apple event activity on your system
- sample code and a SimpleText file that explains how the sample code functions
- the AppleSearch Update File Format Developer's Kit, which includes the *AppleSearch Update File Format Reference* and a set of utility routines

After you read this guide, refer to the *AppleSearch Update File Format Reference* for information about the format of AppleSearch-generated update documents.

Note: In order to test and use the AppleSearch client capability, you must purchase the AppleSearch server software.

Suggested reading

Here is a list of reference materials that you may find helpful:

- *AppleSearch User's Guide* (included in the AppleSearch Server and Client for Macintosh, version 1.5) describes how to use the AppleSearch client to search for and retrieve information.
- *AppleSearch Administrator's Guide* (included in the AppleSearch Server and Client for Macintosh, version 1.5) describes how to set up and maintain an AppleSearch Server.
- *AppleSearch Update File Format Reference* (included in the AppleSearch File Format Developer's Kit, which is in the AppleSearch Client Developer's Kit) describes the format of AppleSearch-generated update documents.
- *Inside Macintosh: Interapplication Communication* provides a complete description of Apple events, explains how to send and receive Apple events, and includes reference information for all Apple Event Manager routines. *Inside Macintosh: Interapplication Communication* (Addison-Wesley Publishing Co., 1993) is available in bookstores.
- *Inside Macintosh: Processes* provides background information on the parts of the Macintosh Operating System that allow you to manage processes and tasks. *Inside Macintosh: Processes* (Addison-Wesley Publishing Co., 1992) is available in bookstores.

Software licensing

For information on developer licensing, please contact the Software Licensing Department:

Software Licensing Department
Apple Computer, Inc.
2420 Ridgepoint Drive, M/S 198-SWL
Austin, TX 78754
U.S.A.
Telephone: (512) 919-2645
AppleLink: SW.LICENSE

1 Architectural Overview

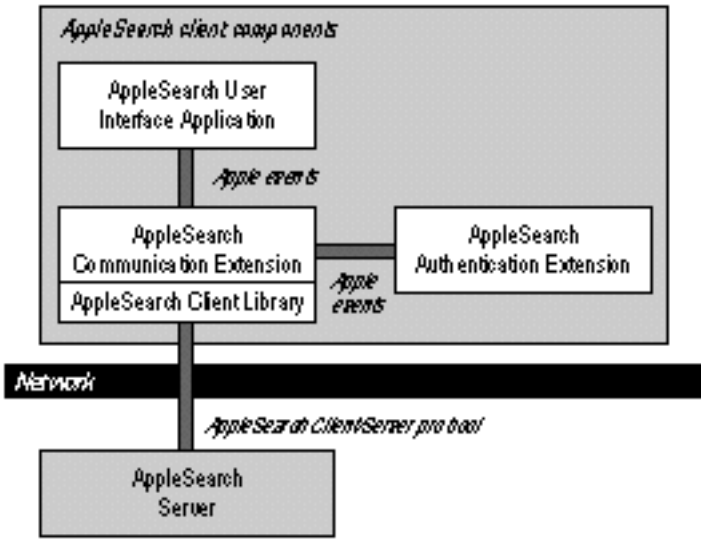
This chapter presents an overview of the AppleSearch client/server architecture and describes key concepts required to use the AppleSearch Client application programming interface (API).

AppleSearch software components

AppleSearch Client consists of three components:

- AppleSearch User Interface application
- AppleSearch Communication Extension—the background task application
- AppleSearch Authentication Extension—the log-on application

The following diagram shows the relationships among these components and between the client and the AppleSearch server software.



The AppleSearch User Interface application is a MacApp 3.0–based Macintosh application that provides the user interface functionality of the client. This is the only one of the three components that can be launched by the user.

The AppleSearch Communication Extension has no menus and cannot display any windows. Because this extension runs in the background, generating update files at delivery time, the user does not have to keep the user interface application running in order to receive updates. The AppleSearch Communication Extension is responsible for communicating with the AppleSearch server; thus the user can perform operations such as reporter creation and article retrieval only if it is running. (If it is not running, the user can only open and view existing article and update files on the mounted volumes.) The AppleSearch Communication Extension is stored in the user's Extensions folder.

The AppleSearch Authentication Extension displays a connection dialog box that allows the user to open a session with an AppleSearch server. Once the user completes the log-on process, the AppleSearch Authentication Extension sends session information to the AppleSearch Communication Extension and then quits. The AppleSearch Authentication Extension application is stored in the user's Extensions folder.

All three components communicate with each other by way of Apple events. The AppleSearch Communication Extension and the AppleSearch Server communicate byusing the AppleSearch Client/Server protocol (an Apple proprietary protocol). For instance, when a user double-clicks a reporter in the AppleSearch window, the following occurs:

- The AppleSearch Client sends an Apple event to the AppleSearch Communication Extension, asking for information on that reporter.
- The AppleSearch Communication Extension retrieves the information from the server by using the AppleSearch Client/Server protocol.
- The AppleSearch Communication Extension passes the information to the AppleSearch Client in a reply event to the original request.
- The AppleSearch Client displays the reporter window, which contains information about that reporter.

Writing your own client application

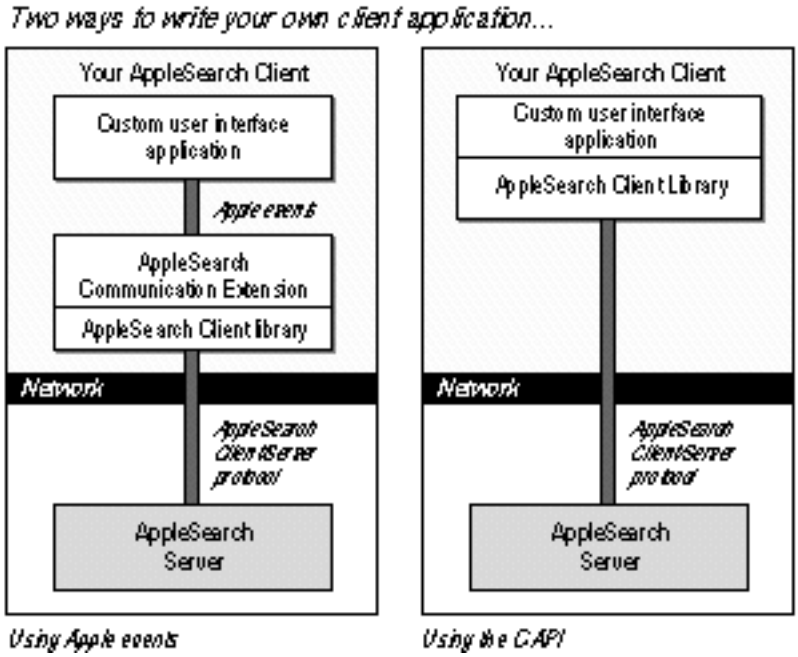
You can write your own client application to access AppleSearch servers in either of two ways:

- You can write an application that replaces the AppleSearch User Interface application provided by Apple. In this case, your application must use Apple events to communicate with the AppleSearch Communication Extension, which in turn communicates with the AppleSearch server.
- You can link your application directly to the AppleSearch Client Library, a part of the AppleSearch Communication Extension. You will need to make C function calls to the AppleSearch Client Library in order to access the server.

The advantage of using Apple events is that you won't have to change your user interface application if Apple replaces the transport-layer-level protocol in future releases of the product. Also, if your application supports update delivery using the AppleSearch Communication Extension, users can receive updates without having their user interface application running.

The advantage of using the C API is you have more flexibility than you do with Apple events. For instance, you can make a C function call that allows a user to connect to the server without using the AppleSearch Authentication Extension.

The following diagram shows how the various components interact.



Client/server connection

The AppleSearch Client/Server protocol resides on top of the Program-to-Program Communications (PPC) protocol. The AppleSearch Client/Server protocol uses PPC to send and receive high-level message blocks between the AppleSearch Communication Extension and the AppleSearch Server.

Note: In future versions of AppleSearch, another transport-layer-level protocol besides PPC may be used.

The client/server connection is established when the client's log-on request is successfully processed by the server. In AppleSearch, a client can connect to only one server at a time. Once the connection is made, the user can perform all AppleSearch functions—including searching information sources, retrieving articles, and adding reporters to the server—until the connection is terminated. If the user is connected as a guest, the client cannot make reporter- or object-database-related requests to the server. (The server object database is described in the section "Server Object Database" later in this chapter.)

The client/server connection is terminated in one of three ways:

- When a client logs off, a Logoff call is issued; the AppleSearch client indicates a disconnect to the AppleSearch Communications Extension, which makes the Logoff call to the AppleSearch server.
- When a client encounters a fatal error, such as a PPC communication error, it disconnects itself.
- When the communication layer notifies the client and/or server that the connection is no longer there, the server disconnects itself.

Information sources

The local information sources that users can search are shared volumes or folders, containing documents in various formats, that reside on an AppleSearch server. Users must have appropriate access privileges to folders on the server before they can view or search information sources on that server; access privileges are provided by AppleShare or by Macintosh file sharing in System 7.

AppleSearch also supports WAIS (wide area information source) searches. A WAIS search allows you to search through material on outside sources such as the Internet.

Several of the API calls described in this document are related to information sources. For example, the enumerate-sources call returns a list of information sources available to the client on the currently connected server.

Note: AppleSearch ranks articles in terms of how relevant they are to a user's search request. It will rank the articles for each information source; that is, it will not determine relevancy across more than one information source at a time.

Immediate searches

When an immediate search is initiated by the user, its results are returned immediately by the server. (Scheduled searches, which return results at a specified time, are discussed later in this chapter.) To perform an immediate search, your application must make an AppleSearch open-search-session call to open a search session, and then make an AppleSearch search call with appropriate parameters. Your application can make as many search calls as you want within one search session and can also open as many search sessions simultaneously as resources on the server allow. When your application is done searching, your application must close the session by issuing an AppleSearch close-search-session call.

As a result of an immediate search, the server returns a list of articles that match the user's search request, ranked in order of relevance within each information source. The list shows the article ID, along with its modification date, title, and file size. To get the actual article text or information about an article, you must make separate calls, which are described in the next section.

Article retrieval

There are three API calls related to article retrieval:

- An AppleSearch get-article-text call returns the text of an article.
- An AppleSearch get-article-info call returns an article's attributes such as its modification date, title, and size.
- An AppleSearch get-article-matches call returns a list of offsets and lengths, which represent the words matching the original query (used for the highlight matching words feature in the AppleSearch Client software). This list is not returned in the case of a WAIS search.

To make these calls, your application must have the article's information source ID and article ID, both of which are returned by a search call. Also, to get matches, you must have a search session ID for the search session which the article was found, and the search engine must be open.

Original file retrieval

When AppleSearch retrieves articles, it extracts plain text from various document types, ignoring graphics and formatting instructions. The text of each article is returned to the client by means of individual calls to a get-article-text call. The server uses XTND file-format translators to extract the text.

A get-original-file call allows you to retrieve an article in its original format. For instance, if the original article on the server was formatted with MacWrite, a get-original-file call would retrieve the article as a MacWrite document. To make this call, your application needs the article's information source ID and article ID.

Scheduled searches

Reporters are queries that are saved with parameters specified by the user. They are stored as database objects on the server. (See the next section for more information about database objects.) Each reporter object contains its search query, information sources to search, other search options, and its delivery date, time, and update file destination. The AppleSearch Client Library retrieves reporter object information from the server in order to deliver updates or to allow users to modify the object information.

Your application can use API calls to add, delete, or modify reporters, or to get reporter objects from and to the server. Each reporter can be active or inactive; only those reporters that are active deliver updates. The active flag is one of the reporter object attributes and can be set by an AppleSearch modify-reporter call.

Server object database

The AppleSearch server maintains a database, called the *server object database*, that stores a list of reporters and other relevant information. Your application can use this database to store custom information. A set of routines is provided to allow your application to gain access to the database. For instance, an add-object call adds custom objects to the currently connected server's object database, and an enumerate-objects call lists all objects of a specific type in the database. A database object stored by one registered user is accessible only by that user.

Note: Internally, a reporter is a special kind of object that the AppleSearch Client Library uses. An object is any chunk of information—basically, an array of bytes—with a type and ID. Each object of the same type in the database must have a unique ID for each registered user.

2 Apple Events Application Programming Interface

This chapter describes the Apple events application programming interface (API) provided by AppleSearch. It describes how to use the API and then gives a detailed specification for each Apple event.

Note: In AppleSearch, the AppleSearch client uses Apple events as a private, interapplication transport mechanism for accessing the AppleSearch Communication Extension API. Thus, AppleSearch does not support the Object Support Library, nor does it establish a standard information access suite.

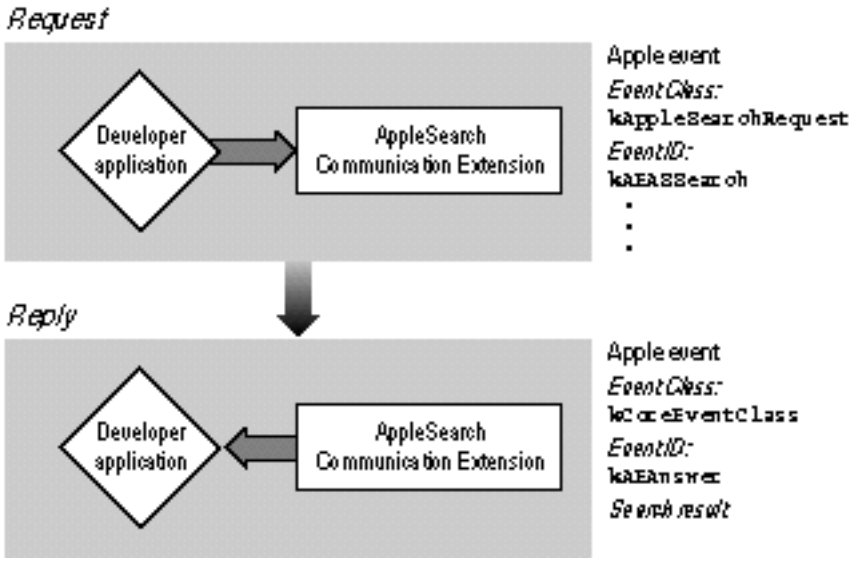
Sending and receiving AppleSearch Apple events

Each AppleSearch request is an Apple event that your application must send to the AppleSearch Communication Extension. You can specify the AppleSearch Communication Extension by its creator type, which is `kAppleSearchCommunicationSignature`. The event class of the request should be `kAppleSearchRequest`. Your application can send it in any mode you prefer—for example, `kAEQueueReply` or `kAEWaitReply`. Both the creator type and event class are defined in the header file `AppleSearchAEAPI.h`.

When the AppleSearch Communication Extension receives a request from your application, it sends a request to a server asynchronously. When a reply comes back from the server, the AppleSearch Communication Extension sends a reply Apple event back to your application.

For example, to conduct an immediate search you need to send an Apple event of event class `kAppleSearchRequest` and event ID `kAEASSearch` with appropriate parameters. The AppleSearch Communication Extension packages the search parameters into an appropriate parameter block and then sends a search request to the server. After the server responds to the request, a reply of event class `kCoreEventClass` and event ID `kAEAnswer`, which contains a search result, will be returned.

The following diagram shows the relationship between an AppleSearch request and its replies.



The AppleSearch Communication Extension

The AppleSearch Communication Extension is a background-only process that receives Apple events from another application (such as your custom client) and in turn communicates with an AppleSearch server. For example, when your application sends a Search event to the AppleSearch Communication Extension, it checks parameters and then issues a search request to the server it is logged on to, using the AppleSearch Client/Server protocol. When a reply comes back from the server, the AppleSearch Communication Extension sends a search reply event to your application. In essence, the AppleSearch Communication Extension appears as a server to your application.

Before sending any Apple event to the AppleSearch Communication Extension, your application must determine whether the extension is already running. If not, your application needs to find it and launch it. Since the AppleSearch Communication Extension is a system extension, it can be found in the Extensions folder. Your application can find it by making a PBCatSearch call with the AppleSearch Communication Extension's creator type and Extensions folder as the parent directory. Then, you can make an FSSpec out of the CatSearch parameter block and use it in the `LaunchApplication` trap. When launched, the AppleSearch Communication Extension automatically displays the PPCBrowser dialog box, prompting the user to select an AppleSearch server to connect to. If the user is successfully authenticated, a session is established between the client and the server. If the user does not connect to the server properly, the AppleSearch Communication Extension sends an error message to your application and quits.

The following sample code demonstrates how to find and launch the AppleSearch Communication Extension from your application.

```
//_____________________________________________________________________________
// Function:      LaunchASCommunication
// Abstract:      Determines if the AppleSearch Communication extension is
//                running.  If it is not yet running, it tries to launch it.
// Parameters:    isRunning - This value gets set to true if the extension is
//                already running, or it was successfully launched by this
//                function.
// Returns:       OSErr - If the extension wasn't already running and the attempt
//                to launch failed, the error code from the attempt is returned.
//_____________________________________________________________________________

OSErr LaunchASCommunication( Boolean* isRunning )
{
   OSErr          err = noErr;
   const OSType   kASCommunicationSignature = 'bgdp';

   if ( ApplicationIsRunning( kASCommunicationSignature ) ) {
         *isRunning = true;
   }
   else {
      err = LaunchMyExtension( kASCommunicationSignature );
      *isRunning = ( err == noErr );
   }
   return err;
}

//_____________________________________________________________________________
// Function:      ApplicationIsRunning
// Abstract:      Determine if the specified application is running on this Mac.
// Parameters:    signature - The creator signature of the desired application
// Returns:       Boolean - True if the process specified by the signature is
//                running, false if no process with that signature could be found.
//_____________________________________________________________________________

Boolean ApplicationIsRunning( OSType signature )
{
   ProcessSerialNumber  psn;
   ProcessInfoRec    info;

   psn.processInfoLength = sizeof( ProcessInfoRec );
   psn.processName = 0L;                        // we don't care about the name
   psn.processAppSpec = 0L;                     // or FSSpec of it

   psn.highLongOfPSN = 0L;                      // start from the beginning of
   psn.lowLongOfPSN = kNoProcess;               // the process list

   while ( GetNextProcess( &psn ) != procNotFound ) {
                                                // for each process
   GetProcessInformation( &psn, &info );        // get process info
```

```
   if ( info.processSignature == signature )
                                                  // if looking for signature matches
      return ( TRUE );
   }
   return ( FALSE );
}

//_____________________________________________________________________________
// Function:      LaunchMyExtension
// Abstract:      This function uses PBCatSearch to find the extension in the
//                Extensions folder, and if found, launches it with appropriate
//                parameters.
// Parameters:    signature - The creator signature of the desired Extensions
// Returns:       OSErr - noErr if the application was found in the Extensions
//                folder and launched successfully.
//                Otherwise, an appropriate error from FindFolder, PBCatSearch,
//                or LaunchApplication.
//_____________________________________________________________________________

OSErr LaunchMyExtension( OSType signature )
{
   OSErr             err;
   short             vRefNum;
   long              extensionsDirID;
   CSParam           searchPB;
   FSSpecArrayPtr myMatches;

   // first find the Extension folder
   err = FindFolder( -1, kExtensionFolderType, false, &vRefNum, &extensionsDirID );

   // if found, then find the extension wanted in the folder
   if ( err == noErr ) {
   CInfoPBRec  info1, info2;

   // setup paramBlock for the CatSearch call
   info1.hFileInfo.ioNamePtr = nil;
   // don't search on string (don't care what its name is)
   info1.hFileInfo.ioFlAttrib = 0x00;           // clear bit 4 to ask for files
   info1.hFileInfo.ioFlFndrInfo.fdCreator = signature;
                                                // signature we're interested in
   info1.hFileInfo.ioFlParID = extensionsDirID;
   // search in Extensions folder
   info1.hFileInfo.ioFlFndrInfo.fdType = 0x00;
   info1.hFileInfo.ioFlFndrInfo.fdFlags = 0x00;
   info1.hFileInfo.ioFlFndrInfo.fdLocation.h = 0x00;
   info1.hFileInfo.ioFlFndrInfo.fdLocation.v = 0x00;
   info1.hFileInfo.ioFlFndrInfo.fdFldr = 0x00;
```

Continued on following page ▶

```
    info2.hFileInfo.ioNamePtr = nil;
    info2.hFileInfo.ioFlAttrib = 0x10;
    info2.hFileInfo.ioFlFndrInfo.fdType = 0x00;              // mask out type
    info2.hFileInfo.ioFlFndrInfo.fdCreator = 0xFFFFFFFF;     // mask for creator
    info2.hFileInfo.ioFlFndrInfo.fdFlags = 0x00;             // mask out flags
    info2.hFileInfo.ioFlFndrInfo.fdLocation.h = 0x00;        // mask out location
    info2.hFileInfo.ioFlFndrInfo.fdLocation.v = 0x00;        // mask out location
    info2.hFileInfo.ioFlFndrInfo.fdFldr = 0x00;              // mask out folder
    info2.hFileInfo.ioFlParID = extensionsDirID;             // max folder ID

    myMatches = (FSSpec*) NewPtrClear( sizeof(FSSpec) );

    if ( myMatches == nil ) {
        err = MemError();
    }
    else {
        searchPB.ioCompletion = nil;        // no completion routine
        searchPB.ioNamePtr = nil;           // no name, use vRefNum
        searchPB.ioVRefNum = -1;            // vRefNum -1 = Boot volume
        searchPB.ioMatchPtr = myMatches;    // pointer to result array of FSSpecs
        searchPB.ioReqMatchCount = 1;       // we only want one match (only allocated
                                            // enough memory for one, should only
                                            // find one)
        searchPB.ioSearchBits = fsSBFlFndrInfo
                              + fsSBFlAttrib
                              + fsSBFlParID;       // only interested in finder info
        searchPB.ioSearchInfo1 = &info1;          // pointer to first info block
        searchPB.ioSearchInfo2 = &info2;          // pointer to second info block
        searchPB.ioSearchTime = 0;                // no timeout on search
        searchPB.ioCatPosition.initialize = 0;    // start at beginning of catalog
        searchPB.ioOptBuffer = nil;               // no optional buffer
        searchPB.ioOptBufSize = 0;                // since no optional buffer, its
                                                  // size is 0

        err = PBCatSearch( &searchPB, false );
    }
}

    // if we found the extension, now try to launch it
    if ( err == noErr ) {

        LaunchParamBlockRec     launchPB;
        AppParametersPtr        appPBPtr;
```

```
      launchPB.launchBlockID = extendedBlock;
      launchPB.launchEPBLength = extendedBlockLen;
      launchPB.launchFileFlags = 0;
      launchPB.launchControlFlags = launchNoFileFlags + launchContinue +
      launchUseMinimum;
      launchPB.launchAppSpec = &searchPB.ioMatchPtr[0];
      launchPB.launchAppParameters = appPBPtr;

      err =  LaunchApplication( launchPB );

      if ( appPBPtr != nil )  {
         DisposePtr( (Ptr) appPBPtr );
      }
   }

   if ( myMatches != nil ) {
      DisposePtr( (Ptr)myMatches );
   }

   return err;
}
```

Connecting to a server

Before your application can connect to a server, it must send a Register event to register with the AppleSearch Communication Extension. Once your application receives a registration ID in the reply event, you can send a Logon event with that registration ID. If the Logon reply event comes back without any errors in its `keyErrorNumber` parameter, then your application has successfully logged on to the server. If the AppleSearch Communication Extension is already connected to a server, you may get a `kAlreadyLoggedOnErr` error.

Apple event API calls

This section specifies in detail each Apple event request and its replies.

Register

The Register Apple event registers your application with the AppleSearch Communication Extension. This request must be issued before any other request. The application should pass its registration ID, which is returned in the reply, in all subsequent requests.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASRegister (rstr)`

Parameter: `userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: an error if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: use this ID in all subsequent requests

Result Codes

`noErr`	0	The request was completed successfully.
`kMaxRegisteredErr`	3101	The AppleSearch Communication Extension cannot handle any more user interface applications at this time.

Unregister

The Unregister Apple event unregisters your application when it no longer needs the services of the AppleSearch Communication Extension.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASUnregister (urtr)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: an error if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.

Logon

Your application must send this Apple event to the AppleSearch Communication Extension in order to log on to an AppleSearch server. The AppleSearch Communication Extension will use the AppleSearch Authentication Extension to display the PPCBrowser dialog box so that the user can select a server. If the user selects a server and is authenticated, information about the server and log-on status is returned in the reply.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASLogon (cnct)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`deliverUpdate`
Type: Boolean
Size: 2
Keyword: `keyASDeliverUpdateFlag (dupd)`
Comment: pass `true` in this descriptor if you want the AppleSearch Client Library to automatically deliver updates while connected; optional (defaults to true)

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: error code

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`isGuest`
Type: Boolean
Size: 2
Keyword: `keyASIsGuest (igst)`
Comment: `true` if the user logged on as a guest

`serverName`
Type: char
Size: variable
Keyword: `keyASServerName (svrn)`
Comment: connected server's name

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kAlreadyLoggedOnErr`	3103	The AppleSearch Communication Extension is already logged on to an AppleSearch server.
`kLogOnCancelErr`	3104	The user canceled the log-on process in the PPCBrowser or authentication dialog box.

Set Auto Logon

The Set Auto Logon Apple event is sent to the AppleSearch Communication Extension to set or clear the automatic log-on feature of the extension. If your application turns this option on, the extension automatically displays the PPCBrowser dialog box at startup time so that the user can connect to the server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASSetAutoLogonFlag (salg)`

Parameter: `autoLogonFlag`
Type: Boolean
Size: 2
Keyword: `keyASAutoLogonFlag (alin)`
Comment: pass `true` if you want the option to be turned on

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: an error if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

Result Codes

`noErr`	0	The request was completed successfully.

Get Auto Logon

The Get Auto Logon Apple event is sent to the AppleSearch Communication Extension to get the status of the auto-log-on flag.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetAutoLogonFlag (galg)`

Parameter: `userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: an error if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`autoLogonFlag`
Type: Boolean
Size: 2
Keyword: `keyASAutoLogonFlag (alin)`
Comment: returns true if it is set

Result Codes

`noErr`	0	The request was completed successfully.

Logoff

The Logoff Apple event disconnects the client from the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASLogoff (loff)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: an error if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Status Changed

Initiated by the AppleSearch Communication Extension, the Status Changed Apple event is sent to all registered applications when the server's status has been changed. One parameter, `keyChangeType`, is passed to indicate what type of change has occurred.

Request

Event Class: `kAEASAppleSearchRequest (bgrt)`

Event ID: `kAEASStatusChanged (chng)`

Parameter: `changeType`
Type: long
Size: 4
Keyword: `keyChangeType (ctyp)`
Comment: indicates the nature of the change that occurred on the server

Notes

The `keyChangeType` parameter may be one of the following:

- `kServerDownChangeType`—the server has gone down.
- `kReporterListChangeType`—the list of reporters for the user has been changed.

Enumerate Sources

The Enumerate Sources Apple event returns a list of information sources available on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASEnumerateSources (esrc)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`numberOfSources`
Type: long
Size: 4
Keyword: `keyASNumberOfSources (nums)`
Comment: number of sources in the list

`sourcesList`
Type: list
Size: variable
Keyword: `keyASSourceList (dlst)`
Comment: list of `InfoSourceRecord` structures

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Notes

Each element in the `sourcesList` parameter in the final reply is a block of bytes that can be cast to the `InfoSourceRecord` structure, which is shown here.

```
struct InfoSourceRecord {
     long      sourceID;
     Str31     sourceName;
     long      sourceFlags;
};
```

For example, once you put the `sourcesList` parameter into your own buffer, `myBuffer`, by using `AEGetParamPtr`, you can obtain the name of the first source in the list, as follows:

```
str31                   firstName;
InfoSourceRecord        myBuffer;
firstName = myBuffer[0].sourceName;
```

The highest byte of sourceFlags in the InfoSourceRecord structure returned in the ASEnumerateSources() reply is used to indicate the type of information source, as follows:

```
const     Byte kASInfoSourceTypeNormal  = 0;
const     Byte kASInfoSourceTypeWAIS    = 1;
```

The low 24 bits are used to indicate the capabilities of the information source. Of the 24 bits, the lowest three bits are currently used:

```
Bit 0: canCopyOriginal
Bit 1: canDoMatchTerms
Bit 2: canDoCOW
```

Some information sources may be filtered out by the server if the user doesn't have the appropriate access privileges.

Open Search Session

The Open Search Session Apple event opens a search session and must be issued before a search request can be initiated. Your application can open as many search sessions as needed, and there can be any number of searches within each search session.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASOpenSearchSession (ossn)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`searchSessionID`
Type: long
Size: 4
Keyword: `keyASSearchSessionID (seid)`
Comment: use this when making a search request, a `GetArticleText` request, or a `GetArticleMatches` request

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Close Search Session

The Close Search Session Apple event closes a search session when your application is done with it.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASCloseSearchSession (cssn)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`searchSessionID`
Type: long
Size: 4
Keyword: `keyASSearchSessionID (seid)`
Comment: ID of search session to be closed

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidSearchSessionIDErr`	1028	Search session ID passed was invalid.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Search

The Search Apple event initiates an immediate search of the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASSearch (srch)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`searchSessionID`
Type: long
Size: 4
Keyword: `keyASSearchSessionID (seid)`
Comment: ID from Open Search Session request

`queryString`
Type: char
Size: variable
Keyword: `keyASQueryString (qrey)`
Comment: search text to be submitted to the server

`searchType`
Type: long
Size: 4
Keyword: `keyASSearchType (srtp)`
Comment: not used in this version, but reserved for future use

`minimumRank`
Type: long
Size: 4
Keyword: `keyASMinimumRank (mino)`
Comment: must be between 1 and 5 inclusive

`maximumHits`
Type: long
Size: 4
Keyword: `keyASMaximumHits (maxo)`
Comment: must be between 1 and 30,000 inclusive

`sourceID`
Type: long
Size: 4
Keyword: `keyASSourceID (isid)`
Comment: ID of a source to be searched

`earliestModDate`
Type: long
Size: 4
Keyword: `keyASEarliestModDate (dato)`
Comment: the earliest modification date for articles to be returned; this number is the difference in seconds between the current date and 12:00 midnight, January 1, 2000

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter:

`error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`searchID`
Type: long
Size: 4
Keyword: `keyASSearchID (srid)`
Comment: transaction ID for this search

`numberOfHits`
Type: long
Size: 4
Keyword: `keyASNumberOfHits (thit)`
Comment: number of articles returned

`hitList`
Type: list
Size: variable
Keyword: `keyASHitList (hlst)`
Comment: list of `HitDataRecord` structures

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidSourceIDErr`	1006	Information source ID passed was invalid.
`kInvalidArticleIDErr`	1007	Article ID passed was invalid.
`kQueryMissingStringErr`	1017	Query length was specified but the string pointer was empty or length was 0.
`kQueryLengthRangeErr`	1018	Query length is out of range.
`kMinRankRangeErr`	1019	Minimum rank search option value is out of range.
`kMaxHitsRangeErr`	1020	Maximum hits search option value is out of range.
`kInvalidSearchSessionIDErr`	1028	Search session ID passed was invalid.
`kQuerySyntaxErr`	1032	Syntax error in query string.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Notes

Each element in the `hitList` parameter in the final reply is a block of bytes that can be cast to the `HitDataRecord` structure, as follows:

```
struct ASHitInfo {
     long      HitArticleID;
     long      HitModDate;
     long      hitSize;
     long      hitRank;
     Str31     hitTitle
};
```

Get Related Terms

The Get Related Terms Apple event obtains a list of co-occurring terms for a specific term from the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetRelatedTerms (gcow)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`sourceID`
Type: long
Size: 4
Keyword: `keyASSourceID (isid)`
Comment: ID of a source to be searched

`queryString`
Type: char
Size: variable
Keyword: `keyASQueryString (qrey)`
Comment: the original term to expand

`maximumTerms`
Type: long
Size: 4
Keyword: `keyASMaxTerms (mxtm)`
Comment: maximum number of terms to return

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`numberOfTerms`
Type: long
Size: 4
Keyword: `keyASNumberOfTerms (numt)`
Comment: number of terms returned

`termList`
Type: list
Size: variable
Keyword: `keyASTermList (tlst)`
Comment: list of terms

Result Codes

`noErr` 0 The request was completed successfully.

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kQueryMissingStringErr`
1017 Query length was specified but the string pointer was empty or length was 0.

`kQueryLengthRangeErr`
1018 Query length is out of range.

`kMaxTermsRangeErr`
1021 Value specified in `maximumTerms` field is out of range.

`kQuerySyntaxErr`
1032 Syntax error in query string.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Notes

Each element in the `hitList` parameter in the final reply is a Pascal string.

Get Article Text

The Get Article Text Apple event retrieves the text of an article whose ID and source ID are specified. Your application can retrieve a portion of the text by specifying its start offset and length; however, the length cannot exceed 30,000 bytes.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetArticleText (gatx)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`sourceID`
Type: long
Size: 4
Keyword: `keyASSourceID (isid)`
Comment: ID of source to which this article belongs

`searchSessionID`
Type: long
Size: 4
Keyword: `keyASSearchSessionID (seid)`
Comment: ID of search session in which this article was found

`articleID`
Type: long
Size: 4
Keyword: `keyASArticleID (arid)`
Comment: ID of article to be returned

`startOffset`
Type: long
Size: 4
Keyword: `keyASStartOffset (ofst)`
Comment: byte offset from which to start

`textLength`
Type: long
Size: 4
Keyword: `keyASMaxLength (mlen)`
Comment: number of bytes to be returned (must be between 1 and 30,000 inclusive); if you need more than 30,000 bytes, you must make another request

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter:

`error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`articleLength`
Type: long
Size: 4
Keyword: `keyASArticleLength (atln)`
Comment: number of bytes returned

`articleText`
Type: char
Size: variable
Keyword: `keyASArticleText (atxt)`
Comment: text returned

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidSourceIDErr`	1006	Information source ID passed was invalid.
`kInvalidArticleIDErr`	1007	Article ID passed was invalid.
`kInvalidTextBoundErr`	1022	Start offset value or text length specified is out of range.

`kFileModSinceIndexErr`
1033 The information requested could not be retrieved because the file has been modified since the last index time.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Get Article Info

The Get Article Info Apple event obtains an article's modification date, title, and size.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetArticleInfo (gair)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`infosourceID`
Type: long
Size: 4
Keyword: `keyASSourceID (isid)`
Comment: ID of source to which this article belongs

`articleID`
Type: long
Size: 4
Keyword: `keyASArticleID (arid)`
Comment: ID of article desired

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`modDate`
Type: long
Size: 4
Keyword: `keyASEarliestModDate (dato)`
Comment: the modification date of the article

`title`
Type: char
Size: variable
Keyword: `keyASTitle (ttle)`
Comment: article title

`articleLengt`
Type: long
Size: 4
Keyword: `keyASArticleLength (atln)`
Comment: size of the article in bytes

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidSourceIDErr`	1006	Information source ID passed was invalid.
`kInvalidArticleIDErr`	1007	Article ID passed was invalid.
`kFileModSinceIndexErr`	1033	The information requested could not be retrieved because the file has been modified since the last index time.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Get Article Matches

The Get Article Matches Apple event retrieves match terms for an article whose ID and source ID are specified. (This retrieval does not happen in the case of WAIS searches.)

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetArticleMatches (gamt)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`infosourceID`
Type: long
Size: 4
Keyword: `keyASSourceID (isid)`
Comment: ID of source to which this article belongs

`searchSessionID`
Type: long
Size: 4
Keyword: `keyASSearchSessionID (seid)`
Comment: ID of the search session from which this article ID was returned; ID obtained from the Open Search Session call

`articleID`
Type: long
Size: 4
Keyword: `keyASArticleID (arid)`
Comment: ID of the article from which to retrieve matches

`startOffset`
Type: long
Size: 4
Keyword: `keyASStartOffset (ofst)`
Comment: number of bytes from the beginning of the article from which you want to get matches

maximumLength
Type: long
Size: 4
Keyword: keyASMaxLength (mlen)
Comment: must be less than or equal to 30,000; if you want more, you need to make another request

Reply

Event Class: kCoreEventClass (aevt)

Event ID: kAEAnswer (ansr)

Parameter: error
Type: short
Size: 2
Keyword: keyErrorNumber (errn)
Comment: request was unsuccessful if not 0

userRefCon
Type: long
Size: 4
Keyword: keyASUserRefCon (rfcn)
Comment: copied from request; if missing from the request, returned as 0

matchTerms
Type: list
Size: variable
Keyword: keyASTermList (tlst)
Comment: list of 8-byte offset/length pair

Result Codes

noErr	0	The request was completed successfully.
kInvalidSourceIDErr	1006	Information source ID passed was invalid.
kInvalidArticleIDErr	1007	Article ID passed was invalid.
kInvalidTextBoundErr	1022	Start offset value or text length specified is out of range.
kInvalidSearchSessionIDErr	1028	Search session ID passed was invalid.
kFileModSinceIndexErr	1033	The information requested could not be retrieved because the file has been modified since the last index time.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Notes

Each element in the `matchTerms` parameter in the final reply is a pair of long integers. The first is the offset from the beginning of the article where the matched term starts, and the second is the length of that term in bytes.

Add Reporter

The Add Reporter Apple event adds a new reporter to the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASAddReporter (arpt)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`reporterName`
Type: char
Size: variable
Keyword: `keyASReporterName (rpnm)`
Comment: name must be 20 bytes or less

`queryString`
Type: char
Size: variable
Keyword: `keyASQueryString (qrey)`
Comment: search query

`minimumRank`
Type: long
Size: 4
Keyword: `keyASMinimumRank (mino)`
Comment: must be between 1 and 5 inclusive

`maximumHits`
Type: long
Size: 4
Keyword: `keyASMaximumHits (maxo)`
Comment: must be between 1 and 30,000 inclusive

`earliestModDate`
Type: long
Size: 4
Keyword: `keyASEarliestModDate (dato)`
Comment: the earliest modification date for articles to be returned; this number is the difference in seconds between the current date and 12:00 midnight, January 1, 2000

`earliestIndexDate`
Type: long
Size: 4
Keyword: `keyASEarliestIndexDate (idto)`
Comment: pass 0 in this field

`deliveryDir`
Type: alias
Size: variable
Keyword: `keyASDeliveryDir (dldd)`
Comment: alias record of a directory to which scheduled search results should be delivered

`deliveryDays`
Type: long
Size: 4
Keyword: `keyASDeliveryDays (dldy)`
Comment: days of the week to deliver scheduled search result

`deliveryTime`
Type: long
Size: 4
Keyword: `keyASDeliveryTime (dldt)`
Comment: scheduled search delivery time

`deliveryFlags`
Type: long
Size: 4
Keyword: `keyASDeliveryFlags (dlfl)`
Comment: set bit 0 to 1 if this scheduled search is active

`infoSourcesIDList`
Type: list
Size: variable
Keyword: `keyASSourceIDList (isil)`
Comment: list of source IDs

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of the reporter that was added

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidSourceIDErr`	1006	Information source ID passed was invalid.
`kDuplicateObjectNameErr`	1014	Reporter by that name already exists.
`kQueryMissingStringErr`	1017	Query length was specified but the string pointer was empty or length was 0.
`kQueryLengthRangeErr`	1018	Query length is out of range.
`kMinRankRangeErr`	1019	Minimum rank search option value is out of range.
`kMaxHitsRangeErr`	1020	Maximum hits search option value is out of range.

`kInvalidObjectNameErr`
1025 Reporter name is invalid or missing.

`kInvalidObjectDataErr`
1026 Object data or data length is invalid.

`kInvalidFlagErr`
1027 `deliveryFlags` has an invalid flag set.

`kInvalidSearchSessionIDErr`
1028 Search session ID passed was invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kInvalidDeliveryDaysErr`
3407 Invalid delivery days were specified.

`kInvalidDeliveryTimeErr`
3408 Invalid delivery time was specified.

Notes

The `deliveryDays` field specifies the days of the week on which the scheduled search result should be delivered. It is a long integer, but only the lowest 7 bits are used. Bit assignments are as follows:

bit 0 = Sunday

bit 1 = Monday

bit 2 = Tuesday

bit 3 = Wednesday

bit 4 = Thursday

bit 5 = Friday

bit 6 = Saturday

To specify every day of the week, set all of the bits ($007F).

The `deliveryTime` field specifies what time the scheduled search should be delivered. The time is expressed in the number of seconds since midnight. For example, if the delivery time is 1:10:00 A.M., the value of the field should be 4200:

(1 x 60 x 60) + (10 x 60) = 4200

All other bits in the `deliveryTime` field are reserved for future use by Apple.

Modify Reporter

The Modify Reporter Apple event modifies an existing reporter on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASModifyReporter (mrpt)`

Parameter:

`registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of reporter to be modified

`reporterName`
Type: char
Size: variable
Keyword: `keyASReporterName (rpnm)`
Comment: name must be 20 bytes or less

`queryString`
Type: char
Size: variable
Keyword: `keyASQueryString (qrey)`
Comment: search query

`minimumRank`
Type: long
Size: 4
Keyword: `keyASMinimumRank (mino)`
Comment: must be between 1 and 5 inclusive

`maximumHits`
Type: long
Size: 4
Keyword: `keyASMaximumHits (maxo)`
Comment: must be between 1 and 30,000 inclusive

`earliestModDate`
Type: long
Size: 4
Keyword: `keyASEarliestModDate (dato)`
Comment: the earliest modification date for articles to be returned; this number is the difference in seconds between the current date and 12:00 midnight, January 1, 2000

`earliestIndexDate`
Type: long
Size: 4
Keyword: `keyASEarliestIndexDate (idto)`
Comment: pass 0 in this field

`deliveryDir`
Type: alias
Size: variable
Keyword: `keyASDeliveryDir (dldd)`
Comment: alias record of a directory to which scheduled search results should be delivered

`deliveryDays`
Type: long
Size: 4
Keyword: `keyASDeliveryDays (dldy)`
Comment: days of the week on which to deliver scheduled search result

`deliveryTime`
Type: long
Size: 4
Keyword: `keyASDeliveryDays (dldy)`
Comment: scheduled search delivery time

`deliveryFlags`
Type: long
Size: 4
Keyword: `keyASDeliveryFlags (dlfl)`
Comment: set bit 0 to 1 if this scheduled search is active

`infoSourcesIDList`
Type: list
Size: variable
Keyword: `keyASSourceIDList (isil)`
Comment: list of source IDs

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of the reporter that was modified

Result Codes

`noErr` 0 The request was completed successfully.

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kDuplicateObjectNameErr`
1014 Reporter by that name already exists.

`kQueryMissingStringErr`
1017 Query length was specified but the string pointer was empty or length was 0.

`kQueryLengthRangeErr`
1018 Query length is out of range.

`kMinRankRangeErr`
1019 Minimum rank search option value is out of range.

`kMaxHitsRangeErr`
1020 Maximum hits search option value is out of range.

`kInvalidObjectNameErr`
1025 Reporter name is invalid or missing.

`kInvalidObjectDataErr`
1026 Object data or data length is invalid.

`kInvalidFlagErr`
1027 `deliveryFlags` has an invalid flag set.

`kInvalidSearchSessionIDErr`
1028 Search session ID passed was invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kDeliveryDirLengthErr`
3405 Delivery directory length is out of range.

`kInvalidDeliveryDaysErr`
3407 Invalid delivery days were specified.

`kInvalidDeliveryTimeErr`
3408 Invalid delivery time was specified.

Notes

The `deliveryDays` field specifies the days of the week on which the scheduled search result should be delivered. It is a long integer, but only the lowest 7 bits are used. Bit assignments are as follows:

bit 0 = Sunday

bit 1 = Monday

bit 2 = Tuesday

bit 3 = Wednesday

bit 4 = Thursday

bit 5 = Friday

bit 6 = Saturday

To deliver every day of the week, set all of the bits ($007F).

The `deliveryTime` field specifies the time on which the scheduled search should be delivered. The time is expressed in number of seconds since midnight. For example, if the delivery time is 1:10:00 a.m., it should be 4200:

(1 x 60 x 60) + (10 x 60) = 4200

All other bits in the `deliveryTime` field are reserved for future use by Apple.

Get Reporter

The Get Reporter Apple event returns information about a reporter whose ID is specified. The reporter must be on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetReporter (grpt)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of reporter desired

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of reporter desired

`reporterName`
Type: char
Size: variable
Keyword: `keyASReporterName (rpnm)`
Comment: name of the reporter

`queryString`
Type: char
Size: variable
Keyword: `keyASQueryString (qrey)`
Comment: search query

`minimumRank`
Type: long
Size: 4
Keyword: `keyASMinimumRank (mino)`
Comment: minimum rank

`maximumHits`
Type: long
Size: 4
Keyword: `keyASMaximumHits (maxo)`
Comment: maximum number of hits to find

`earliestModDate`
Type: long
Size: 4
Keyword: `keyASEarliestModDate (dato)`
Comment: the earliest modification date for articles to be found

`earliestIndexDate`
Type: long
Size: 4
Keyword: `keyASEarliestIndexDate (idto)`
Comment: pass 0 in this field

`deliveryDir`
Type: alias
Size: variable
Keyword: `keyASDeliveryDir (dldd)`
Comment: alias record of a directory to which scheduled search results should be delivered

`deliveryDays`
Type: long
Size: 4
Keyword: `keyASDeliveryDays (dldy)`
Comment: days of the week on which the scheduled search results are delivered

`deliveryTime`
Type: long
Size: 4
Keyword: `keyASDeliveryDays (dldy)`
Comment: time of scheduled search delivery

`deliveryFlags`
Type: long
Size: 4
Keyword: `keyASDeliveryFlags (dlfl)`
Comment: scheduled search is active if bit 0 is set

`infoSourcesIDList`
Type: list
Size: variable
Keyword: `keyASSourceIDList (isil)`
Comment: list of source IDs

Result Codes

`noErr` 0 The request was completed successfully.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Delete Reporter

The Delete Reporter Apple event deletes a reporter whose ID is specified. The reporter must be on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASDeleteReporter (drpt)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of the reporter to be deleted

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`reporterID`
Type: long
Size: 4
Keyword: `keyASReporterID (rpid)`
Comment: ID of the reporter that was deleted

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidObjectIDErr`	1010	Reporter ID passed was invalid.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Enumerate Reporters

The Enumerate Reporters Apple event lists all reporters that are on the currently connected server and that belong to the user.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASEnumerateReporters (erpt)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`reporterList`
Type: list
Size: variable
Keyword: `keyASReporterList (ridl)`
Comment: list of the `ReporterInfoRecord` structure

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Notes

Each element in the `reporterList` parameter in the final reply is a block of bytes that can be cast to the `ReporterInfoRecord` structure, which is shown here. The first bit of the `repFlags` field is set if the scheduled search is active.

```
struct ReporterInfoRecord {
     long      repID;
     Str31     repName;
     long      repFlags;
};
```

Add DB Object

The Add DB Object Apple event adds a generic object containing any custom data to the currently connected server's object database.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASAddDBObject (adbo)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the object

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object; must be a positive integer and unique per type

`objectName`
Type: char
Size: variable
Keyword: `keyASDBObjectName (onam)`
Comment: name must be 31 bytes or less

`objectFlags`
Type: long
Size: 4
Keyword: `keyASDBObjectFlags (obfl)`
Comment: object flags; can be used for any purpose

`objectData`
Type: wildcard
Size: variable
Keyword: `keyASDBObjectData (odat)`
Comment: object data; must be less than 30,000 bytes

Final Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter:

`error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidObjectNameErr`	1025	Reporter name is invalid or missing.
`kInvalidObjectDataErr`	1026	Object data or data length is invalid.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Modify DB Object

The Modify DB Object Apple event modifies a database object on the currently connected server whose type and ID is specified.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASModifyDBObject (mdbo)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the object

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object to be modified

`objectName`
Type: char
Size: variable
Keyword: `keyASDBObjectName (onam)`
Comment: name must be 31 bytes or less

`objectFlags`
Type: long
Size: 4
Keyword: `keyASDBObjectFlags (obfl)`
Comment: object flags

`objectData`
Type: wildcard
Size: variable
Keyword: `keyASDBObjectData (odat)`
Comment: object data; must be less than 30,000 bytes

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object that was modified

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidObjectNameErr`	1025	Reporter name is invalid or missing.
`kInvalidObjectDataErr`	1026	Object data or data length is invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Get DB Object

The Get DB Object Apple event returns information about a specific database object on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetDBObject (gdbo)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the object

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object desired

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the object

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object

`objectName`
Type: char
Size: variable
Keyword: `keyASDBObjectName (onam)`
Comment: name must be 31 bytes or less

`objectFlags`
Type: long
Size: 4
Keyword: `keyASDBObjectFlags (obfl)`
Comment: object flags

`objectData`
Type: wildcard
Size: variable
Keyword: `keyASDBObjectData (odat)`
Comment: object data; must be less than 30,000 bytes

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidObjectIDErr`	1010	Reporter ID passed was invalid.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Delete DB Object

The Delete DB Object Apple event deletes a database object on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASDeleteDBObject (ddbo)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the object

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object to be deleted

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`objectID`
Type: long
Size: 4
Keyword: `keyASDBObjectID (obid)`
Comment: ID of the object that was deleted

Result Codes

`noErr` 0 The request was completed successfully.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Enumerate DB Objects

The Enumerate DB Objects Apple event lists all database objects of a specific type on the currently connected server.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASEnumerateDBObjects (edbo)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the objects to be enumerated

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`objectType`
Type: long
Size: 4
Keyword: `keyASDBObjectType (otyp)`
Comment: type of the objects

`objectList`
Type: list
Size: variable
Keyword: `obls`
Comment: list of `DBObjectRecord` structures

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

Notes

Each element in the `objectList` parameter in the final reply is a block of bytes that can be cast to the `DBObjectRecord` structure, which is shown here.

```
struct DBObjectRecord {
     long      objID;
     Str31     objName;
     long      objFlags;
};
```

Import Reporter

The Import Reporter Apple event adds a new reporter to the currently connected server from a reporter file. A new reporter ID is assigned to the reporter and is returned in the final reply.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASImportReporter (irpt)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`sourcefile`
Type: FSSpec
Size: 70
Keyword: `keyASFileSpec (fspc)`
Comment: reporter file from which the reporter should be imported

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

reporterID
Type: long
Size: 4
Keyword: keyASReporterID (rpid)
Comment: ID of the imported reporter

reporterName
Type: char
Size: variable
Keyword: keyASReporterName (rpnm)
Comment: name of the reporter

Result Codes

noErr	0	The request was completed successfully.
kInvalidRegistrationIDErr	3102	Invalid registration ID was passed.
kNotLoggedOnErr	3106	Client is currently not logged on to any server.

Export Reporter

The Export Reporter Apple event exports a specific reporter to a reporter file.

Request

Event Class: kAppleSearchRequest (bgrt)

Event ID: kAEASExportReporter (xrpt)

Parameter: registrationID
Type: long
Size: 4
Keyword: keyASRegistrationID (btid)
Comment: registration ID

userRefCon
Type: long
Size: 4
Keyword: keyASUserRefCon (rfcn)
Comment: for the requester's own use; optional

reporterID
Type: long
Size: 4
Keyword: keyASReporterID (rpid)
Comment: ID of the reporter to be exported

`targetFile`
Type: FSSpec
Size: 70
Keyword: `keyASFileSpec (fspc)`
Comment: specification of target reporter file

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter: `error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: an error if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

Result Codes

`noErr` 0 The request was completed successfully.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Get Original File

The Get Original File Apple event copies the original file of an article or its alias from the server to the local disk.

Request

Event Class: `kAppleSearchRequest (bgrt)`

Event ID: `kAEASGetFile (cpfi)`

Parameter: `registrationID`
Type: long
Size: 4
Keyword: `keyASRegistrationID (btid)`
Comment: registration ID

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: for the requester's own use; optional

`infoSourceID`
Type: long
Size: 4
Keyword: `keyASSourceID (isid)`
Comment: ID of the source to which this file belongs

`articleID`
Type: long
Size: 4
Keyword: `keyASArticleID (arid)`
Comment: ID of the article whose original is to be retrieved

`flags`
Type: long
Size: 4
Keyword: `keyASGetOriginalFileFlags (gofl)`
Comment: set bit 0 if original is wanted, or set bit 1 if alias is wanted

`destination`
Type: FSSpec
Size: 70
Keyword: `keyASFileSpec (fspc)`
Comment: needed only if flag bit 0 is set

Reply

Event Class: `kCoreEventClass (aevt)`

Event ID: `kAEAnswer (ansr)`

Parameter:

`error`
Type: short
Size: 2
Keyword: `keyErrorNumber (errn)`
Comment: request was unsuccessful if not 0

`userRefCon`
Type: long
Size: 4
Keyword: `keyASUserRefCon (rfcn)`
Comment: copied from request; if missing from the request, returned as 0

`alias`
Type: aliasSize: variable
Keyword: `keyASAliasRecord (oals)`
Comment: alias to original; returned only if flag bit 1 is set

Result Codes

`noErr`	0	The request was completed successfully.
`kInvalidSourceIDErr`	1006	Information source ID passed was invalid.
`kInvalidArticleIDErr`	1007	Article ID passed was invalid.
`kInvalidFlagErr`	1027	`keyAsGetOriginalFileFlags` has an invalid flag set.
`kInvalidRegistrationIDErr`	3102	Invalid registration ID was passed.
`kNotLoggedOnErr`	3106	Client is currently not logged on to any server.

3 C Library Application Programming Interface

This chapter describes the C Library application programming interface (API) function calls provided by AppleSearch. It describes how to use the API and then gives a detailed specification for each C function call.

IMPORTANT The AppleSearch Client Library communicates with the AppleSearch server using the Program-to-Program Communications (PPC) protocol. In future releases of AppleSearch, another protocol may be used as a client/server communication mechanism. If another protocol is used, you will need to rebuild your software with the new client library. This is one reason you may want to create your client application by using Apple events.

Making a C Library API function call

The AppleSearch API function calls are normal C functions that you can link to and call from your application. Most of the function calls must be made asynchronously, and they all have a similar calling convention. They take the following two parameters:

- A pointer to a parameter block—each parameter block differs from function to function, but the first four fields, called `ASParamBlockHeader`, are the same in all parameters.
- A Boolean parameter—should always pass `true`.

Since they are always asynchronous, the functions return values, of type `ASErr`, immediately to your calling code before sending the request to the server. If this error code is nonzero, it usually means that your application passed one or more invalid values in the parameter block, and the function doesn't perform any further operations. If the error code is 0, the function performs the task—usually sending a request to the server—and calls your completion routine when a reply from the server is returned.

Every AppleSearch API parameter block uses the following parameter block header:

```
#define ASParamBlockHeader
     long           userRefCon;
     ASErr          error;
     ASProcPtr      completionProc;
     long           registrationID;
```

- The `userRefCon` field is for your own use. It remains unchanged during and after the execution of the call.
- The `error` field returns the error code from the execution. If the function was executed successfully, it should contain 0. Otherwise, it contains either an AppleSearch-specific or a Macintosh Toolbox or Macintosh Operating System error code. AppleSearch-specific errors for a particular call are listed in each call section.
- If you want to be notified when the execution is complete (which is almost always the case), you should pass an address of a function to the `completionProc` field.
- You must specify your registration ID in the `registrationID` field. You obtain your registration ID by calling the ASRegister function before making any other calls (except ASInitialize).

Since the AppleSearch Client Library extracts all necessary information from your parameter block while preparing to process your request, you are free to dispose of the parameter block once you make an API call, even before the call is completed and your completion routine is called.

Completion routines

Your completion routine is called by the AppleSearch Client Library with two parameters. The first parameter is of type `char*`, which is actually a parameter block pointer. You should cast it to an appropriate parameter block pointer. For example, a pointer of type `char*` passed to a completion routine called from ASAddReporter should be cast to type ASReporterPBPtr. This parameter block is a copy of the parameter block originally supplied by your code.

Note: Do not dispose of this pointer. It is disposed of automatically by the AppleSearch Client Library after your completion routine is executed.

The second parameter, of type `long`, is the identifier value that was passed in the ASRegister function call.

The syntax of the completion routine is as follows:

```
ASErr MyCompletionProc(char* pb, long identifier);
```

Note: Completion routines have to be universal proc pointers. This means that they have to be instantiated (or created) using the NewASUPP macro defined in the ASClient.h header file. For more information, see *Making the Leap to Power PC* (“Mixed Mode Manager,” “Universal Interfaces,” and “Universal ProcPtrs”), in the Macintosh On-RISC Software Development Kit.

Errors

Each function can return a number of different error codes. There are two places where the caller should check for errors. First, all of the API functions return a synchronous error code. If the error code is not 0, an error occurred while preparing to execute the function, and the asynchronous portion will not be executed. For example, if you pass an invalid registration ID to the ASSearch function call, ASSearch catches it during its parameter-checking and returns an appropriate error code. In this case, a query to the server is not made, and therefore the completion routine will not be called.

The second type of error is returned by the server. Errors of this type are returned in the `error` field of the parameter block, which your software must check in its completion routine. For example, making an ASGetReporter function call with a reporter ID that doesn’t exist causes the server to return an appropriate error code to the completion routine in the `error` field of the parameter block.

Synchronous errors—those generated by the client code before accessing the server—are in the range 3,000 through 3,999, and asynchronous errors returned by the server are in the range 1,000 through 1,999. You may also get errors in the range 2,000 through 2,999 from the PPC library, which provides communications between the client and server. In addition, Macintosh Toolbox or Macintosh Operating System errors, such as out-of-memory and file-not-found errors, may be returned in either place. These Macintosh Toolbox and Operating System errors are always identified by a negative number.

Initializing the AppleSearch Client Library and connecting to a server

Before making any other function calls, your application must initialize the AppleSearch Client Library by calling ASInitialize. You then need to register your application to the library by calling ASRegister. You should pass 0 to the identifier field of the ASRegister parameter block.

Your application must then make a connection to an AppleSearch server. You can make the connection by calling either ASLogon with the `autoLogon` flag set or ASSelectServer followed by ASLogon. The ASLogon call with the `autoLogon` flag set automatically causes the PPCBrowser dialog box to appear, which prompts the user to select a server. The server must then authenticate that user. If the log-on operation is successful, the call returns the name of the server and a flag indicating whether the user logged on as a guest or registered user. Error codes are returned only if the call is unsuccessful.

Building your application

When building your application, you must include three header files—`ASClient.h`, `AppleSearchErrors.h`, and `AppleSearchAPI.h`—and link `ASClientLib.o`. You must also use the MPW `rez` command to compile `ASClient.r` into your application. The following makefile example illustrates how your application might be built:

```
#  File: MyApplication.make
#
#  Contains: A sample makefile for a custom AppleSearch
#  client application.
#
#  Note: Be sure to include all appropriate header files in your
#  source; place the AppleSearchClientLib.o object file in your
#  object directory and place the AppleSearchClientLib.r file
#  in your build directory
```

```
#  ----Variables----

AppName=MyApplication
AppCreator= 'myap'
AppType= 'APPL'

SrcDir= :Sources:
HdrDir= :Headers:
ObjDir= :Objects:

SymOption= full
MacsBugOption= full

CompileOptions= -mf -sym {SymOption} -mbg {MacsBugOption} -i
“{HdrDir}”
RezOptions= -i “{HdrDir}”
LinkOptions= -mf -sym {SymOption}

xC = {C} {CompileOptions}
xCPlus = CPlus {CompileOptions}

Libs= “{Libraries}”Runtime.o ∂
      “{Libraries}”Interface.o

CLibs= “{CLibraries}”CPlusLib.o ∂
      “{CLibraries}”StdCLib.o

OtherLibs= “{ObjDir}”ASClientLib.o

#  ----C++----

# Main application components

“{ObjDir}”MyApplication.cp.o ƒ “{SrcDir}”MyApplication.cp ∂
      “{HdrDir}”MyApplication.h
CPlus “{SrcDir}”MyApplication.cp  -o “{ObjDir}” {CompileOptions}

#  ----Objects List----

OBJFiles= “{ObjDir}”MyApplication.cp.o

#  ----Link & Rez----

“{AppName}” ƒ     {OBJFiles} {OtherLibs} {Libs} {CLibs} “{AppName}”.r
      Rez {RezOptions} “{AppName}”.r “ASClient.r” -a -o “{AppName}”
      Link -t {AppType} -c {AppCreator} -o “{AppName}” {LinkOptions}
∂
      {OBJFiles}∂
      {OtherLibs}∂
      {Libs} {PLibs} {CLibs}
```

C Library API calls

This section specifies in detail each C Library API function call.

ASInitialize

The ASInitialize function call initializes internal globals and creates several objects used internally. Your application's initialization code must call this function once. It should be called before any other API function calls are made.

Syntax

```
void ASInitialize ( void );
```

ASQuit

The ASQuit function call must be called by your application right before it quits. This function call must be called after your application has logged off the server and has unregistered itself from the AppleSearch Client Library. The call closes the open PPC ports and performs other cleanup tasks.

Syntax

```
void ASQuit ( void );
```

ASDoIdle

The ASDoIdle function call sends requests in the queue, checks server responses, and checks for the update delivery time. Your application should call this function from the main event loop. Any function that performs a lengthy operation should also call it.

Syntax

```
ASErr ASDoIdle ( void );
```

ASRegister

The ASRegister function call registers your application to the library before making any other call (except ASInitialize). The identifier parameter must be 0.

Syntax

```
ASErr ASRegister ( ASRegisterPBPtr pb );

struct ASRegisterPB {
     ASParamBlockHeader
     long      identifier;
};
typedef ASRegisterPB ASRegisterPB , *ASRegisterPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	This field is not used by ASRegister since it is a synchronous call.
←	`registrationID`	A registration ID is returned.
→	`identifier`	You must pass 0 in this field.

Errors Returned

`kMaxRegisteredErr` 3101	There is already a maximum number of applications registered to the Library.

Notes

If the update generation code is present, it registers itself to the library with an identifier of –1. Other than that, only one registration is allowed.

ASUnregister

The ASUnregister function call unregisters your application when it no longer needs library services. Call ASLogoff to be sure that your application is already disconnected from the server.

Syntax

```
ASErr ASUnregister ( ASRegisterPBPtr pb );

struct ASRegisterPB {
     ASParamBlockHeader
     long      identifier;
};
typedef ASRegisterPB ASRegisterPB , *ASRegisterPB Ptr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc` This field is not used by ASUnregister since it is a synchronous call.

→ `registrationID` Pass your registration ID.

→ `identifier` This field is not used by ASUnregister.

Errors Returned

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

ASSelectServer

The ASSelectServer function call displays the PPCBrowser dialog box, which allows the user to select a server to connect to. This function call can be used only if you are using the AppleSearch Client Library as the foreground application. The information passed back by this function can be used in the ASLogon parameter block to make a connection to a server.

Syntax

```
ASErr ASSelectServer ( ASSelectServerPBPtr pb );

struct ASSelectServerPB {
     ASParamBlockHeader
     Ptr            defaultServer;
     unsigned long  dataSize;
     Ptr            dataPtr;
};
typedef ASSelectServerPB ASSelectServerPB,
*ASSelectServerPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
This field is not used by ASSelectServer since it is a synchronous call.

→ `registrationID`
A registration ID.

→ `defaultServer`
Not currently used; should be set to nil.

← `dataSize` If no error is returned, this field contains the size of the
data pointed to by `dataPtr`.

← `dataPtr` If no error is returned, this field points to a data buffer created by the function; this field and the `dataSize` field should be passed to the ASLogon call.

Errors Returned

`kMaxRegisteredErr`
3101 There is already a maximum number of applications registered to the Library.

Notes

Since this call displays a dialog box, it will fail if the application that makes the call is a background-only application.

The memory allocated for `dataPtr` by the function is disposed of by the ASLogon call. If your application is not passing the `dataPtr` to ASLogon, you must dispose `dataPtr` by calling `DisposePtr`.

ASLogon

The ASLogon function call can be used in one of two ways:

- Your application can make the call with the `autoLogon` flag set, which causes the function to automatically display the PPCBrowser dialog box and authenticate the user. The AppleSearch Client Library uses the AppleSearch Authentication Extension to handle the user interaction.
- Your application can call the ASSelectServer function call, which displays the PPCBrowser dialog box, allowing the user to select a server to connect to. The data size and data pointer returned by ASSelectServer must be passed by your application into the ASLogon function call with the `autoLogon` flag cleared. Since the function displays a dialog box, you can use this method only if your application is running in the foreground.

Syntax

```
ASErr ASLogon ( ASLoginPBPtr pb );
```

```
struct ASLogonPB {
     ASParamBlockHeader
     unsigned long  flags;
     Ptr            dataPtr;
     unsigned long  defaultUserNameLength;
     char*          defaultUserNameString;
     unsigned long  serverNameLength;
     char*          serverNameString
};
typedef ASLogonPB ASLogonPB, *ASLogonPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to your completion routine.

→ `registrationID`
Your application's registration ID.

↔ `flags` Set bit 0 to set `autoLogon`; set bit 1 if you want updates to be delivered automatically; bit 2 gets set by the function if the user logged on to the server as a guest.

→ `dataSize` If not `autoLogon`, then the `dataSize` field from the ASSelectServer parameter block must be passed in this field.

→ `dataPtr` If not `autoLogon`, then the `dataPtr` field from the ASSelectServer parameter block must be passed in this field.

→ `defaultUseNameLength`
If not `autoLogon` and you want to specify the default log-on name in the authentication dialog box, pass the length of the name in this field. If it is 0, then the owner name from the Sharing Setup dialog box is used.

→ `defaultUserNameString`
If not `autoLogon` and you want to specify the default log-on name in the authentication dialog box, pass a string in this field.

← `serverNameLength`
The length of the server name.

← `serverNameString`
The length of the server to which the user was connected.

Errors Returned

`kTransactionFailed`
1005 Server search engine transaction failed.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kWrongProtocol`
1012 Client is using the wrong protocol version.

`kTooManyUsers`
1015 The maximum number of users are already logged on.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kAlreadyLoggedOnErr`
3103 The client is already logged on to a server.

`kLogOnCancelErr`
3104 The user canceled the log-on operation in the PPCBrowser or authentication dialog box.

Notes

If the `autoLogon` flag is set, this call launches the AppleSearch Authentication Extension, which displays the PPCBrowser dialog box, starts a PPC session, and sends the session information back to your application by means of Apple events. Since it uses Apple events, your application's main event loops must handle high-level events. You should have code similar to the following example in your source code:

```
HandleEvent(EventRecord& event)
{
     OSErr err;
     switch(event.what) {
          case kHighLevelEvent:
               err=AEProcessAppleEvent(&event);
               break
          case mouseDown:
          .
          .
          .
     }
}
```

The Apple event handler for the event sent back by the AppleSearch Authentication Extension is installed by the AppleSearch Client Library when your application calls ASInitialize.

An alternative way of logging on to a server without using the AppleSearch Authentication Extension, and therefore any Apple events, is to call ASSelectServer to select a server and then call ASLogon with values returned by ASSelectServer to make a connection. This alternative method can be used only if your application is running in the foreground (because it displays a dialog box).

ASLogoff

The ASLogoff function call logs your application off the currently connected server.

Syntax

```
ASErr ASLogoff ( ASLogoffPBPtr pb );

struct ASLogoffPB {
     ASParamBlockHeader
     Boolean   forceLogoff;
};
typedef ASLogoffPB ASLogoffPB , *ASLogoffPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
This field is not used for ASLogoff.

→ `registrationID`
Your registration ID.

→ `forceLogoff`
Set this bit to force log-off regardless of the registration ID passed.

Errors Returned

`kSessionIDNotFound`
1004 Client has attempted to log off with an invalid session ID.

`kTransactionFailed`
1005 Server search engine transaction failed.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Notes

Your application should use this function to log off a server before making the ASUnregister call.

ASEnumerateSources

The ASEnumerateSources function call gets a list of information sources available to the user. In the reply, `Sources` is an array of long integers, which actually are pointers. There are as many pointers as the number of information sources available, and each of them points to an `ASDBObjectInfo` structure.

Syntax

```
ASErr ASGetInfoSources ( ASEnumeratePBPtr pb );
```

```
struct ASEnumeratePB {
     ASParamBlockHeader
     long                dbObjectType;
     unsigned long       numberOfDBObjects;
     DBObjectInfoPtr     dbObjectList;
};
typedef ASSearchSessionPB ASSearchSessionPB,
*ASSearchSessionPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Your registration ID.

→ `dbObjectType`
This field is not used by ASEnumerateSources.

← `numberOfDBObjects`
A number of information sources returned in `dbObjectList`.

← `dbObjectList`
An array of `DBOjectInfo` structures

Errors Returned

`kTransactionFailed`
1005 Server search engine transaction failed.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kOutOfMemoryNoObjectErr`
3002 Not enough memory to create a request object.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

Available information sources are returned in an array of `DBObjectInfo` structure, which is as follows:

```
struct DBObjectInfo {
     long      dbObjectID;
     long      dbObjectFlags;
     long      dbObjectNameLength;
     char*     dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

The highest byte of DBObjectFlags is used to indicate the type of information source, as follows:

```
const     Byte kASInfoSourceTypeNormal  = 0;
const     Byte kASInfoSourceTypeWAIS    = 1;
```

The low 24 bits are used to indicate the capabilities of the information source. Of the 24 bits, the lowest three bits are used:

```
Bit 0: canCopyOriginal
Bit 1: canDoMatchTerms
Bit 2: canDoCOW
```

ASOpenSearchSession

Your application must open a search request with the ASOpenSearchSession function call before you can make a search request. You then need to pass a search session ID, returned in its reply parameter block, to any subsequent search calls. You also need to close the search session when you no longer need it.

Syntax

```
ASErr ASOpenSearchSession ( ASSearchSessionPBPtr );

struct ASSearchSessionPB {
     ASParamBlockHeader
     long      searchSessionID;
};
typedef ASSearchSessionPB ASSearchSessionPB,
*ASSearchSessionPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
You need to pass your registration ID in this field.

← `searchSessionID`
The function returns a new search session ID in this field.

Errors Returned

`kTransactionFailed`
1005 Server search engine transaction failed.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kOutOfMemoryNoObjectErr`
3002 Not enough memory to create a request object.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

Be sure to close the search session when you are done with it; keeping a search session open consumes disk space on the server. Once you close a search session, all search result information on the server is lost. Thus, you will not be able to issue an ASGetArticleMatches request, which requires that a search session from which the article was found be specified.

ASCloseSearchSession

The ASCloseSearchSession function call closes a previously opened search session. You should always close search sessions when you no longer need them.; doing so helps reduce the consumption of RAM and disk space on the server.

Syntax

```
ASErr ASCloseSearchSession ( ASSearchSessionPBPtr );

struct ASSearchSessionPB {
     ASParamBlockHeader
     long      searchSessionID;
};
typedef ASSearchSessionPB ASSearchSessionPB,
*ASSearchSessionPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	You need to pass your registration ID in this field.
→	`searchSessionID`	Search session ID returned from ASOpenSearchSession.

Errors Returned

`kTransactionFailed` 1005	Server search engine transaction failed.
`kInvalidSearchSessionIDErr` 1028	Search session ID passed was invalid.

`kOutOfMemoryNoObjectErr`
3002 Not enough memory to create a request object.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

ASSearch

The ASSearch function call initiates a new search with specified parameters. Note that you can specify only one information source for each search request.

Syntax

```
ASErr ASBeginSearch ( ASSearchPBPtr pb );
```

```
struct ASSearchPB {
     ASParamBlockHeader
     long           searchSessionID;
     long           sourceID;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           searchType;
     long           searchID;
     unsigned long  numberOfHitsReturned;
     unsigned long  totalNumberOfHits;
     ASHitInfoPtr*  hitList;
};
typedef ASSearchPB ASSearchPB, *ASSearchPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ searchSessionID
Search session ID returned from ASOpenSearchSession.

→ sourceID ID of an information source to be searched.

→ queryLength Length of query string.

→ queryString A pointer to a buffer containing a query string; The buffer must be less than 1000 characters and cannot be empty.

→ minimumRank Minimum relevance rank required to be returned; it must be between 1 and 5, inclusive.

→ maximumHits Maximum number of hits to be returned; it must be between 1 and 30,000, inclusive.

→ earliestModDate
The earliest modification date an article must have to be returned. This number is the difference in seconds between the current date and 12:00 midnight, January 1, 2000.

→ searchType This field is reserved for future use.

← searchID A search ID is returned in this field.

← numberOfHitsReturned
The number of hits returned in the hitList field.

← totalNumberOfHits
The total number of hits found by the server; it may be different from numberOfHitsReturned, but is always equal or greater.

← hitList An array of pointers to ASHitPB structures.

Errors Returned

kTransactionFailed
1005 Server search engine transaction failed.

kInvalidSourceIDErr
1006 Information source ID passed was invalid.

kInvalidArticleIDErr
1007 Article ID passed was invalid.

kServerGoingDown
1009 Server is in the process of shutting down.

kNoSharingOnServer
1011 Server is not running AppleShare or file sharing.

`kQueryMissingStringErr`
1017 Query length was specified, but the string pointer was empty or length was 0.

`kQueryLengthRangeErr`
1018 Query length is out of range.

`kMinRankRangeErr`
1019 Minimum rank search option value is out of range.

`kMaxHitRangeErr`
1020 Maximum hits search option value is out of range.

`kInvalidSearchSessionIDErr`
1028 Search session ID passed was invalid.

`kQuerySyntaxErr`
1032 Syntax error in query string.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

You must call ASOpenSearchSession before calling ASSearch. The result is returned in a form of `ASHitInfo`. The `hitList` field of the parameter block is a list of pointers to the following structure:

```
struct ASHitInfo {
     long           articleID;
     long           modDate;
     unsigned long  fileSize;
     unsigned long  rank;
     unsigned long  titleLength;
     char*          titleString;
};
typedef ASHitInfo ASHitInfo, *ASHitInfoPtr;
```

ASGetRelatedTerms

The ASGetRelatedTerms function call retrieves a list of co-occurring terms for a particular term from an information source on the server.

Syntax

```
ASErr ASGetRelatedTerms ( ASRelatedTermsPBPtr pb );
struct ASGetRelatedTermsPB {
     ASParamBlockHeader
     long           sourceID;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  maximumTerms;
     unsigned long  numberOfTerms;
     StringPtr*          termList;
};
typedef ASGetRelatedTermsPB ASGetRelatedTermsPB,
*ASGetRelatedTermsPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
→	`sourceID`	Source ID to be used.
→	`queryLength`	Length of the original term in bytes.
→	`queryString`	Pointer to the original term to be expanded.
→	`maximumTerms`	The maximum number of terms to be returned.
←	`numberOfTerms`	Number of related terms returned.
←	`termList`	An array of pointers to co-occurring terms in a Pascal string.

Errors Returned

`kTransactionFailed`
1005 Server search engine transaction failed.

`kInvalidSourceID`
1006 Information source ID passed was invalid.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kQueryMissingStringErr`
1017 Query length was specified, but the string pointer was empty or length was 0.

`kQueryLengthRangeErr`
1018 Query length is out of range.

`kMaxTermsRangeErr`
1021 Value specified in `maximumTerms` field is out of range.

`kQuerySyntaxErr`
1032 Syntax error in query string.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

You can specify only one source at a time.

ASGetArticleText

The ASGetArticleText function call retrieves the text of an article whose information source ID and document ID are specified. Since this call allows your application to retrieve only a portion of the text, you also need to specify the start byte offset and the length of text wanted.

Syntax

```
ASErr ASGetArticleText ( ASGetArticleTextPBPtr pb );

struct ASGetArticleTextPB {
     ASParamBlockHeader
     long           searchSessionID;
     long           sourceID;
     long           articleID;
     unsigned long  startOffset;
     unsigned long  length;
     Handle         text;
};
typedef ASGetArticleTextPB ASGetArticleTextPB ,
*ASGetArticleTextPB Ptr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
→	`searchSessionID`	ID of a search session in which this article was found; use 0 if no session is open.
→	`sourceID`	ID of a source to which this article belongs.
→	`articleID`	ID of the article desired.
→	`startOffset`	Byte offset from the beginning of the article.
→	`length`	Pass the number of bytes wanted.
←	`text`	A handle to the article text.

Errors Returned

`kTransactionFailed`
1005 Server search engine transaction failed.

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kInvalidArticleIDErr`
1007 Article ID passed was invalid.

`kNoAccessPrivileges`
1008 User tried to gain access to a file but lacked access privileges for it.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kFileNotFound`
1013 The article was not found on the server.

`kInvalidTextBoundErr`
1022 Start offset value or text length specified is out of range.

`kFileModSinceIndexErr`
1033 The information requested could not be retrieved because the file has been modified since the last index time.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

ASGetArticleInfo

The ASGetArticleInfo function call retrieves information about an article, returning the article's modification date, title, and size.

Syntax

```
ASErr ASGetArticleInfo ( ASGetArticleInfoPBPtr pb );

struct ASGetDocInfoPB {
     ASParamBlockHeader
     long           sourceID;
     long           articleID;
     long           modDate;
     unsigned long  size;
     unsigned long  titleLength;
     char*titleString;
};
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
→	`sourceID`	ID of a source to which this article belongs.
→	`articleID`	ID of the article desired.
←	`modDate`	The article's modification date.
←	`size`	Size of the article.
←	`titleLength`	The length of the article title string.
←	`titleString`	The article title.

Errors Returned

`kTransactionFailed` 1005 Server search engine transaction failed.

`kInvalidSourceIDErr` 1006 Information source ID passed was invalid.

`kInvalidArticleIDErr` 1007 Article ID passed was invalid.

`kNoAccessPrivileges`
1008 User tried to gain access to a file but lacked access privileges for it.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kFileNotFound`
1013 The article was not found on the server.

`kFileModSinceIndexErr`
1033 The information requested could not be retrieved because the file has been modified since the last index time.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

ASGetArticleMatches

The ASGetArticleMatches function call retrieves a set of long integer pairs that identify the words in an article that matched the query. (This retrieval does not happen in a WAIS search.) Since this call retrieves from only a portion of the text, you need to specify the start byte offset and the length.

Syntax

```
ASErr ASGetArticleMatches ( ASGetArticleMatchesPBPtr pb );
struct ASGetArticleMatchesPB{
     ASParamBlockHeader
     long           searchSessionID;
     long           sourceID;
     long           articleID;
     unsigned long  startOffset;
     unsigned long  length;
     unsigned long  numberOfHiliteInfoRecs;
     long*          hiliteInfo;
};
typedef ASGetArticleMatchesPB ASGetArticleMatchesPB,
*ASGetArticleMatchesPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
→	`searchSessionID`	ID of a search session in which this article was found.
→	`sourceID`	ID of a source to which this article belongs.
→	`articleID`	ID of the article desired.
→	`startOffset`	Byte offset from the beginning of the article.
↔	`length`	Pass the number of bytes wanted.
←	`numberOfHiliteInfoRec`	Number of matches returned.
←	`hiliteInfo`	An array of matches in long integer pairs.

Errors Returned

`kTransactionFailed`
1005 Server search engine transaction failed.

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kInvalidArticleIDErr`
1007 Article ID passed was invalid.

`kNoAccessPrivileges`
1008 User tried to gain access to a file but lacked access privileges for it.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server is not running AppleShare or file sharing.

`kFileNotFound`
1013 The article was not found on the server.

`kInvalidTextBoundErr`
1022 Start offset value or text length specified is out of range.

`kInvalidSearchSessionIDErr`
1028 Search session ID passed was invalid.

`kFileModSinceIndexErr`
1033 The information requested could not be retrieved because the file has been modified since the last index time.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

When the function completes, the `HiliteInfo` field of the parameter block points to an array of the `ASHiliteInfoRec` structure as follows:

```
struct ASHiliteInfoRec {
     unsigned long  offset;
     unsigned long  length;
};
typedef ASHiliteInfoRec ASHiliteInfoRec,
*ASHiliteInfoRecPtr;
```

ASAddReporter

The ASAddReporter function call adds a new reporter to the server's object database.

Syntax

```
ASErr ASAddReporter( ASReporterPBPtr pb );

struct ASReporterPB {
     ASParamBlockHeader
     long reporterID;
     unsigned long  nameLength;
     char*          nameString;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           earliestIndexDate;
     unsigned long  numberOfSources;
     long*          sourceList;
     unsigned long  deliveryDays;
     unsigned long  deliveryTime;
     long           deliveryFlags;
     unsigned long  deliveryDirLength;
     AliasPtr       deliveryDir;
};
typedef   ASReporterPB ASReporterPB, *ASReporterPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
↔	`reporterID`	Specify the ID for this reporter; if it is 0, then the library assigns an ID for you.
→	`nameLength`	The length of the reporter name string.
→	`nameString`	The reporter name.

→ `queryLength` Length of the query in bytes.

→ `queryString` A null-terminated query string.

→ `minimumRank` Must be between 1 and 5, inclusive.

→ `maximumHits` Must be between 1 and 30,000, inclusive.

→ `earliestModDate`
The earliest modification date desired.

→ `earliestIndexDate`
Set this field to 0.

→ `numberOfSources`
The number of sources in `sourceList`.

→ `sourceList` A pointer to an array of source IDs.

→ `deliveryDays`
Days of the week on which to deliver scheduled search results.

→ `deliveryTime`
Scheduled search delivery time.

→ `deliveryFlags`
Set bit 0 to 1 if this scheduled search is active.

→ `deliveryDirLength`
Number of bytes in `deliveryDir`.

→ `deliveryDir` An alias pointer of a directory to which scheduled search results should be delivered.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kInvalidObjectIDErr`
1010 Reporter ID was invalid.

`kNoSharingOnServer`
1011 Server isn’t running AppleShare or file sharing.

`kDuplicateName`
1014 A reporter by that name already exists.

`kQueryMissingStringErr`
1017 Query length was specified, but the string pointer was empty or length was 0.

`kQueryLengthRangeErr`
1018 Query length is out of range.

`kMinRankRangeErr`
1019 Minimum rank search option value is out of range.

`kMaxHitsRangeErr`
1020 Maximum hits search option value is out of range.

`kInvalidObjectNameErr`
1025 Reporter name is invalid or missing.

`kInvalidObjectDataErr`
1026 Object data or data length is invalid.

`kInvalidFlagErr`
1027 `deliveryFlags` has an invalid flag set.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

`kInvalidDeliveryDaysErr`
3407 Invalid delivery days were specified.

`kInvalidDeliveryTimeErr`
3408 Invalid delivery time was specified.

Notes

The `deliveryDays` field specifies the days of the week on which the scheduled search result should be delivered. It is a long integer, but only the lowest 7 bits are used. Bit assignments are as follows:

bit 0 = Sunday

bit 1 = Monday

bit 2 = Tuesday

bit 3 = Wednesday

bit 4 = Thursday

bit 5 = Friday

bit 6 = Saturday

To deliver every day of the week, set all of the bits ($007F).

The `deliveryTime` field specifies the time when the scheduled search should be delivered. The time is expressed in the number of seconds since midnight. For example, if the delivery time is 1:10:00 A.M., it should be 4200:

(1 x 60 x 60) + (10 x 60) = 4200

All other bits in the `deliveryTime` field are reserved for future use by Apple.

ASModifyReporter

The ASModifyReporter function call modifies an already existing reporter on the server's object database.

Syntax

```
ASErr ASModifyReporter ( ASReporterPBPtr pb );
```

```
struct ASReporterPB {
     ASParamBlockHeader
     long           reporterID;
     unsigned long  nameLength;
     char*          nameString;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           earliestIndexDate;
     unsigned long  numberOfSources;
     long*          sourceList;
     unsigned long  deliveryDays;
     unsigned long  deliveryTime;
     long           deliveryFlags;
     unsigned long  deliveryDirLength;
     AliasPtr       deliveryDir;
};
typedef   ASReporterPB ASReporterPB, *ASReporterPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `reporterID` Specify the ID for this reporter; if it is 0, then the Library assigns an ID for you.

→ `nameLength` The length of the reporter name string.

→ `nameString` The reporter name.

→ `queryLength` Length of the query in bytes.

→ `queryString` A null-terminated query string.

→ `minimumRank` Must be between 1 and 5, inclusive.

→ `maximumHits` Must be between 1 and 30,000, inclusive.

→ `earliestModDate`
The earliest modification date desired.

→ `earliestIndexDate`
This field is for internal use only.

→ `numberOfSources`
The number of sources in sourceList.

→ `sourceList` A pointer to an array of source IDs.

→ `deliveryDays`
Days of the week on which to deliver scheduled search result.

→ `deliveryTime`
Scheduled search delivery time.

→ `deliveryFlags`
Set bit 0 to 1 if this scheduled search is active.

→ `deliveryDirLength`
Number of bytes in `deliveryDir`.

→ `deliveryDir` An alias pointer of a directory to which scheduled search results should be delivered.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kInvalidObjectIDErr`
1010 Reporter ID specified was invalid.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kQueryMissingStringErr`
1017 Query length was specified, but the string pointer was empty or length was 0.

`kQueryLengthRangeErr`
1018 Query length is out of range.

`kMinRankRangeErr`
1019 Minimum rank search option value is out of range.

`kMaxHitsRangeErr`
1020 Maximum hits search option value is out of range.

`kInvalidObjectNameErr`
1025 Reporter name is invalid or missing.

`kInvalidObjectDataErr`
1026 Object data or data length is invalid.

`kInvalidFlagErr`
1027 `deliveryFlags` has an invalid flag set.

`kInvalidSearchSessionIDErr`
1028 Search session ID passed was invalid.

`kOutOfMemoryNoObjectErr`
3002 Not enough memory to create a request object.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kInvalidDeliveryDaysErr`
3407 Invalid delivery days were specified.

`kInvalidDeliveryTimeErr`
3408 Invalid delivery time was specified.

Notes

The `deliveryDays` field specifies the days of the week on which the scheduled search result should be delivered. It is a long integer, but only the lowest 7 bits are used. Bit assignments are as follows:

bit 0 = Sunday

bit 1 = Monday

bit 2 = Tuesday

bit 3 = Wednesday

bit 4 = Thursday

bit 5 = Friday

bit 6 = Saturday

To deliver every day of the week, set all of the bits ($007F).

The `deliveryTime` field specifies the time at which the scheduled search should be delivered. The time is expressed in the number of seconds since midnight. For example, if the delivery time is 1:10:00 A.M., it should be 4200:

(1 x 60 x 60) + (10 x 60) = 4200

All other bits in the `deliveryTime` field are reserved for future use by Apple.

ASGetReporter

The ASGetReporter function call retrieves information about an already existing reporter on the server's object database. The only parameter you need to specify besides the header is the reporter ID.

Syntax

```
ASErr ASGetReporter( ASReporterPBPtr pb );

struct ASReporterPB {
     ASParamBlockHeader
     long           reporterID;
     unsigned long  nameLength;
     char*          nameString;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           earliestIndexDate;
     unsigned long  numberOfSources;
     long*          sourceList;
     unsigned long  deliveryDays;
     unsigned long  deliveryTime;
     long           deliveryFlags;
     unsigned long  deliveryDirLength;
     AliasPtr       deliveryDir;
};
typedef   ASReporterPB ASReporterPB, *ASReporterPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
→	`reporterID`	ID of the reporter desired.
←	`nameLength`	The length of the reporter name string.
←	`nameString`	The reporter name.
←	`queryLength`	Length of the query in bytes.

← `queryString` A null-terminated query string.

← `minimumRank` Minimum rank.

← `maximumHits` Maximum hits to find.

← `earliestModDate`
The earliest modification date for articles to be found.

← `earliestIndexDate`
This field is for internal use only.

← `numberOfSources`
The number of sources in sourceList.

← `sourceList` A pointer to an array of source IDs.

← `deliveryDays`
Days of the week scheduled search results are delivered.

← `deliveryTime`
Scheduled search delivery time.

← `deliveryFlags`
Scheduled search is active if bit 0 is set.

← `deliveryDirLength`
Number of bytes in `deliveryDir`.

← `deliveryDir` An alias pointer of a directory to which scheduled search results are delivered.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

ASDeleteReporter

The ASDeleteReporter function call deletes an existing reporter on the server object database. The only parameter you need to specify besides the header is the reporter ID.

Syntax

```
ASErr ASDeleteReporter( ASReporterPBPtr pb );

struct ASReporterPB {
     ASParamBlockHeader
     long           reporterID;
     unsigned long  nameLength;
     char*          nameString;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           earliestIndexDate;
     unsigned long  numberOfSources;
     long*          sourceList;
     unsigned long  deliveryDays;
     unsigned long  deliveryTime;
     long           deliveryFlags;
     unsigned long  deliveryDirLength;
     AliasPtrdeliveryDir;
};
typedef   ASReporterPB ASReporterPB, *ASReporterPBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.
→	`completionProc`	A pointer to a completion routine.
→	`registrationID`	Pass your registration ID in this field.
→	`reporterID`	ID of the reporter to be deleted.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

ASEnumerateReporters

The ASEnumerateReporters function call returns a list of all reporters on the server object database for this user.

Syntax

```
ASErr ASEnumerateReporters( ASEnumeratePBPtr pb );

struct ASEnumeratePB {
     ASParamBlockHeader
     long           dbObjectType;
     unsigned long  numberOfDBObjects;
     DBObjectInfo*  dbObjectList;
};
typedef   ASEnumeratePB ASEnumeratePB, *ASEnumeratePBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `dbObjectType`
This field is not used for the ASEnumerateReporters call.

← `numberOfDBObjects`
Number of reporters returned.

← `dbObjectList`
List of `ASDBObjectInfo` structures that contain information about reporters.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexisting session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kOutOfMemoryNoObjectErr`
3002 Not enough memory to create a request object.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

Each element in the `dbObjectList` field of the parameter block is a block of bytes that can be cast to the `DBObjectInfo` structure, which is shown here. The first bit of the `dbObjectFlags` field is set if the scheduled search is active.

```
struct DBObjectInfo {
     long           dbObjectID;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

ASAddDBObject

The ASAddDBObject function call adds an object that contains custom information to the server's object database.

Syntax

```
ASErr ASAddDBObject( ASDBObjectPBPtr pb );

struct ASDBObjectPB {
     ASParamBlockHeader
     long           dbObjectType;
     DBObjectInfo   dbObjectInfo;
     long           sourceID;
     unsigned long  dbObjectDataSize;
     Ptr            dbObjectData;
};
typedef ASDBjectPB ASDBObjectPB, *ASDBObjectPBPtr;

struct DBObjectInfo {
     long           dbObjectID;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

Parameters

→ `userRefCo` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `dbObjectType`
Type of the object.

→ `dbObjectInfo`
ID, name, and flags for the object; the ID must be unique for the object type.

→ `sourceID` This field is not used by ASAddDBObject call.

→ `dbObjectDataSize`
Number of bytes in `dbObjectData`.

→ `dbObjectData`
A pointer to a buffer containing the object's data.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kDuplicateName`
1014 A reporter by that name already exists.

`kInvalidObjectNameErr`
1025 Reporter name is invalid or missing.

`kInvalidObjectDataErr`
1026 Object data or data length is invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

ASModifyDBObject

The ASModifyDBObject function call modifies an object that is already in the server's object database. Making this call is equivalent to making an ASDeleteDBObject call and ASAddDBObject call, except that the object's ID is retained.

Syntax

```
ASErr ASModifyDBObject( ASDBObjectPBPtr pb );

struct ASDBObjectPB {
     ASParamBlockHeader
     long           dbObjectType;
     DBObjectInfo   dbObjectInfo;
     long           sourceID;
     unsigned long  dbObjectDataSize;
     Ptr            dbObjectData;
};
typedef ASDBObjectPB ASDBObjectPB, *ASDBObjectPBPtr;

struct DBObjectInfo {
     long           dbObjectID;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `dbObjectType`
Type of the object.

→ `dbObjectInfo`
ID, name, and flags for the object.

→ `sourceID` This field is not used by the ASModifyDBObject call.

→ `dbObjectDataSize`
Number of bytes in `dbObjectData`.

→ `dbObjectData`
A pointer to a buffer containing the object's data.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexisting session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kDuplicateName`
1014 A reporter by that name already exists.

`kInvalidObjectNameErr`
1025 Reporter name is invalid or missing.

`kInvalidObjectDataErr`
1026 Object data or data length is invalid.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

ASGetDBObject

The ASGetDBObject function call retrieves information on an object in the server's object database.

Syntax

```
ASErr ASGetDBObject ( ASDBObjectPBPtr pb );

struct ASDBObjectPB {
     ASParamBlockHeader
     long           dbObjectType;
     DBObjectInfo   dbObjectInfo;
     long           sourceID;
     unsigned long  dbObjectDataSize;
     Ptr            dbObjectData;
};
typedef ASDBObjectPB ASDBObjectPB, *ASDBObjectPBPtr;
```

```
struct DBObjectInfo {
     long           dbObjectID;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `dbObjectType`
Type of the object.

↔ `dbObjectInfo`
Specify the ID of the object; name and flags are returned.

→ `sourceID` This field is not used by the ASGetDBObject call.

→ `dbObjectDataSize`
Number of bytes in `dbObjectData`.

→ `dbObjectData`
A pointer to a buffer containing the object's data.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexisting session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

ASDeleteDBObject

The ASDeleteDBObject function call deletes an object in the server object database.

Syntax

```
ASErr ASDeleteDBObject ( ASDBObjectPBPtr pb );

struct ASDBObjectPB {
     ASParamBlockHeader
     long           dbObjectType;
     DBObjectInfo   dbObjectInfo;
     long           sourceID;
     unsigned long  dbObjectDataSize;
     Ptr            dbObjectData;
};
typedef ASDBObjectPB ASDBObjectPB, *ASDBObjectPBPtr;

struct DBObjectInfo {
     long           dbObjectID;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `dbObjectType`
Type of the object.

↔ `dbObjectInfo`
Specify only the ID of the object to be deleted.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kInvalidObjectIDErr`
1010 Reporter ID passed was invalid.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

ASEnumerateDBObjects

Returns a list of objects whose type was specified from the server's object database.

Syntax

```
ASErr ASEnumerateDBObjects( ASEnumeratePBPtr pb );

struct ASEnumeratePB {
     ASParamBlockHeader
     long           dbObjectType;
     unsigned long  numberOfDBObjects;
     DBObjectInfo*  dbObjectList;
};
typedef   ASEnumeratePB ASEnumeratePB, *ASEnumeratePBPtr;
```

Parameters

→	`userRefCon`	This field remains unchanged; for the caller's private use.
←	`error`	Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `dbObjectType`
Type of objects to be enumerated.

← `numberOfDBObjects`
Number of objects returned in `dbObjectList`.

← `dbObjectList`
A pointer to an array of `DBObjectInfo` structure.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

`kNoCompletionProcErr`
3109 No completion routine pointer was specified.

Notes

Each element in the `dbObjectList` field of the parameter block is a block of bytes that can be cast to the `DBObjectInfo` structure, which is shown here.

```
struct DBObjectInfo {
     long           dbObjectID;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
```

ASImportReporter

The ASImportReporter function call imports a reporter from a reporter file. This call makes an ASAddReporter call internally and returns a new reporter ID in the `reporterID` field. The reporter file must be of type `RPTR` or `kReporterFileType`.

Syntax

```
ASErr ASImportReporter( ASImportExportPBPtr pb );

struct ASImportExportPB {
     ASParamBlockHeader
     FSSpec         reporterFile;
     long           reporterID;
     unsigned long  nameLength;
     char*          nameString;
};
typedef   ASImportExportPB ASImportExportPB,
*ASImportExportPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `reporterFile`
FSSpec of the reporter file from which a reporter is imported.

← `reporterID` ID of the reporter that is imported.

← `nameLength` Length of the reporter name string.

← `nameString` Name of the reporter that is imported.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexistent session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

ASExportReporter

The ASExportReporter function call exports an existing reporter to a reporter file. You only need to specify the reporter's ID and the destination file. The function completes the request by making an ASGetReporter call to obtain the latest information about the reporter from the server and by saving it in the proper format on the disk.

Syntax

```
ASErr ASExportReporter ( ASImportExportPBPtr pb );

struct ASImportExportPB {
     ASParamBlockHeader
     FSSpec         reporterFile;
     long           reporterID;
     unsigned long  nameLength;
     char*          nameString;
};
typedef   ASImportExportPB ASImportExportPB,
*ASImportExportPBPtr;
```

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `reporterFile`
FSSpec to be used for reporter file creation.

← `reporterID` ID of the reporter to be exported.

← `nameLength` Length of the reporter name string.

← `nameString` Name of the reporter that was imported.

Errors Returned

`kInvalidRequestForGuest`
1001 Guest is attempting to make a database request.

`kInvalidRequestForGuest`
1002 Client sent a nonexisting session ID.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

ASGetOriginalFile

The ASGetOriginalFile function call retrieves the original file of a document or its alias, if the information source ID and document ID are specified. In addition to these two IDs, you also need to specify, in the `fileDestination` field, the destination to which the original file should be copied. The function automatically copies the file, both data and resource forks, and sets the Finder information for you once it is copied. By setting the bit 1 of the `flags` field, you can get an alias to the original file instead of copying the actual file.

Syntax

```
ASErr ASGetOriginalFile ( ASGetOriginalFilePBPtr pb );
struct ASGetOriginalFilePB {
     ASParamBlockHeader
     long           sourceID;
     long           articleID;
     long           flags;    // 1 = original; 2 = alias
     FSSpec         fileDestination;
     AliasHandle    alias;
};
typedef ASGetOriginalFilePB ASGetOriginalFilePB,
*ASGetOriginalFilePBPtr;
```

In the case of a WAIS search, the server returns a "file not found" error (`kFileNotFound = -43`) because you cannot retrieve the original from a WAIS server.

Parameters

→ `userRefCon` This field remains unchanged; for the caller's private use.

← `error` Error code; returns 0 if the function was executed successfully.

→ `completionProc`
A pointer to a completion routine.

→ `registrationID`
Pass your registration ID in this field.

→ `sourceID` ID of a source to which the article belongs.

→ `articleID` ID of the article whose original is desired.

→ `flags` Set bit 0 if you want to copy the original file into `fileDestination`, or set bit 1 if you want an alias to the original file returned in `alias` field.

← `fileDestination`
FSSpec to the original file that was copied; this field is used only if bit 0 of `flags` is set.

← `alias` Alias to the original file on the server; this field is used only if bit 1 of `flags` is set.

Errors Returned

`kInvalidSourceIDErr`
1006 Information source ID passed was invalid.

`kInvalidArticleIDErr`
1007 Article ID passed was invalid.

`kNoAccessPrivileges`
1008 User tried to gain access to a file but lacked access privileges for it.

`kServerGoingDown`
1009 Server is in the process of shutting down.

`kNoSharingOnServer`
1011 Server isn't running AppleShare or file sharing.

`kFileNotFound`
1013 The file was not found on the server.

`kInvalidFlagErr`
1027 `keyAsGetOriginalFileFlags` has an invalid flag set.

`kInvalidRegistrationIDErr`
3102 Invalid registration ID was passed.

`kNotLoggedOnErr`
3106 Client is currently not logged on to any server.

Appendix A Constants and Errors

This appendix presents a summary of AppleSearch application programming interface (API) constants and errors.

Search parameter block–related constants

```
const long     kMinQuerySize            = 1;
const long     kMaxQuerySize            = 1000;
const long     kMinRankAllowed          = 1;
const long     kMaxRankAllowed          = 5;
const long     kMinHitsAllowed          = 1;
const long     kMaxHitsAllowed          = 30000;
```

Reporter parameter block–related constants

```
const long     kMaxReporterName         = 20;
const long     kMinReporterQuerySize    = 0;
const long     kMinReporterSources      = 0;
const long     kMinReporterFlags        = 0;
const long     kMaxReporterFlags        = 1;
const long     kMaxDeliveryDays         = 0x7F;
const long     kMaxDeliveryTime         = 86399;
```

Get Text/Get Original parameter block–related constants

```
const long     kMinTextSize             = 1;
const long     kMaxTextSize             = 30000;
const long     kMaxGetOriginalFileFlag  = 2;
```

Get Related Terms parameter block–related constants

```
const long     kMinTermsAllowed         = 1;
const long     kMaxTermsAllowed         = 100;
```

Client-generated errors

```
const ASErr    kUnknownErr                  = 3000;
const ASErr    kOutOfMemoryErr              = 3001;
const ASErr    kOutOfMemoryNoObjectErr      = 3002;
const ASErr    kOutOfMemoryNoPBErr          = 3003;
const ASErr    kMaxRegisteredErr            = 3101;
const ASErr    kInvalidRegistrationIDErr    = 3102;
const ASErr    kAlreadyLoggedOnErr          = 3103;
const ASErr    kLogOnCancelErr              = 3104;
const ASErr    kStillLoggedOnErr            = 3105;
const ASErr    kNotLoggedOnErr              = 3106;
const ASErr    kNoIdentificationErr         = 3108;
const ASErr    kNoCompletionProcErr         = 3109;
const ASErr    kAlreadyLoggingOnErr         = 3112;
const ASErr    kReplyMatchesNoRequestErr    = 3601;
const ASErr    kInvalidReplyDataErr         = 3602;
const ASErr    kInvalidRequestDataErr       = 3603;
```

Server-generated errors

```
const ASErr    kInvalidRequestForGuestErr   = 1001;
const ASErr    kInvalidSessionIDErr         = 1002;
const ASErr    kRequestIDNotFoundErr        = 1003;
const ASErr    kSessionIDNotFoundErr        = 1004;
const ASErr    kTransactionFailedErr        = 1005;
const ASErr    kInvalidSourceIDErr          = 1006;
const ASErr    kInvalidDocIDErr             = 1007;
const ASErr    kNoAccessPrivilegesErr       = 1008;
const ASErr    kServerGoingDownErr          = 1009;
const ASErr    kInvalidObjectIDErr          = 1010;
const ASErr    kNoSharingOnServerErr        = 1011;
const ASErr    kWrongProtocolErr            = 1012;
const ASErr    kFileNotFoundErr             = 1013;
const ASErr    kDuplicateNameErr            = 1014;
const ASErr    kTooManyUsersErr             = 1015;
const ASErr    kInfoSourceHiddenErr         = 1016;
const ASErr    kNoQueryStringErr            = 1017;
const ASErr    kQueryStringTooLongErr       = 1018;
const ASErr    kInvalidMinRankErr           = 1019;
const ASErr    kInvalidMaxHitsErr           = 1020;
const ASErr    kInvalidMaxTermsErr          = 1021;
const ASErr    kInvalidTextBoundsErr        = 1022;
const ASErr    kInvalidOperationErr         = 1023;
const ASErr    kInvalidObjectTypeErr        = 1024;
```

```
const ASErr    kInvalidObjectNameErr         = 1025;
const ASErr    kInvalidObjectDataErr         = 1026;
const ASErr    kInvalidFlagsErr              = 1027;
const ASErr    kInvalidSubSessionIDErr       = 1028;
const ASErr    kParamErr                     = 1030;
const ASErr    kInternalErr                  = 1031;
const ASErr    kQuerySyntaxErr               = 1032;
const ASErr    kFileModSinceIndexErr         = 1033;
const ASErr    kWAISError                    = 1039;
const ASErr    kWAISSrcUnavailable           = 1040;
const ASErr    kTCPProblem                   = 1041;
```

Appendix B Apple Events Summary

This appendix presents a summary of the Apple events application programming interface (API).

Event classes

```
#define kAppleSearchRequest        'bgrt'
#define kAppleSearchReply          'brep'
```

Event IDs

```
#define kAEASRegister              'rstr'
#define kAEASUnregister            'ustr'
#define kAEASLogon                 'cnct'
#define kAEASLogoff                'ocnt'
#define kAEASSetAutoLogonFlag      'salg'
#define kAEASGetAutoLogonFlag      'galg'
#define kAEASStatusChanged         'chng'

#define kAEASOpenSearchSession     'ossn'
#define kAEASCloseSearchSession    'cssn'
#define kAEASSearch                'srch'

#define kAEASImportReporter        'irpt'
#define kAEASExportReporter        'xrpt'

#define kAEASAddReporter           'arpt'
#define kAEASModifyReporter        'mrpt'
#define kAEASGetReporter           'grpt'
#define kAEASDeleteReporter        'drpt'
#define kAEASEnumerateReporters    'erpt'

#define kAEASAddDBObject           'adbo'
#define kAEASModifyDBObject        'mdbo'
#define kAEASGetDBObject           'gdbo'
#define kAEASDeleteDBObject        'ddbo'
#define kAEASEnumerateDBObjects    'edbo'
```

```
#define kAEASGetArticleText        'gatx'
#define kAEASGetArticleMatches     'gamt'
#define kAEASGetArticleInfo        'gain'
#define kAEASEnumerateSources      'esrc'
#define kAEASGetRelatedTerms       'gcow'
#define kAEASGetOriginalFile       'epfi'
```

Apple event keywords

```
#define keyASRegistrationID        'btid'
#define keyASPPCSessionID          'ssid'
#define keyASUserRefCon            'rfcn'
#define keyASServerName            'svrn'
#define keyASIsGuest               'igst'
#define keyASDeliverUpdateFlag     'dupd'
#define keyASHaveConnection        'vldc'
#define keyASAutoLogonFlag         'alin'

#define keyASSearchSessionID       'seid'
#define keyASSearchID              'srid'
#define keyASQueryString           'qrey'
#define keyASSearchType            'srtp'
#define keyASMinimumRank           'mino'
#define keyASMaximumHits           'maxo'
#define keyASEarliestModDate       'dato'
#define keyASEarliestIndexDate     'idto'
#define keyASSourceIDList          'isil'
#define keyASSourceList            'dlst'
#define keyASArticleID             'arid'
#define keyASNumberOfHits          'thit'
#define keyASNumberOfSources       'nums'
#define keyASHitList               'hlst'

#define keyASArticleText           'atxt'
#define keyASArticleLength         'atln'
#define keyASSourceID              'isid'
#define keyASStartOffset           'ofst'
#define keyASMaxLength             'mlen'
#define keyASHiliteInfoList        'hlst'

#define keyASReporterName          'rpnm'
#define keyASReporterID            'rpid'
#define keyASDeliveryDir           'dldd'
#define keyASDeliveryDays          'dldy'
#define keyASDeliveryTime          'dldt'
#define keyASDeliveryFlags         'dlfl'

#define keyASReporterFile          'rpfl'
#define keyASFileSpec              'fspc'
```

```
#define keyASGetOriginalFileFlags  'gofl'
#define keyASAliasRecord           'oals'

#define keyASMaxTerms              'mxtm'
#define keyASNumberOfTerms         'numt'
#define keyASTermList              'tlst'

#define keyASDBObjectType          'otyp'
#define keyASDBObjectID            'obid'
#define keyASDBObjectFlags         'obfl'
#define keyASDBObjectName          'onam'
#define keyASDBObjectDataLength    'odln'
#define keyASDBObjectData          'odat'
#define keyASDBObjectList          'olst'

#define keyASReporterList          'ridl'
#define keyASTitleLength           'tlen'
#define keyASTitle                 'ttle'
#define keyASModDate               'mdat'
```

Structures

```
typedef struct {
     long      hitArticleID;
     long      hitModDate;
     long      hitSize;
     long      hitRank;
     Str31     hitTitle;
} HitDataRecord, *HitDataRecPtr, **HitDataRecHndl;

typedef struct {
     long      sourceID;
     Str31     sourceName;
     long      sourceFlags;
} InfoSourceRecord, *InfoSourceRecPtr,
**InfoSourceRecHndl;

typedef struct {
     long      objID;
     Str31     objName;
     long      objFlags;
} DBObjectRecord, *DBObjectRecPtr, **DBObjectRecHndl;

typedef struct {
     long      repID;
     Str31     repName;
     long      repFlags;
} ReporterInfoRecord, *ReporterInfoRecPtr,
**ReporterInfoRecHndl;
```

Appendix C C Library Summary

This appendix presents a summary of the C Library application programming interface (API).

Object types

```
#define kObjType_Reporter          1
#define kObjType_HitList           2
#define kObjType_NewspaperDate     3
```

Type definitions

```
typedef long   ASErr;
typedef long   (*ASProcPtr)(char*, long);
```

Parameter blocks

```
#define ASParamBlockHeader \
     long      userRefCon;
     ASErr     error;
     ASProcPtr completionProc;
     long      registrationID;
```

ASRegisterPB

```
struct ASRegisterPB {
     ASParamBlockHeader
     long      identifier;
};
typedef ASRegisterPB ASRegisterPB, *ASRegisterPBPtr;
```

ASSelectServerPB

```
struct ASSelectServerPB {
     ASParamBlockHeader
     Ptr            defaultServer;
     unsigned long  dataSize;
     Ptr            dataPtr;
};
typedef ASSelectServerPB ASSelectServerPB,
*ASSelectServerPBPtr;
```

ASLogonPB

```
struct ASLogonPB {
     ASParamBlockHeader
     unsigned long  flags;
     Ptr            dataPtr;
     unsigned long  defaultUserNameLength;
     char*          defaultUserNameString;
     unsigned long  serverNameLength;
     char*          serverNameString
};
typedef ASLogonPB ASLogonPB, *ASLogonPBPtr;
```

ASLogoffPB

```
struct ASLogoffPB {
     ASParamBlockHeader
     Boolean        forceLogoff;
};
typedef ASLogoffPB ASLogoffPB, *ASLogoffPBPtr;
```

ASSearchSessionPB

```
struct ASSearchSessionPB {
     ASParamBlockHeader
     long           searchSessionID;
};
typedef ASSearchSessionPB ASSearchSessionPB,
*ASSearchSessionPBPtr;
```

ASEnumeratePB

```
struct ASDBObjectInfo {
     long           dbObjectID;
     Str31          dbObjectName;
     long           dbObjectFlags;
     unsigned long  dbObjectNameLength;
     char*          dbObjectNameString;
};
typedef DBObjectInfo DBObjectInfo, *DBObjectInfoPtr;
struct ASEnumeratePB {
     ASParamBlockHeader
     long           dbObjectType;
     long           numberOfDBObjects;
     DBObjectInfo*  dbObjectList;
};
typedef ASDBObjectInfo ASDBObjectInfo, *ASDBObjectInfoPtr;
struct ASEnumeratePB {
     ASParamBlockHeader
     long               dbObjectType;
     unsigned long      numberOfDBObjects;
     ASDBObjectInfoPtr  dbObjectList;
};
typedef ASEnumeratePB ASEnumeratePB, *ASEnumeratePBPtr;
```

ASSearchPB

```
struct ASHitInfo {
     long           articleID;
     long           modDate;
     unsigned long  fileSize;
     unsigned long  rank;
     unsigned long  titleLength;
     char*          titleString;
};
typedef ASHitInfo ASHitInfo, *ASHitInfoPtr;
```

```
struct ASSearchPB {
     ASParamBlockHeader
     long           searchSessionID;
     long           sourceID;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           searchType;
     long           searchID;
     unsigned long  numberOfHitsReturned;
     unsigned long  totalNumberOfHits;
     ASHitInfoPtr*  hitList;
};
typedef ASSearchPB ASSearchPB, *ASSearchPBPtr;
```

ASGetArticleInfoPB

```
struct ASGetArticleInfoPB {
     ASParamBlockHeader
     long           sourceID;
     long           articleID;
     long           modDate;
     unsigned long  size;
     unsigned long  titeLength;
     unsigned long  titleString;
typedef ASGetArticleInfoPB ASGetArticleInfoPB,
*ASGetArticleInfoPBPtr;
```

ASGetArticleTextPB

```
struct ASGetArticleTextPB {
     ASParamBlockHeader
     long           searchSessionID;
     long           sourceID;
     long           articleID;
     unsigned long  startOffset;
     unsigned long  length;
     Handle         text;
};
typedef ASGetArticleTextPB    ASGetArticleTextPB,
                              *ASGetArticleTextPBPtr;
```

ASGetArticleMatchesPB

```
struct ASGetArticleMatchesPB {
     ASParamBlockHeader
     long           searchSessionID;
     long           sourceID;
     long           articleID;
     unsigned long  startOffset;
     unsigned long  length;
     unsigned long  numberOfHiliteInfoRecs;
     long*          hiliteInfo;
};
typedef ASGetArticleMatchesPB ASGetArticleMatchesPB,
*ASGetArticleMatchesPBPtr;
struct ASHiliteInfoRec {
     unsigned long  offset;
     unsigned long  length;
};
typedef ASHiliteInfoRec ASHiliteInfoRec,
*ASHiliteInfoRecPtr;
```

ASGetFilePB

```
struct ASGetOriginalFilePB {
     ASParamBlockHeader
     long           sourceID;
     long           articleID;
     long           flags;
     FSSpec         fileDestination;
     AliasHandle    alias;
};
typedef ASGetOriginalFilePB ASGetOriginalFilePB,
*ASGetOriginalFilePBPtr;
```

ASReporterPB

```
struct ASReporterPB {
     ASParamBlockHeader
     long reporterID;
     unsigned long  nameLength;
     char*          nameString;
     unsigned long  queryLength;
     char*          queryString
     unsigned long  minimumRank;
     unsigned long  maximumHits;
     long           earliestModDate;
     long           earliestIndexDate;
     unsigned long  numberOfSources;
     long*          sourceList;
     unsigned long  deliveryDays;
     unsigned long  deliveryTime;
     long           deliveryFlags;
     unsigned long  deliveryDirLength;
     AliasPtr       deliveryDir;
};
typedef ASReporterPB ASReporterPB, *ASReporterPBPtr;
```

DBObjectPB

```
struct ASDBObjectPB {
     ASParamBlockHeader
     long           dbObjectType;
     DBObjectInfo   dbObjectInfo;
     long           sourceID;
     unsigned long  dbObjectDataSize;
     Ptr            dbObjectData;
};
typedef ASDBObjectPB ASDBObjectPB, *ASDBObjectPBPtr;
```

ASImportExportPB

```
struct ASImportExportPB {
     ASParamBlockHeader
     FSSpec         reporterFile;
     long           reporterID;
     Str31          reporterName;
     unsigned long  nameLength;
     char*          nameString;
};
typedef ASImportExportPB ASImportExportPB,
*ASImportExportPBPtr;
```

ASGetRelatedTermsPB

```
struct ASGetRelatedTermsPB {
     ASParamBlockHeader
     long           sourceID;
     unsigned long  queryLength;
     char*          queryString;
     unsigned long  maximumTerms;
     unsigned long  numberOfTerms;
     StringPtr*     termList;
};
typedef ASGetRelatedTermsPB ASGetRelatedTermsPB,
*ASGetRelatedTermsPBPtr;
```

C API functions

Housekeeping routines

```
void      ASInitialize ( void );
void      ASQuit ( void );
ASErr     ASDoIdle ( void );
ASErr     ASRegister ( ASRegisterPBPtr pb );
ASErr     ASUnregister ( ASRegisterPBPtr pb );
ASErr     ASSelectServer ( ASSelectServerPBPtr pb);
ASErr     ASLogon ( ASLogonPBPtr pb );
ASErr     ASLogoff ( ASLogoffPBPtr pb );
```

Search routines

```
ASErr     ASOpenSearchSession( ASSearchSessionPBPtr pb,
          Boolean async );
ASErr     ASCloseSearchSession( ASSearchSessionPBPtr pb,
          Boolean async );
ASErr     ASSearch( ASSearchPBPtr pb, Boolean asynch );
```

Data retrieval routines

```
ASErr     ASEnumerateSources ( ASEnumeratePBPtr pb,
          Boolean async );
ASErr     ASGetArticleText ( ASGetArticleTextPBPtr pb,
          Boolean asynch );
ASErr     ASGetArticleMatches ( ASGetArticleMatchesPBPtr
          pb, Boolean asynch );
ASErr     ASGetArticleInfo ( ASGetArticleInfoPBPtr pb,
          Boolean async );
ASErr     ASGetOriginalFile ( ASGetOriginalFilePBPtr pb,
          Boolean asynch );
ASErr     ASGetRelatedTerms ( ASGetRelatedTermsPBPtr pb,
          Boolean async );
```

Reporter routines

```
ASErr     ASAddReporter ( ASReporterPBPtr pb, Boolean
          async );
ASErr     ASModifyReporter ( ASReporterPBPtr pb, Boolean
          async );
ASErr     ASGetReporter ( ASReporterPBPtr pb, Boolean
          async );
ASErr     ASDeleteReporter ( ASReporterPBPtr pb, Boolean
          async );
ASErr     ASEnumerateReporters ( ASEnumeratePBPtr pb,
          Boolean async );
ASErr     ASImportReporter ( ASImportExportPBPtr pb,
          Boolean async );
ASErr     ASExportReporter ( ASImportExportPBPtr pb,
          Boolean async );
```

DBObject routines

```
ASErr     ASAddDBObject ( ASDBObjectPBPtr pb, Boolean
          async );
ASErr     ASModifyDBObject ( ASDBObjectPBPtr pb, Boolean
          async );
ASErr     ASGetDBObject ( ASDBObjectPBPtr pb, Boolean
          async );
ASErr     ASDeleteDBObject ( ASDBObjectPBPtr pb, Boolean
          async );
ASErr     ASEnumerateDBObjects ( ASEnumeratePBPtr pb,
          Boolean async );
```