AppleSearch Update File Format Reference

Apple Computer, Inc.

This manual is copyrighted by Apple or by Apple's suppliers, with all rights reserved. Under the copyright laws, this manual may not be copied, in whole or in part, without the written consent of Apple Computer, Inc. This exception does not allow copies to be made for others, whether or not sold, but all of the material purchased may be sold, given, or lent to another person. Under the law, copying includes translating into another language.

The Apple logo is a registered trademark of Apple Computer, Inc. Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for printing or clerical errors.

© Apple Computer, Inc., 1993
20525 Mariani Avenue
Cupertino, CA 95014-6299
(408) 996-1010

Apple, the Apple logo, AppleTalk, LaserWriter, Macintosh and MPW are trademarks of Apple Computer, Inc., registered in the United States and other countries.

AppleSearch and QuickTime are trademarks of Apple Computer, Inc.

Adobe, Adobe Illustrator, and PostScript are registered trademarks of Adobe Systems, Inc.

Helvetica and Times are registered trademarks of Linotype Company.

Microsoft is a registered trademark of Microsoft Corporation.

QuarkXPress is a registered trademark of Quark, Inc.

Simultaneously published in the United States and Canada.

Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. Apple assumes no responsibility with regard to the performance of these products.

Contents

Preface / v

1 Data Containers / 1

Data container structure / 3

Standard data container types / 4
- Update file header / 4

Other data container types / 5
- Article / 5
- Article list header / 6
- Free / 7
- Compressed / 8

2 Articles / 9

Article header / 11
- Article type constants / 12

Compound articles / 12

3 AppleSearch Utility Routines / 15

File utility routines / 16
- Basic type definitions / 16
- Update info data structure / 16
- Routines / 16
- Create update file / 17
- Open update file / 17
- Close update file / 18
- Read update info / 18
- Write update info / 19
- Compact update file / 19

Article routines / 20
- Article's date/time data structure / 20
- Article header data structure / 20

Routines / 20
Count articles / 21
Find article / 21
Read article header / 22
Read article data / 23
Read partial article data / 23
Write article header / 24
Write article data / 24
Add article / 25
Delete article / 26
Get article size / 26
Get article type / 26
Get article user bytes / 27
Dispose article header / 27

Data container routines / 28
Data container info data structure / 28
Routines / 28
Add data container / 29
Delete data container / 29
Read data container header / 30
Write data container header / 30
Read data container data / 31
Write data container data / 31
Count data containers / 32
Find data container / 32
Get data container size / 33
Get data container type / 33
Get data container ID / 33
Get data container UserRefCon / 34

Preface

AppleSearch allows Macintosh users to access information on a server by providing full text search and retrieval. Search agents—called Reporters—search information sources on the AppleSearch server. The results of the search can be displayed in a newspaper-like “update,” created automatically, on a regular basis. This update can be displayed immediately or delivered to a specified folder on a hard disk.

For developers who want to customize AppleSearch for a specific purpose or simply to understand how AppleSearch works, this document describes the update file format, definitions of standard update components, and the accompanying utility routines that can be used to manipulate update documents. All the routines are written in MPW C, version 3.2.

Before using this manual, you will be better prepared if you are familiar with AppleSearch and have read the following manuals:

- *AppleSearch User’s Guide*
- *AppleSearch System Administrator’s Guide*
- *AppleSearch Client Developer’s Guide*

Key definitions

Article An article contains a title, a date, a source where the article came from, and the data itself. The data may be text, a PICT, a QuickTime movie, any other data types, or a combination of data types. From an application point of view, an update is a collection of articles.

Compound article An article that contains two or more subarticles. The subarticles can be of the same or different data types. For example, a compound article might contain two separate images of type PICT. Another one might be composed of text, an image of type 'TIFF', and a sound.

Data container A building block of an AppleSearch update document. All the data in a document are organized into a set of data containers. Typically, one document contains a number of data containers of different types.

Subarticle A subarticle is an element within a compound article. Its data structure includes its size, a type, and the data itself.

Software licensing

For information on developer licensing, please contact

Software Licensing Department
Apple Computer, Inc.
20525 Mariani Avenue, M/S 38-1
Cupertino, California 95014
U.S.A.
Telephone: (408) 974-4667
AppleLink: SW.LICENSE

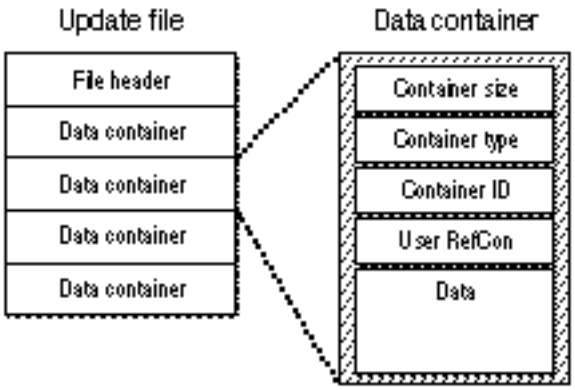
1 Data Containers

An AppleSearch update file is composed of *data containers*. A data container is a repository for information. A single data container can be of any size up to 4 gigabytes (if the operating system allows files that large) and can contain any type of information. For example, it can contain information for a text article, the sorting state of the update, or layout information for the update.

The *data container type* indicates the kind of data that a data container holds. Standard data container types are defined by Apple and described in this document.

A *file header* is a specialized data container that contains general information about the update file.

Each update file consists of a file header followed by zero or more data containers as illustrated.



Because an AppleSearch update file is made up of modular data containers, an application is able to read only the data it requires or is capable of reading. At the same time, the data types can be extended. This is similar to the way in which the Apple Event Manager uses descriptors to make up an Apple event.

Data container structure

The general structure for each data container is a 16-byte header with information about that particular data container, followed by the actual data.

The following section, “Standard Data Container Types,” describes the set of predefined data types. Other data types can be created as needed following the structure defined in this section.

The “File Utility Routines” section in Chapter 3, “AppleSearch Utility Routines,” describes a set of routines that will help you examine, read, and write data containers easily, without having to know the details of this structure.

Format

The information above the broken line is the data container header. The information below the broken line is the data container data.

```
long      ContainerSize;
long      ContainerType;
long      ContainerID
long      UserRefCon;
-------------------------------------------------------------
Bytes[]   Data;
```

Parameters

`ContainerSize`	The size of the data in the container (not including the 16 bytes taken up by the container header).
`ContainerType`	The kind of data kept in the container. For example, the designation ‘ARTL' indicates that article information is kept in the data section. There are several predesigned standard data container types. Each is explained in detail in the following section “Standard Data Container Types.”
`ContainerID`	A 4-byte application-specific value. It can be used by the application for any purpose. It can, for example, be used to assign a unique ID for each container of the same type, enabling it to be referred to from other containers by that ID.
`UserRefCon`	This field can be used by applications for any purpose.
`Data`	This section contains the actual data, the size of which is defined in the `ContainerSize` parameter.

Standard data container types

Apple supplies a set of five standard data container types. Your application should not use these reserved data container types except in the defined manner.

The five data container types are as follows:

- Update file header
- Article
- Article list header
- Free
- Compressed

Each type is discussed in detail in the following sections. The constants for the data container types, as well as other constants, are defined in the ASUConstants.h file.

Update file header

`ASUDCTypeFileHeader ('FHDR')`

As noted earlier, each AppleSearch update file has a file header. The update file header is always the first data container in an update file. Its data container type is 'FHDR'. It is made up of four fields, each using 4 bytes. These fields are defined below.

The constants for this data container type are defined in the ASUConstants.h file.

Format

The information above the broken line is the data container header, the standard 16-byte header required for every data container. The information below the broken line is the data container data, which in this case is the update file header information.

```
long 16        // size of the data section
long 'FHDR'    // data container type ASUDCTypeFileHeader
long 0         // data container id - not used
long 0         // data container userRefCcon - not used
-------------------------------------------------------
long FileFormatVersion   //
long UpdateType               //
long UpdateVersion       //
long NumOfDataContainers //
```

Description

`FileFormatVersion`
: This field indicates the version of the file format. Applications use this field to determine the format of a file, such as its file structure and header. For example, if this file header is expanded in the future to include more information and the version number is changed, then by reading the file format version an application can determine if the additional information is expected to be in the file header. The file format version for this implementation is 1.

`UpdateType`
: This field is set by the application that created the file, and it serves a purpose similar to that of an HFS file creator type. It signals viewer applications what kind of information can be in the file and how it is organized. AppleSearch client software sets this to be 'BNP1'.

`UpdateVersion`
: Applications can use this field to distinguish variations of the same update type.

`NumOfDataContainers`
: This field keeps track of the number of data containers in the file. Since the file header itself is a data container, this field has a value of at least 1.

Note

The `id` and `UserRefcon` parameters in the data container header are currently not used and are set to 0.

Other data container types

After the update file header, the other data containers follow in the update file. The available predefined types are defined in this section. Other data containers can be created according to the specification in the “Data Container Structure” section earlier in this chapter.

Article

`ASUDCTypeArticle ('ARTL')`

This standard data container type is likely to be the most frequently used data container type. It contains the individual articles that make up an update.

Articles are explained in more detail in Chapter 2, “Articles.”

The constants for this data container type are defined in the ASUConstants.h file.

Format

The information above the broken line is the data container header. The information below the broken line is the data container data.

```
long VARIABLE  // size of the data section
long 'ARTL'    // data container type ASUDCTypeArticle
long 0         // data container id - not used
long 0         // data container user ref con
-----------------------------------------------------------
long ArticleHeaderSize   // size of the article header
                            (w/o data)
long ArticleDataSize     // size of article data
long ArticleType         // article type, such as 'TEXT'.
ASUArticleDate
     ArticleDateTime          // date and time of this
article
long UserBytes           // application-specific data
CString
     ArticleTitle        // title of the article
CString
     ArticleSource       // source of the article
data                     // actual data
```

Article list header

`ASUDCTypeArticleListHeader ('ALHD')`

This standard data container type is used as the beginning of a linked list of data containers of type 'ARTL'. After this data container, the number of 'ARTL' data containers specified in the `NumOfArticles` parameter should directly follow.

The constants for this data container type are defined in the ASUConstants.h file.

Format

The information above the broken line is the data container header. The information below the broken line is the data container data.

```
long 8         // size of the data section
long 'ALHD'    // data container type
                  ASUDCTypeArticleListHeader
long 0         // data container id
long 0         // data container user ref con (could be
                  a pointer to the next article if part
                  of a linked list)
-------------------------------------------------------
long NumOfArticles  // number of articles in the list
long Head           // byte offset of the first list item
```

Notes

When the 'ALHD' data container is found, the UserRefCon field of the 'ARTL' data container that follows points to the next article in the linked list. This pointer is a byte offset to the next article data container.

If an article list header ('ALHD') data container uses this byte offset to point to the next article in a list and it is compressed using the compressed ('COMP') data container, then that byte offset will no longer be valid, since compressing a data container changes its size.

Free

`ASUDCTypeFree ('FREE')`

This standard data container type is used to indicate that the data container is no longer used and can be purged. The update utility routines set a data container’s type to 'FREE' when `ASUDeleteDataContainer()` is called.

The constants for this data container type are defined in the ASUConstants.h file.

Format

The information above the broken line is the data container header. The information below the broken line is the data container data.

```
long VARIABLE  // size of the data section from the
                  prior header
long 'FREE'    // data container type ASUDCTypeFree
long VARIABLE  // data container id from prior header
long VARIABLE  // data container userRefCon from the
                  prior header
-------------------------------------------------------
Bytes[]        // old data
```

Compressed

`ASUDCTypeCompressed ('COMP')`

The initial version of this developer's kit does not provide utility routines to work with compressed data containers; however, this type is defined for future use.

The constants for this data container type are defined in the ASUConstants.h file.

Format

The information above the broken line is the data container header. The information below the broken line is the data container data.

```
long VARIABLE  // size of compressed data
long 'COMP'    // data container type ASUDCTypeCompressed
long VARIABLE  // data container id from prior header
long VARIABLE  // data container userRefCon from prior
                  header
-------------------------------------------------------
long VARIABLE  // the original data container type
long VARIABLE  // the original data size
long VARIABLE  // compression used
data           // actual data (compressed)
```

Note

If an article list header ('ALHD') data container uses a byte offset to point to the next article in a list and it is compressed using the compressed ('COMP') data container, then that byte offset will no longer be valid, since compressing a data container changes its size.

2 Articles

A typical update file consists of a set of data containers of the type article ('ARTL'). An article is data that has been gathered by an AppleSearch Reporter. A title, date, and source are associated with each article. The data structure of the article data container type is designed to ensure compatibility among applications.

The initial release of the AppleSearch client can read and write only text articles (article type 'TEXT'). Your custom application, however, can support other data types while maintaining compatibility.

Format

The information above the broken line is the standard data container header. The information below the broken line is the data container data that describes an article and contains its data.

```
long VARIABLE  // size of the data section
long 'ARTL'    // data container type ASUDCTypeArticle
long 0         // data container id - not used
long 0         // data container user ref con
-------------------------------------------------------
long ArticleHeaderSize   // size of the article header
long ArticleDataSize     // size of article data
long ArticleType         // article type, such as 'TEXT'.
ASUArticleDate
     ArticleDateTime          // date and time of this
article
long UserBytes           // application-specific data
CString
     ArticleTitle        // title of the article
CString
     ArticleSource       // source of the article
bytes[]
     data                // actual data
```

Notes

An article is contained in the data section of the data container. The article is composed of an article header and the data itself. The article header section contains the article’s size, type, date and time, title, and source. The article data section contains the actual article. (This version of AppleSearch can read and display text articles only, although other types—such as an image—can be added to an update if your application chooses to support them.)

Article header

The article header is defined in the ASUArticle.h file and contains the following seven fields:

`ArticleHeaderSize`

This field indicates the size of the article's header, which includes `ArticleHeaderSize`, `ArticleDataSize`, `ArticleType`, `ArticleDateTime`, `UserBytes`, `ArticleTitle`, and `ArticleSource`. This number changes depending on the lengths of ArticleTitle and ArticleSource strings.

`ArticleDataSize`

This field indicates the size of the article's actual data.

`ArticleType`

This field specifies the article type. For example, 'TEXT' (`kTextArticleItem`) indicates that the article data in the `ArticleData` field is simply a packed array of ASCII characters terminated by appropriate end of line character(s). If it is 'PICT' (`kPICTArticleItem`), then the ArticleData field contains a PICT image.

'CMPD' (`kCompoundArticleItem`) is a special type indicating that the article is a compound article made up of more than one data element. The compound article format is described in more detail later in the section "Compound Articles."

Article types are defined in the next section, "Article Type Constants."

(The AppleSearch version 1.0 client can only read and display text articles, although other article types can be used by your application.)

`ArticleDate`

This field contains the article's date and time. It is in the ASUArticleDate structure format, which contains six short integers indicating year, month, day, hour, minute, and second as shown in the following example. This structure is defined in the ASUTypes.h file. (Note that structures are word-aligned whenever possible).

```
struct ASUArticleDate {
     short     year;     // YYYY
     short     month;    // 1-12
     short     day;      // 1-31
     short     hour;     // 0-23
     short     minute;   // 0-59
     short     second;   // 0-59
};
```

`UserBytes`	An application can use this field for any purpose.
`ArticleTitle`	This field is a null-terminated string that contains the article's title. All articles, regardless of type, must have a title.
`ArticleSource`	This field is a null-terminated string. It contains the name of the information source from which this article came. If the name is not available, the first byte of the string should be 0.

Article type constants

The following constants—defined in the ASUConstants.h file—are to be used in the ArticleType field:

```
const ASUArticleType kTextArticleItem = 'TEXT';
const ASUArticleType kStyledTextArticleItem = 'STXT';
const ASUArticleType kPICTArticleItem = 'PICT';
const ASUArticleType kTIFFArticleItem = 'TIFF';
const ASUArticleType kQuickTimeArticleItem = 'QTIM';
const ASUArticleType kSoundArticleItem = ' SND';
const ASUArticleType kCompoundArticleItem = 'CMPD';
```

AppleSearch version 1.0 supports only text articles, which are packed arrays of ASCII characters terminated by appropriate end-of-line character(s).

Compound articles

A compound article is an article that contains more than one element (subarticle). For example, an article with a PICT image and accompanying text is a compound article. It has an article type of 'CMPD'. The data structure of a compound article data section and an example of its use follow.

Format

```
struct ASUCompoundArticle {
     unsigned long       numberOfSubArticles;
     ASUSubArticle []    subArticles;
struct ASUSubArticle {
     ASUDataSize         subArticleDataSize;
     ASUArticleType      subArticleType;
     Byte[]              subArticleData;
     };
```

Example

Here is what a compound article made up of a styled text, a PICT, and a QuickTime movie looks like as part of an update file. Its title is “Apple News” and its source is “Hot News.”

```
//Data container header
long 40201          // size of the data section (header +
data)
long 'ARTL'    // data container type is an article
long 0         // data container id - not used
long 0         // data container user ref con
-------------------------------------------------------------
//Compound article header
long 48                  // size of the article header
long 40153                    // size of article data
long 'CMPD'              // compound article type
ASUArticleTime
     ArticleDateTime     // date and time of this article
long 0                   // user bytes
CString
     “Apple News”        // title; null-terminated
CString
     “Hot News”          // source; null-terminated
-------------------------------------------------------------
//Compound article data section
long 3                   // number of subarticles
ASDataSize 125           // size of the styled text data
ASArticleType 'STXT'          // styled text type
Bytes[] data             // the styled text itself
ASDataSize 8000          // size of the PICT data
ASArticleType 'PICT'          // image type
Bytes[] data             // the PICT itself
ASDataSize 32000         // size of the QT movie data
ASArticleType 'QTIM'          // QuickTime movie type
Bytes[] data             // the QT movie itself
```

Notes

AppleSearch version 1.0 cannot read compound articles. It can only read and display text articles. Compound articles are simply ignored in an update document.

3 AppleSearch Utility Routines

AppleSearch provides routines for working with update files. They are divided into update utility routines, article routines, and data container routines.

File utility routines

The update utility routines provide a set of routines for creating, reading, editing, and writing update files.

Basic type definitions

The basic types are defined in the ASUTypes.h file and are as follows:

```
typedef unsigned long
               ASUOffset;          // relative offset from
the
                              // file ptr
typedef unsigned long
               ASUDataSize;   // data size
typedef unsigned long
               ASUCount;      // number of things
typedef ResType
               ASUType;       // update type
typedef ResType
               ASUArticleType;
                              // article type is a packed
                              // array
typedef ResType
               ASUUpdateType; // type of update file
typedef ResType
               ASUDCType;          // data container type
typedef ASUOffset
               ASUDCPtr       // ptr to data container
```

Update info data structure

The update info data structure is defined in the ASUUpdateUtilities.h file and has the following format:

```
struct ASUUpdateInfo {
     ASUType   updateType;
     long      updateVersion;
};
```

Routines

The following update utility routines are defined in the ASUUpdateUtilities.h file and are described in detail in the sections that follow:

```
OSErr ASUCreateUpdateFile( FSSpec* file, OSType creator,
      ASUType type, long version );
OSErr ASUOpenUpdateFile( FSSpec* file, short* refnum );
OSErr ASUCloseUpdateFile( short refnum );
```

```
OSErr ASUReadUpdateInfo( short refnum,
      ASUpdateInfo* header );
OSErr ASUWriteUpdateInfo( short refnum,
      ASUpdateInfo* header );
OSErr ASUCompactUpdateFile( short refnum );
```

Create update file

```
OSErr ASUCreateUpdateFile( FSSpec* file, OSType creator,
     ASUType type, long version )
```

This function creates a new update file, appends an 'FHDR' (`ASUDCTypeFileHeader`) data container for the file header and writes file information passed to the routine. The file format version is set to `kVersion1`, which is 1.

Parameters

`file`	`FSSpec` specifying the file to be created.
`creator`	Specifies the creator type for this file.
`type`	Specifies the update type.
`version`	A version number.

Returns

`noErr` or a file system error.

Open update file

```
OSErr ASUOpenUpdateFile( FSSpec* file, short* refnum )
```

This function opens an existing update file with read/write permission.

Parameters

`file`	`FSSpec` specifying the file to be opened.
`refnum`	A file reference number is returned in this parameter.

Returns

`noErr` or a file system error.

Notes

The reference number returned by this function is used in most of the other function calls.

This function sets the type of the update file to `kAppleSearchPartialUpdateFileType` before opening.

Close update file

```
OSErr ASUCloseUpdateFile( short refnum )
```

This function closes an already opened update file.

Parameters

`refnum`	File reference number of an update file to be closed.

Returns

`noErr` or a file system error.

Notes

This function sets the file type to `kAppleSearchUpdateFileType` after the file is closed.

Read update info

```
OSErr ASUReadUpdateInfo( short refnum,
     ASUUpdateInfo* header )
```

This function reads update information from the specified update file.

Parameters

`refnum`	File reference number of the file already opened with `ASUOpenUpdateFile` from which the data is to be read.
`header`	A pointer to the `ASUUpdateInfo` structure into which the data is to be read.

Returns

`noErr` or a file system error.

Notes

The parameter `header` must point to an already-allocated storage space of the size of `ASUUpdateInfo.`

Write update info

```
OSErr ASUWriteUpdateInfo( short refnum,
     ASUUpdateInfo* header)
```

This function writes update file information to a specified update file.

Parameters

`refnum`	File reference number of the update file already opened via `ASUOpenUpdateFile()` to which the data will be written.
`header`	A pointer to the `ASUUpdateInfo` structure from which the information is to be written to the update file.

Returns

`noErr` or a file system error.

Notes

The parameter `header` must point to an already-allocated storage space.

Compact update file

```
OSErr ASUCompactUpdateFile( short refnum )
```

This function compacts a file by removing unused data containers of the type 'FREE' (`ASUDCTypeFree`) in the file.

Parameters

`refnum`	File reference number of an update file to be compacted.

Returns

`noErr` or a file system error.

Article routines

The article routines read, write, and manage articles within an update file.

Article date/time data structure

The article's date/time data structure is defined in the ASUArticleUtilities.h file and has the following format:

```
struct ASUArticleDate {
     short          year;          // YYYY
     short          month;    // 1-12
     short          day;      // 1-31
     short          hour;          // 0-24
     short          minute;   // 0-60
     short          second;   // 0-60
};
```

Article header data structure

The article header data structure is defined in the ASUArticleUtilities.h file and has the following format:

```
struct ASUArticleHeader {
     ASUDataSize         articleHeaderSize;
     ASUDataSize         articleDataSize;
     ASUArticleType      articleType;
     ASUArticleDate      articleDateTime;
     long                userBytes;
     char*                    articleTitle;
     char*                    articleSource;
};
```

Depending on the article routine called, pointers may need to be freed or allocated by the calling program. Any function that takes `ASUArticleHeader*` as an argument is likely to be subject to this restriction.

Routines

The following article routines are defined in the ASUArticleUtilities.h file and described in detail in the sections that follow:

```
OSErr ASUCountArticles( short refnum, ASUCount* count );
OSErr ASUFindArticle( short refnum, ASUDCPtr start,
      ASUDCPtr *dc );
OSErr ASUReadArticleHeader( short refnum, ASUDCPtr dc,
      ASUArticleHeader* header );
OSErr ASUReadArticleData( short refnum, ASUDCPtr dc,
      Ptr buffer );
```

```
OSErr ASUReadPartialArticleData( short refnum,
      ASUDCPtr dc, ASUOffset start, ASUDataSize* count,
      Ptr buffer );
OSErr ASUWriteArticleHeader( short refnum, ASUDCPtr dc,
      ASUArticleHeader* header );
OSErr ASUWriteArticleData( short refnum, ASUDCPtr dc,
      ASUOffset start, ASUDataSize count, Ptr buffer );
OSErr ASUAddArticle( short refnum,
      ASUArticleHeader* header, ASUDataSize count,
      Ptr buffer, ASUDCPtr *dc );
OSErr ASUDeleteArticle( short refnum, ASUDCPtr dc );
OSErr ASUGetArticleSize( short refnum, ASUDCPtr dc,
      ASUDataSize* size );
OSErr ASUGetArticleType( short refnum, ASUDCPtr dc,
      ASUArticleType* type );
OSErr ASUGetArticleUserBytes( short refnum, ASUDCPtr dc,
      long* userBytes );
OSErr ASUDisposeArticleHeader( ASUArticleHeader* header );
```

Count articles

```
OSErr ASUCountArticles( short refnum, ASUCount* count )
```

This function returns the number of articles in a specified update file.

Parameters

`refnum`	A file reference number of an already opened update file.
`count`	The number of articles in this file is returned in this parameter.

Returns

`noErr` or a file system error.

Find article

```
OSErr ASUFindArticle( short refnum, ASUDCPtr start,
     ASUDCPtr *dc )
```

Given a pointer to a data container within a file, this function finds the next data container of type 'ARTL' (`ASUDataContainerTypeArticle`) in a file. The `start` parameter must point to a data container. Zero is returned in `dc` if no more articles are found.

Parameters

`refnum`	A file reference number of an already opened update file.
`start`	Points to the data container from which the search starts.
`dc`	A pointer to a data container is returned in this parameter if an article is found.

Returns

`noErr` or a file system error.

Read article header

```
OSErr ASUReadArticleHeader(short refnum, ASUDCPtr dc,
     ASUArticleHeader* header )
```

This function reads in the header information of a specified article. The `dc` parameter must point to a data container of type 'ARTL' (`ASUDataContainerTypeArticle`).

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	A pointer to the article whose header is to be returned.
`header`	A pointer to an `ASUArticleHeader` structure. The routine fills the structure if the header information is found and read successfully.

Returns

`noErr` or a file system error.

`kOutOfMemoryErr  2001`

Notes

This function allocates a storage space for `articleTitle` and `articleSource` strings of the `ASUArticleHeader` structure if these strings are present. Your calling function must de-allocate the memory using `ASUDisposeArticleHeader()` when the calling function is done with the memory.

Read article data

```
OSErr ASUReadArticleData( short refnum, ASUDCPtr dc,
     Ptr buffer )
```

This function reads in the actual data portion (minus header information) of a specified article. The `dc` parameter must point to a data container of type 'ARTL' (`ASUDataContainerTypeArticle`).

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	A pointer to the article whose data is to be read.
`buffer`	A pointer to a buffer in which to store the article data.

Returns

`noErr` or a file system error.

Notes

It is assumed that the `buffer` pointer points to an already-allocated storage space large enough to hold the article data. The caller must take care of allocating and freeing the buffer.

Read partial article data

```
OSErr ASUReadPartialArticleData( short refnum, ASUDCPtr dc,
     ASUOffset start, ASUDataSize* count, Ptr buffer )
```

This function reads in part of the data portion (minus header information) of a specified article. The `dc` parameter must point to a data container of type 'ARTL.' (`ASUDataContainerTypeArticle`)

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	A pointer to the article whose data is to be read.
`start`	Place to begin reading, expressed as a byte offset from the beginning of the file.
`count`	Number of bytes to read from `start`.
`buffer`	A pointer to a buffer in which to store the article data.

Returns

`noErr` or a file system error.

Notes

It is assumed that the `buffer` pointer points to an already-allocated storage space equal to at least the byte size of `count`.

Write article header

```
OSErr ASUWriteArticleHeader( short refnum, ASUDCPtr dc,
     ASUArticleHeader* header )
```

This function writes the header information of a data container to a specified article. The `dc` parameter must point to a data container of type 'ARTL' (`ASUDataContainerTypeArticle`).

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	A pointer to the article to which the header information is written.
`header`	A pointer to the `ASUArticleHeader` structure containing the header information to write.

Returns

`noErr` or a file system error.

Notes

The caller must allocate and free the `ASUArticleHeader`, including the two strings it might contain.

Write article data

```
OSErr ASUWriteArticleData( short refnum, ASUDCPtr dc,
     ASUOffset start, ASUDataSize count, Ptr buffer );
```

This function writes the actual data to a specified article. The `dc` parameter must point to a data container of type 'ARTL' (`ASUDCTypeArticle`).

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	A pointer to the article to which the data is written.

`start`	An offset from the start of the data from which to begin writing.
`count`	Number of bytes to write.
`buffer`	A pointer to a buffer that holds the data to be written.

Returns

`noErr` or a file system error.

Add article

```
OSErr ASUAddArticle( short refnum,
     ASUArticleHeader* header, ASUDataSize count,
     Ptr buffer, ASUDCPtr *dc )
```

This function adds a new data container of the type 'ARTL' (`ASUDCTypeArticle`) to an update file.

Parameters

`refnum`	A file reference number of an already opened update file.
`header`	A pointer to the `ASUArticleHeader` structure.
`count`	Size of article data (in bytes).
`buffer`	A pointer to a buffer that holds the article to be added.
`dc`	A pointer to the beginning of the newly added data container of the type 'ARTL' is returned in this parameter.

Returns

`noErr` or a file system error.

Notes

This routine uses the value stored in `ASUArticleHeader.articleDataSize` to allocate space for the article data. If the `count` parameter is greater than 0, then that number of bytes from the buffer is written to the allocated article
data space.

You can pass a 0 `count` parameter and the `ASUWriteArticleData` function to add the actual article data later, but you must allocate the space in `ASUAddArticle()`.

The caller must allocate and free the buffer, the header, and the strings inside the header.

Delete article

```
OSErr ASUDeleteArticle( short refnum, ASUDCPtr dc )
```

This function deletes a data container of type 'ARTL' (`ASUDCTypeArticle`) pointed to by the `dc` parameter.

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	The file offset of the beginning of the data container of type 'ARTL' which is to be deleted.

Returns

`noErr` or a file system error.

Notes

This routine does not physically delete the article. Instead, it sets the article's data container type to 'FREE' (`ASUDCTypeFree`). The 'FREE' data containers are removed from the file when `ASUCompactUpdateFile` is called.

Get article size

```
OSErr ASUGetArticleSize( short refnum, ASUDCPtr dc,
     ASUDataSize* size )
```

This function returns the size of an article pointed to by the `dc` parameter.

Parameters

`refnum`	Reference number of an already opened update file.
`dc`	Pointer to a data container of type 'ARTL' (`ASUDCTypeArticle`).
`size`	Pointer to the `ASUDataSize` variable in which the article size is stored.

Returns

`noErr` or a file system error.

Get article type

```
OSErr ASUGetArticleType( short refnum, ASUDCPtr dc,
     ASUArticleType* type )
```

This function returns the type of article pointed to by the `dc` parameter.

Parameters

`refnum`	Reference number of an already opened update file.
`dc`	Pointer to a data container of type 'ARTL' (`ASUDCTypeArticle`).
`type`	Pointer to the `ASUArticleType` variable into which the article type is stored.

Returns

`noErr` or a file system error.

Get article user bytes

```
OSErr ASUGetArticleUserBytes( short refnum, ASUDCPtr dc,
     long* userBytes )
```

This function returns the “user bytes” of an article’s header.

Parameters

`refnum`	Reference number of an already opened update file.
`dc`	Pointer to a data container of type 'ARTL' (`ASUDCTypeArticle`).
`userBytes`	Pointer to a long integer variable into which the article header’s user bytes are stored.

Returns

`noErr` or a file system error.

Dispose article header

```
OSErr ASUDisposeArticleHeader( ASUArticleHeader* header )
```

This function disposes of the `ASUArticleHeader` data structure.

Parameters

`header`	A pointer to an `ASUArticleHeader` structure to be disposed.

Returns

`noErr` or a file system error.

Note

This routine must be called to dispose of the `ASUArticleHeader` returned by `ASUReadArticleHeader()`.

Data container routines

The data container routines define and manipulate data containers within an update file.

Data container info data structure

The data container info data structure is defined in the ASUDataContainerUtilities.h file and has the following format:

```
struct ASUDCHeader {
     ASUDataSize    dataContainerSize;
     ASUDCType      dataContainerType;
     long           dataContainerID;
     long           userRefCon;
};
```

Routines

The following data container routines are defined in the ASUDataContainerUtilities.h file and described in detail in the sections that follow:

```
OSErr ASUAddDataContainer( short refnum,
      ASUDCHeaderPtr header, ASUCount count, Ptr data,
      ASUDCPtr *dc );
OSErr ASUDeleteDataContainer( short refnum, ASUDCPtr dc );
OSErr ASUReadDataContainerHeader( short refnum,
      ASUDCPtr dc, ASUDCHeaderPtr header );
OSErr ASUWriteDataContainerHeader( short refnum,
      ASUDCPtr dc, ASUDCHeaderPtr header );
OSErr ASUReadDataContainerData( short refnum, ASUDCPtr dc,
      ASUOffset start, Ptr data, ASUDataSize* size );
OSErr ASUWriteDataContainerData( short refnum,
      ASUDCPtr dc, ASUOffset start, Ptr data,
      ASUDataSize size );
OSErr ASUCountDataContainers( short refnum,
      ASUDCType type, ASUCount* count );
OSErr ASUFindDataContainer( short refnum, ASUDCType type,
      ASUDCPtr start, ASUDCPtr *dc );
OSErr ASUGetDataContainerSize( short refnum, ASUDCPtr dc,
      ASUDataSize* size );
OSErr ASUGetDataContainerType( short refnum, ASUDCPtr dc,
      ASUDCType* type );
OSErr ASUGetDataContainerID( short refnum, ASUDCPtr dc,
      long* id );
OSErr ASUGetDataContainerUserRefCon( short refnum,
      ASUDCPtr dc, long* userRefCon );
```

Add data container

```
OSErr ASUAddDataContainer( short refnum,
     ASUDCHeaderPtr header, ASUCount count, Ptr data,
     ASUDCPtr *dc )
```

This function appends a new data container to the end of an update file. If the `data` parameter is not 0, it reads the data from that pointer and writes it to the data section of the newly added data container. (See Chapter 1, “Data Containers,” for more information on the structure of a data container.) The `count` parameter determines the number of bytes to read from the `data` and write to the container.

Parameters

`refnum`	File reference number of an already opened update file.
`header`	Pointer to the `ASUDCInfo` structure, which is a data container header. The `dataContainerSize` field is used to allocate an appropriate space for this container. The `count` parameter, however, is used to determine the actual number of bytes read from the `data` pointer and written to the data section of the container.
`count`	Number of bytes to write (read from the `data` pointer).
`data`	Pointer to data to be written into the data section of the container. If it is 0, then nothing is written.
`dc`	Returns a pointer to the beginning of the newly created data container.

Returns

`noErr` or a file system error.

Delete data container

```
OSErr ASUDeleteDataContainer( short refnum, ASUDCPtr dc )
```

This function “deletes” a data container by setting its type to 'FREE' (`ASUDCTypeFree`).

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	A pointer to the data container to be deleted.

Returns

`noErr` or a file system error.

Notes

The data container is marked as 'FREE' (`ASUDCTypeFree`) and it is not physically removed from the file until `ASUCompactUpdateFile()` is called.

Read data container header

```
OSErr ASUReadDataContainerHeader ( short refnum,
     ASUDCPtr dc, ASUDCHeaderPtr header )
```

This function reads the data container header of a specified data container into an `ASUDCHeader` structure. (See Chapter 1, “Data Containers,” for more information on the structure of a data container.)

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	Pointer to a data container from which the header is read.
`header`	A pointer to an `ASUDCHeader` structure into which the header information is read.

Returns

`noErr` or a file system error.

Write data container header

```
OSErr ASUWriteDataContainerHeader( short refnum,
     ASUDCPtr dc, ASUDataContainerHeader* header )
```

This function writes the header information into the data container header of the specified data container. (See Chapter 1, “Data Containers,” for more information on the structure of a data container.)

Parameters

`refnum`	A file reference number of an already opened update file.
`dc`	A pointer to the data container to which the header information is written.
`header`	A pointer to the `ASUDataContainerHeader` structure.

Returns

`noErr` or a file system error.

Read data container data

```
OSErr ASUReadDataContainerData( short refnum, ASUDCPtr dc,
     ASUOffset start, Ptr data, ASUDataSize* size )
```

This function reads the contents of a data container's data section into a specified buffer. (See Chapter 1, "Data Containers," for more information on the structure of a data container.)

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	A pointer to a data container from which this data is read.
`start`	Byte offset from the beginning of the data section from which point the data is read.
`data`	Pointer to a buffer into which data is read.
`size`	Number of bytes to be read.

Returns

`noErr` or a file system error.

Write data container data

```
OSErr ASUWriteDataContainerData( short refnum, ASUDCPtr
dc,
     ASUOffset start, Ptr data, ASUDataSize size )
```

This function writes the specified number of bytes to the data section of a data container. Data can be written at any offset within the data section. (See Chapter 1, "Data Containers," for more information on the structure of a data container.)

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	Pointer to a data container to which this data is written.
`start`	Offset from the beginning of the data container's data section to begin writing the data.
`data`	Pointer to the data to be written.
`size`	Number of bytes to be written.

Returns

`noErr` or a file system error.

Note

The caller is responsible for allocating and freeing `data`. The `data` must be at least `size` bytes large.

Count data containers

```
OSErr ASUCountDataContainers( short refnum, ASUDCType type,
     ASUCount* count)
```

This function counts the number of data containers of a specified type within a file.

Parameters

`refnum`	File reference number of an already opened update file.
`type`	The data container type to be counted.
`count`	The number of data containers matching the `type` is returned in this parameter.

Returns

`noErr` or a file system error.

Find data container

```
OSErr ASUFindDataContainer( short refnum, ASUDCType type,
     ASUDCPtr start, ASUDCPtr *dc )
```

This function finds the next occurrence of a data container of the specified type and returns a pointer to it.

Parameters

`refnum`	File reference number of an already opened update file.
`type`	The data container type to be found.
`start`	Pointer to the place to begin searching. It must point to a data container.
`dc`	Pointer to the next data container matching the `type` specified, if found.

Returns

`noErr` or a file system error.

Get data container size

```
OSErr ASUGetDataContainerSize ( short refnum, ASUDCPtr dc,
     ASUDataSize* size )
```

This function reads and returns the size of a specified data container.

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	Pointer to a data container from which the `size` is read.
`size`	The data container size is returned in this parameter.

Returns

`noErr` or a file system error.

Get data container type

```
OSErr ASUGetDataContainerType( short refnum, ASUDCPtr dc,
     ASUDCType* type )
```

This function reads and returns the type of the specified data container.

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	Pointer to a data container from which the `type` is read.
`type`	The data container type is returned in this parameter.

Returns

`noErr` or a file system error.

Get data container ID

```
OSErr ASUGetDataContainerID( short refnum, ASUDCPtr dc,
     long* id)
```

This function reads and returns the ID of a specified data container.

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	Pointer to a data container from which the ID is read.
`id`	The ID of the specified data container is returned in this parameter.

Returns

`noErr` or a file system error.

Get data container UserRefCon

```
OSErr ASUGetDataContainerUserRefCon( short refnum,
     ASUDCPtr dc, long* userRefCon )
```

This function reads and returns the `UserRefCon` of the specified data container.

Parameters

`refnum`	File reference number of an already opened update file.
`dc`	Pointer to a data container from which the `UserRefCon` is read.
`userRefCon`	The `UserRefCon` of the specified data container is returned in this field.

Returns

`noErr` or a file system error.