Search Kit Reference

User Experience

2010-03-24

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

Simultaneously published in the United States and Canada.

Contents

Tables

Search Kit Reference

Framework:	CoreServices/CoreServices.h
Declared in	SKAnalysis.h SKDocument.h SKIndex.h SKSearch.h SKSummary.h

Overview

Search Kit is a powerful and streamlined C language framework for indexing and searching text in most human languages. It provides fast information retrieval in System Preferences, Address Book, Help Viewer, and Xcode. Apple's Spotlight technology is built on top of Search Kit to provide content searching in Finder, Mail, and the Spotlight menu.

You can use Search Kit or Spotlight to provide similar functionality and powerful information-access capabilities within your Mac OS X application. Search Kit is appropriate when you want your application to have full control over indexing and searching, and when your focus is file content. Search Kit is thread-safe and works with Cocoa and command-line tools.

Beginning with Mac OS X version 10.4, Search Kit supports phrase searches, prefix/suffix/substring searches, improved Boolean searches, and improved relevance ranking. Search Kit now uses Spotlight's metadata importers when indexing documents, and takes advantage of any additional importers available on a system. Searching and indexing are much faster with Search Kit's new asynchronous search APIs. And, starting in Mac OS X v10.4, Search Kit provides a summarization API that supplants Find By Content.

Functions by Task

Functions are grouped according to the tasks you perform using them. For an alphabetical list of functions, go to the API index at the end of the document.

Creating, Opening, and Closing Indexes

Search Kit performs its searches not on documents but on its indexes of documents. The functions in this group let your application create memory-based and persistent indexes. Indexes are initially empty. Functions in "Managing Indexes" (page 10) let you add document content to these indexes.

`SKIndexCreateWithURL` (page 28)

Creates a named index in a file whose location is specified with a CFURL object.

`SKIndexCreateWithMutableData` (page 27)
Creates a named index stored in a CFMutableData object.

`SKIndexOpenWithData` (page 39)
Opens an existing, named index for searching only.

`SKIndexOpenWithMutableData` (page 39)
Opens an existing, named index for searching and updating.

`SKIndexOpenWithURL` (page 40)
Opens an existing, named index stored in a file whose location is specified with a CFURL object.

`SKIndexClose` (page 20)
Closes an index.

`SKIndexGetIndexType` (page 35)
Gets the category of an index.

`SKIndexGetTypeID` (page 38)
Gets the type identifier for Search Kit indexes.

Managing Indexes

The functions in this section let your application add document content to (and remove document content from) indexes, work with memory- and disk-based indexes, and retrieve metadata from indexes.

`SKIndexAddDocumentWithText` (page 19)
Adds a document URL object, and the associated document's textual content, to an index.

`SKIndexAddDocument` (page 18)
Adds location information for a file-based document, and the document's textual content, to an index.

`SKIndexFlush` (page 30)
Invokes all pending updates associated with an index and commits them to backing store.

`SKIndexCompact` (page 21)
Invokes all pending updates associated with an index, compacts the index if compaction is needed, and commits all changes to backing store.

`SKIndexGetDocumentCount` (page 32)
Gets the total number of documents represented in an index.

`SKIndexGetMaximumDocumentID` (page 36)
Gets the highest-numbered document ID in an index.

`SKIndexGetMaximumTermID` (page 36)
Gets the highest-numbered term ID in an index.

`SKIndexDocumentIteratorCreate` (page 29)
Creates an index-based iterator for document URL objects owned by a parent document URL object.

`SKIndexDocumentIteratorCopyNext` (page 29)
Obtains the next document URL object from an index using a document iterator.

`SKIndexDocumentIteratorGetTypeID` (page 30)
Gets the type identifier for Search Kit document iterators.

`SKIndexGetAnalysisProperties` (page 31)
Gets the text analysis properties of an index.

`SKIndexMoveDocument` (page 38)
Changes the parent of a document URL object in an index.

SKIndexRemoveDocument (page 41)
Removes a document URL object and its children, if any, from an index.

SKIndexRenameDocument (page 42)
Changes the name of a document URL object in an index.

SKIndexSetMaximumBytesBeforeFlush (page 43)
Not recommended. Sets the memory size limit for updates to an index, measured in bytes.

SKIndexGetMaximumBytesBeforeFlush (page 35)
Not recommended. Gets the memory size limit for updates to an index, measured in bytes.

Working With Text Importers

Search Kit can import the textual content of file-based documents into indexes using the Spotlight metadata importers.

SKLoadDefaultExtractorPlugIns (page 43)
Tells Search Kit to use the Spotlight metadata importers.

Working with Documents and Terms

From Search Kit's perspective, a document is anything that contains text—an RTF document, a PDF file, a Mail message, an Address Book entry, an Internet URL, the result of a database query, and so on.

The functions in this section let your application create new document URL objects (SKDocumentRefs), retrieve metadata from documents, get information on document hierarchies, and work with documents and their terms in the context of Search Kit indexes.

SKDocumentCreateWithURL (page 15)
Creates a document URL object from a CFURL object.

SKDocumentCreate (page 14)
Creates a document URL object based on a scheme, parent, and name.

SKDocumentCopyURL (page 14)
Builds a CFURL object from a document URL object.

SKDocumentGetName (page 16)
Gets the name of a document URL object.

SKDocumentGetParent (page 16)
Gets a document URL object's parent.

SKDocumentGetSchemeName (page 17)
Gets the scheme name for a document URL object.

SKDocumentGetTypeID (page 17)
Gets the type identifier for Search Kit document URL objects.

SKIndexCopyDocumentForDocumentID (page 21)
Obtains a document URL object from an index.

SKIndexCopyInfoForDocumentIDs (page 25)
Gets document names and parent IDs based on document IDs.

SKIndexCopyDocumentRefsForDocumentIDs (page 23)
Gets document URL objects based on document IDs.

`SKIndexCopyDocumentURLsForDocumentIDs` (page 24)
Gets document URLs based on document IDs.

`SKIndexCopyDocumentIDArrayForTermID` (page 22)
Obtains document IDs for documents that contain a given term.

`SKIndexCopyTermIDArrayForDocumentID` (page 25)
Obtains the IDs for the terms of an indexed document.

`SKIndexCopyTermStringForTermID` (page 26)
Obtains a term, specified by ID, from an index.

`SKIndexGetTermIDForTermString` (page 37)
Gets the ID for a term in an index.

`SKIndexSetDocumentProperties` (page 42)
Sets the application-defined properties of a document URL object.

`SKIndexCopyDocumentProperties` (page 22)
Obtains the application-defined properties of an indexed document.

`SKIndexGetDocumentState` (page 33)
Gets the current indexing state of a document URL object in an index.

`SKIndexGetDocumentTermCount` (page 34)
Gets the number of terms for a document in an index.

`SKIndexGetDocumentTermFrequency` (page 34)
Gets the number of occurrences of a term in a document.

`SKIndexGetTermDocumentCount` (page 37)
Gets the number of documents containing a given term represented in an index.

`SKIndexGetDocumentID` (page 32)
Gets the ID of a document URL object in an index.

Fast Asynchronous Searching

In Mac OS X v10.4 and later, Search Kit's fast asynchronous searching replaces synchronous searching. Synchronous searching, which relied on search groups, is deprecated.

`SKSearchCreate` (page 45)
Creates an asynchronous search object for querying an index, and initiates search.

`SKSearchFindMatches` (page 47)
Extracts search result information from a search object.

`SKSearchCancel` (page 44)
Cancels an asynchronous search request.

`SKSearchGetTypeID` (page 48)
Gets the type identifier for Search Kit search objects.

Working With Summarization

Search Kit's Summarization functions supplant those in Apple's Find by Content API.

`SKSummaryCreateWithString` (page 50)
Creates a summary object based on a text string.

`SKSummaryGetSentenceSummaryInfo` (page 53)
Gets detailed information about a body of text for constructing a custom sentence-based summary string.

`SKSummaryGetParagraphSummaryInfo` (page 51)
Gets detailed information about a body of text for constructing a custom paragraph-based summary string.

`SKSummaryGetSentenceCount` (page 52)
Gets the number of sentences in a summarization object.

`SKSummaryGetParagraphCount` (page 51)
Gets the number of paragraphs in a summarization object.

`SKSummaryCopySentenceAtIndex` (page 49)
Gets a specified sentence from the text in a summarization object.

`SKSummaryCopyParagraphAtIndex` (page 48)
Gets a specified paragraph from the text in a summarization object.

`SKSummaryCopySentenceSummaryString` (page 50)
Gets a text string consisting of a summary with, at most, the requested number of sentences.

`SKSummaryCopyParagraphSummaryString` (page 49)
Gets a text string consisting of a summary with, at most, the requested number of paragraphs.

`SKSummaryGetTypeID` (page 53)
Gets the type identifier for Search Kit summarization objects.

Legacy Support for Synchronous Searching

Developers should avoid using the functions listed in this section; instead, use the replacement functions that are recommended. Search Kit retains the functions in this section for backward compatibility.

`SKSearchGroupCopyIndexes` (page 65) Deprecated in Mac OS X v10.4
Obtains the indexes for a search group. (Deprecated. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.)

`SKSearchGroupCreate` (page 65) Deprecated in Mac OS X v10.4
Creates a search group as an array of references to indexes. (Deprecated. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.)

`SKSearchGroupGetTypeID` (page 66) Deprecated in Mac OS X v10.4
Deprecated. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.

`SKSearchResultsCopyMatchingTerms` (page 67) Deprecated in Mac OS X v10.4
Obtains the terms in a document that match a query. (Deprecated. Use `SKSearchCreate` (page 45) instead.)

`SKSearchResultsCreateWithDocuments` (page 67) Deprecated in Mac OS X v10.4
Finds documents similar to given example documents. (Deprecated. Use `SKSearchCreate` (page 45) instead.)

`SKSearchResultsCreateWithQuery` (page 69) Deprecated in Mac OS X v10.4
Queries the indexes in a search group. (Deprecated. Use `SKSearchCreate` (page 45) instead.)

`SKSearchResultsGetCount` (page 70) Deprecated in Mac OS X v10.4
Gets the total number of found items in a search. (Deprecated. Use `SKSearchCreate` (page 45) instead.)

`SKSearchResultsGetInfoInRange` (page 70) Deprecated in Mac OS X v10.4
: Extracts information from a Search Kit query result. (Deprecated. Use `SKSearchCreate` (page 45) instead.)

`SKSearchResultsGetTypeID` (page 71) Deprecated in Mac OS X v10.4
: Gets the type identifier for Search Kit search results. (Deprecated. Use `SKSearchCreate` (page 45) instead.)

Functions

SKDocumentCopyURL

Builds a CFURL object from a document URL object.

```
CFURLRef SKDocumentCopyURL (
    SKDocumentRef   inDocument
);
```

Parameters

inDocument
: The document URL object (SKDocumentRef) that you want a `CFURLRef` object for.

Return Value

A `CFURLRef` object representing a document location, or `NULL` on failure.

Discussion

You can use this function to create a *CFURL Reference* object to represent a document's location. Do this to gain access to the Core Foundation functionality provided by CFURL. This functionality includes accessing parts of the URL string, getting properties of the URL, and converting the URL to other representations.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SKDocument.h`

SKDocumentCreate

Creates a document URL object based on a scheme, parent, and name.

```
SKDocumentRef SKDocumentCreate (
CFStringRef     inScheme,
SKDocumentRef   inParent,
CFStringRef     inName
);
```

Parameters

inScheme

The scheme to use—analogous to the scheme of a URL. Only documents referenced with the "file" scheme can be read by the SKIndexAddDocument (page 18) function. The scheme can be anything you like if you use the SKIndexAddDocumentWithText (page 19) function. The scheme can be NULL, in which case it will be set to be the same scheme as the document URL object's (SKDocumentRef's) parent. For more information on schemes, see http://www.iana.org/assignments/uri-schemes.html.

inParent

The document URL object one step up in the document hierarchy. Can be NULL.

inName

The name of the document that you're creating a document URL object for. For the "file" scheme, it is the name of the file or the container, not its path. The path can be constructed by following parent links. The maximum length for a document name is 256 bytes.

Return Value

The new document URL object, or NULL on failure.

Discussion

The new document URL object's (SKDocumentRef's) parent can be NULL, but you must specify either a scheme or a parent. When your application no longer needs the document URL object, dispose of it by calling CFRelease.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

SeeMyFriends

Declared In

SKDocument.h

SKDocumentCreateWithURL

Creates a document URL object from a CFURL object.

```
SKDocumentRef SKDocumentCreateWithURL (
    CFURLRef    inURL
);
```

Parameters

inURL

The URL for the document URL object (SKDocumentRef) you are creating. The scheme of the document URL object gets set to the scheme of the URL used. Only URLs with a scheme of "file" can be used with the SKIndexAddDocument (page 18) function, but the URL scheme may be anything you like if you use the SKIndexAddDocumentWithText (page 19) function. For more information on schemes, see http://www.iana.org/assignments/uri-schemes.html.

Return Value
The new document URL object (SKDocumentRef), or `NULL` if the document URL object could not be created.

Discussion
Use `SKDocumentCreateWithURL` to create a unique reference to a file or to another, arbitrary URL that your application will use as a document URL object (SKDocumentRef). When your application no longer needs the document URL object, dispose of it by calling `CFRelease`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKDocument.h`

SKDocumentGetName

Gets the name of a document URL object.

```
CFStringRef SKDocumentGetName (
    SKDocumentRef   inDocument
);
```

Parameters

`inDocument`
The document URL object (SKDocumentRef) whose name you want to get.

Return Value
A `CFStringRef` object containing the document URL object's name, or `NULL` on failure.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKDocument.h`

SKDocumentGetParent

Gets a document URL object's parent.

```
SKDocumentRef SKDocumentGetParent (
    SKDocumentRef   inDocument
);
```

Parameters

`inDocument`
The document URL object (SKDocumentRef) whose parent you want to get.

Return Value
The parent document URL object, or `NULL` on failure.

Discussion
As described in `SKDocumentRef` (page 55), Search Kit manages document locations in terms of URLs as Document URL objects (SKDocumentRefs). The parent document URL object typically contains the document's URL up to but not including the document name.

Typically, document URL objects contain the complete URL to a file-based document. But you can use this function iteratively to build up the complete file-system path for a document that you are managing as part of a document hierarchy. See `SKDocumentRef` (page 55) for more on this.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKDocument.h`

SKDocumentGetSchemeName

Gets the scheme name for a document URL object.

```
CFStringRef SKDocumentGetSchemeName (
    SKDocumentRef   inDocument
);
```

Parameters

`inDocument`
The document URL object (SKDocumentRef) whose scheme you want to get.

Return Value
A `CFStringRef` object containing the document URL object's scheme name, or `NULL` on failure.

Discussion
The scheme of a document URL object (SKDocumentRef), which represents how it can be accessed, can be any character string but is typically "`file`" or "`http`". The scheme is one of a Search Kit document URL object's three properties—see `SKDocumentRef` (page 55) for details.

For more information on schemes, see http://www.iana.org/assignments/uri-schemes.html

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKDocument.h`

SKDocumentGetTypeID

Gets the type identifier for Search Kit document URL objects.

```
CFTypeID SKDocumentGetTypeID (void);
```

Return Value
A `CFTypeID` object containing the type identifier for the document URL object (SKDocumentRef).

Discussion
Search Kit represents document URL objects with the `SKDocumentRef` (page 55) opaque type. If your code needs to determine whether a particular data type is a document URL object, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Never hard-code the document URL object type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKDocument.h

SKIndexAddDocument

Adds location information for a file-based document, and the document's textual content, to an index.

```
Boolean SKIndexAddDocument (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument,
    CFStringRef     inMIMETypeHint,
    Boolean         inCanReplace
);
```

Parameters

inIndex

The index you are adding the document URL object to.

inDocument

The document URL object (SKDocumentRef), containing a file-based document's location information, to add to the index. You can release the document URL object immediately after adding it to the index.

inMIMETypeHint

The MIME type hint for the specified file-based document. Can be NULL. In Search Kit, common MIME type hints include text/plain, text/rtf, text/html, text/pdf, and application/msword.

Specify a MIME type hint to help Spotlight determine which of its metadata importers to use when Search Kit is indexing a file-based document. Search Kit uses filename extensions and type/creator codes in attempting to determine file types when indexing files. See SKLoadDefaultExtractorPlugIns (page 43). You can circumvent Search Kit's file type determination process, or override it, by using a MIME type hint.

inCanReplace

A Boolean value specifying whether Search Kit will overwrite a document's index entry (true, indicated by 1 or kCFBooleanTrue), or retain the entry if it exists (false, indicated by 0 or kCFBoolenFalse).

Return Value
A Boolean value of true on success, or false on failure. Also returns false if the document has an entry in the index and *inCanReplace* is set to false.

Discussion
The document scheme must be of type "file" to use this function. If it's not, call SKIndexAddDocumentWithText (page 19) instead. For more information on schemes, see http://www.iana.org/assignments/uri-schemes.html.

This function uses the referenced document and the optional MIME type hint to get the document's textual content using the Spotlight metadata importers. If you do not supply a MIME type hint, Spotlight's importers will use filename extensions and type/creator codes to guess file types.

Search Kit indexes any nonexecutable file associated with a document URL object (SKDocumentRef) that you hand to this function, even nontext files such as images. Your application takes responsibility for ensuring that the document URL objects you pass to SKIndexAddDocument are in fact the locations of files you want to index.

If your application did not call SKLoadDefaultExtractorPlugIns (page 43), Search Kit indexes the first 10 MB of a document. Otherwise, Search Kit indexes the entire document up to the index file size limit of 4 GB.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

A single Search Kit index can hold up to 4 billion document URL objects and their associated textual content.

Special Considerations

In the current implementation of Search Kit, some functions do not provide expected results unless you follow SKIndexAddDocument with a call to SKIndexFlush (page 30). The affected functions include SKIndexGetDocumentCount (page 32), SKIndexGetDocumentTermCount (page 34), SKIndexGetDocumentTermFrequency (page 34), and SKIndexGetTermDocumentCount (page 37). However, in typical use this won't be an issue, because applications call these functions after a search, and you must call SKIndexFlush before a search.

Version Notes

In versions of Mac OS X prior to Mac OS X v10.4, Search Kit used its own text extractor plug-ins rather than using the Spotlight metadata importers. See SKLoadDefaultExtractorPlugIns (page 43) and http://developer.apple.com/macosx/tiger/spotlight.html.

Availability

Available in Mac OS X v10.3 and later.

Declared In

SKIndex.h

SKIndexAddDocumentWithText

Adds a document URL object, and the associated document's textual content, to an index.

```
Boolean SKIndexAddDocumentWithText (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument,
    CFStringRef     inDocumentText,
    Boolean         inCanReplace
);
```

Parameters

inIndex

The index to which you are adding the document URL object (SKDocumentRef).

inDocument

The document URL object to add.

inDocumentText

The document text. Can be NULL.

inCanReplace

A Boolean value specifying whether Search Kit will overwrite a document's index entry (true, indicated by 1 or kCFBooleanTrue), or retain the entry if it exists (false, indicated by 0 or kCFBoolenFalse).

Return Value

A Boolean value of true on success, or false on failure. Also returns false if the document has an entry in the index and *inCanReplace* is set to false.

Discussion
Use this function to add the textual contents of arbitrary document types to an index. With this function, your application takes responsibility for getting textual content and handing it to the index as a CFString object. Because of this, your application can define what it considers to be a document—a database record, a tagged field in an XML document, an object in memory, a text file, and so on.

Search Kit will index any size text string that you give it, up to its 4 GB index file size limit.

To add the textual content of file-based documents to a Search Kit index, you can use this function or take advantage of Search Kit's ability to locate and read certain on-disk, file-based document types—see `SKIndexAddDocument` (page 18).

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

A single Search Kit index file can be up to 4 GB in size.

Special Considerations

In Mac OS X v10.3, some functions do not provide expected results unless you follow a call to `SKIndexAddDocumentWithText` with a call to `SKIndexFlush` (page 30). The affected functions include `SKIndexGetDocumentCount` (page 32), `SKIndexGetDocumentTermCount` (page 34), `SKIndexGetDocumentTermFrequency` (page 34), and `SKIndexGetTermDocumentCount` (page 37). However, in typical use this won't be an issue, because applications call these functions after a search, and you must call `SKIndexFlush` before a search.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
SeeMyFriends

Declared In
`SKIndex.h`

SKIndexClose

Closes an index.

```
void SKIndexClose (
    SKIndexRef      inIndex
);
```

Parameters

inIndex

The index to close.

Discussion
When your application no longer needs an index that it has opened or created, call `SKIndexClose`. Calling this function is equivalent to calling `CFRelease` on an index.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
SeeMyFriends

Declared In
`SKIndex.h`

SKIndexCompact

Invokes all pending updates associated with an index, compacts the index if compaction is needed, and commits all changes to backing store.

```
Boolean SKIndexCompact (
    SKIndexRef  inIndex
);
```

Parameters

`inIndex`

The index you want to compact.

Return Value
A Boolean value of `true` on success, or `false` on failure.

Discussion
Over time, as document URL objects (SKDocumentRefs) and associated contents get added to and removed from an index, the index's disk or memory footprint may grow due to fragmentation.

Compacting can take a significant amount of time. Do not call `SKIndexCompact` on the main thread in an application with a user interface. Call it only if the index is significantly fragmented and according to the needs of your application.

Calling `SKIndexCompact` changes the block allocation for an index's backing store. Close all clients of an index before calling this function.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexCopyDocumentForDocumentID

Obtains a document URL object from an index.

```
SKDocumentRef SKIndexCopyDocumentForDocumentID (
    SKIndexRef      inIndex,
    SKDocumentID    inDocumentID
);
```

Parameters

`inIndex`

The index containing the document URL object (SKDocumentRef).

`inDocumentID`

The ID of the document URL object you want to copy.

Return Value
A Search Kit document URL object.

Version Notes
In versions of Mac OS X prior to Mac OS X v10.4, the parameter type for *inDocumentID* was `CFIndex`. The parameter type in Mac OS X v10.4 and later is `SKDocumentID`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexCopyDocumentIDArrayForTermID

Obtains document IDs for documents that contain a given term.

```
CFArrayRef SKIndexCopyDocumentIDArrayForTermID (
    SKIndexRef      inIndex,
    CFIndex         inTermID
);
```

Parameters

inIndex
The index to search.

inTermID
The ID of the term to search for.

Return Value
An array of CFNumbers, each the ID for a document URL object that points to a document containing the search term.

Discussion
`SKIndexCopyDocumentIDArrayForTermID` searches a single index for documents that contain a given term. The search uses a term ID, not a term string. To get the ID of a term, use `SKIndexGetTermIDForTermString` (page 37).

Term IDs are index-specific; that is, a term has a different ID in each index in which it appears. If you want to search for all the documents containing a term in a set of indexes, call this function in turn for each index, using the index-specific term ID in each case.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexCopyDocumentProperties

Obtains the application-defined properties of an indexed document.

```
CFDictionaryRef SKIndexCopyDocumentProperties (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument
);
```

Parameters

inIndex

The index containing the document URL object (SKDocumentRef) whose properties you want to copy.

inDocument

The document URL object whose properties you want to copy.

Return Value

A CFDictionary object containing the document URL object's (SKDocumentRef's) properties, or `NULL` on failure.

Discussion

Search Kit document URL objects (SKDocumentRefs) can have an optional, application-defined properties dictionary to hold any information you'd like to associate with the document represented by a document URL object—such as timestamp, keywords, and so on. Use `SKIndexSetDocumentProperties` (page 42) to add a properties dictionary to a document URL object, and this function to obtain a copy of the dictionary.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SKIndex.h`

SKIndexCopyDocumentRefsForDocumentIDs

Gets document URL objects based on document IDs.

```
void SKIndexCopyDocumentRefsForDocumentIDs (
    SKIndexRef      inIndex,
    CFIndex         inCount,
    SKDocumentID    *inDocumentIDsArray,
    SKDocumentRef   *outDocumentRefsArray
);
```

Parameters

inIndex

The index containing the document information.

inCount

The number of document IDs in *inDocumentIDsArray*.

inDocumentIDsArray

Points to an array of document IDs corresponding to the document URL objects (SKDocumentRefs) you want.

outDocumentRefsArray

On input, a pointer to an array for document URL objects. On output, points to the previously allocated array, which now contains document URL objects corresponding to the document IDs in `inDocumentIDsArray`.

When finished with the document URL objects array, dispose of it by calling `CFRelease` on each array element.

Discussion
The `SKIndexCopyDocumentRefsForDocumentIDs` function lets you get a batch of document URL objects (SKDocumentRef objects) in one step, based on a list of document IDs.

If you want to get lightweight URLs in the form of CFURL objects instead, use `SKIndexCopyDocumentURLsForDocumentIDs` (page 24).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSearch.h`

SKIndexCopyDocumentURLsForDocumentIDs

Gets document URLs based on document IDs.

```
void SKIndexCopyDocumentURLsForDocumentIDs (
    SKIndexRef      inIndex,
    CFIndex         inCount,
    SKDocumentID    *inDocumentIDsArray,
    CFURLRef        *outDocumentURLsArray
);
```

Parameters

inIndex
The index containing the document information.

inCount
The number of document IDs in *inDocumentIDsArray*.

inDocumentIDsArray
Points to an array of document IDs corresponding to the document URLs (CFURL objects) you want.

outDocumentURLsArray
On input, a pointer to an array for document URLs (CFURL objects). On output, points to the previously allocated array, which now contains document URLs corresponding to the document IDs in `inDocumentIDArray`.

When finished with the document URL array, dispose of it by calling `CFRelease` on each array element.

Discussion
The `SKIndexCopyDocumentURLsForDocumentIDs` function lets you get a batch of document URLs (CFURL objects) in one step, based on a list of document IDs.

If you want to get Search Kit Document URL objects (SKDocumentRefs) instead, use `SKIndexCopyDocumentRefsForDocumentIDs` (page 23).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSearch.h`

SKIndexCopyInfoForDocumentIDs

Gets document names and parent IDs based on document IDs.

```
void SKIndexCopyInfoForDocumentIDs (
    SKIndexRef      inIndex,
    CFIndex         inCount,
    SKDocumentID    *inDocumentIDsArray,
    CFStringRef     *outNamesArray,
    SKDocumentID    *outParentIDsArray
);
```

Parameters

inIndex

The index containing the document information.

inCount

The number of document IDs in *inDocumentIDsArray*.

inDocumentIDsArray

Points to an array of document IDs representing the documents whose names and parent IDs you want.

outNamesArray

On input, a pointer to an array for document names. On output, points to the previously allocated array, which now contains the document names corresponding to the document IDs in *inDocumentIDsArray*. May be `NULL` on input if you don't want to get the document names.

When finished with the names array, dispose of it by calling `CFRelease` on each array element.

outParentIDsArray

On input, a pointer to an array for parent document IDs. On output, points to the previously allocated array, which now contains document IDs representing the parents of the documents whose IDs are in *inDocumentIDsArray*. May be `NULL` on input if you don't want to get the parent document IDs.

Discussion

The `SKIndexCopyInfoForDocumentIDs` function lets you get a batch of document names and parent document IDs in one step, based on a list of document IDs.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

SeeMyFriends

Declared In

`SKSearch.h`

SKIndexCopyTermIDArrayForDocumentID

Obtains the IDs for the terms of an indexed document.

```
CFArrayRef SKIndexCopyTermIDArrayForDocumentID (
    SKIndexRef      inIndex,
    SKDocumentID    inDocumentID
);
```

Parameters

inIndex

The index containing the document URL object (SKDocumentRef) and associated textual content.

inDocumentID

The ID of the document whose term IDs you are copying.

Return Value

A CFArray containing CFNumbers, each of which represents the ID for a term in a document.

Discussion

To derive the list of terms contained in a document, use this function to obtain an array of the term IDs, then convert each ID into the corresponding term with the `SKIndexCopyTermStringForTermID` (page 26) function.

Version Notes

In versions of Mac OS X prior to Mac OS X v10.4, the parameter type for `inDocumentID` was `CFIndex`. In Mac OS X v10.4 and later, the parameter type is `SKDocumentID`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SKIndex.h`

SKIndexCopyTermStringForTermID

Obtains a term, specified by ID, from an index.

```
CFStringRef SKIndexCopyTermStringForTermID (
    SKIndexRef      inIndex,
    CFIndex         inTermID
);
```

Parameters

inIndex

The index whose terms you are searching.

inTermID

The ID of the term whose string you want.

Return Value

A CFString containing the term specified by `inTermID`.

Discussion

When your application has the ID of a term, perhaps as a result of calling `SKIndexCopyTermIDArrayForDocumentID` (page 25), use this function to derive the term's text string.

To perform the inverse operation of deriving a term ID from a term string in a given index, use `SKIndexGetTermIDForTermString` (page 37).

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexCreateWithMutableData

Creates a named index stored in a CFMutableData object.

```
SKIndexRef SKIndexCreateWithMutableData (
    CFMutableDataRef    inData,
    CFStringRef         inIndexName,
    SKIndexType         inIndexType,
    CFDictionaryRef     inAnalysisProperties
);
```

Parameters

inData

An empty CFMutableData object to contain the index being created.

inIndexName

The name of the index. If you call this function with `inIndexName` set to `NULL`, Search Kit assigns the index the default index name `IADefaultIndex`. If you then attempt to create a second index in the same file without assigning a name, no second index is created and this function returns `NULL`. Search Kit does not currently support retrieving index names from an index.

inIndexType

The index type. See "SKIndexType" (page 62).

inAnalysisProperties

The text analysis properties dictionary, which optionally sets the minimum term length, stopwords, term substitutions, maximum unique terms to index, and proximity support (for phrase-based searches) when creating the index. See "Text Analysis Keys" (page 58). The `inAnalysisProperties` parameter can be `NULL`, in which case Search Kit applies the default dictionary, which is `NULL`.

Return Value
The newly created index.

Discussion
`SKIndexCreateWithMutableData` creates an index in memory as a CFMutableData object. Search Kit indexes are initially empty. A memory-based index is useful for quick searching and when your application doesn't need persistent storage. To create a disk-based, persistent index, use `SKIndexCreateWithURL` (page 28).

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

When your application no longer needs the index, dispose of it by calling `SKIndexClose` (page 20).

Special Considerations

You cannot use `CFMakeCollectable` with SKIndex objects.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
SeeMyFriends

Declared In
SKIndex.h

SKIndexCreateWithURL

Creates a named index in a file whose location is specified with a CFURL object.

```
SKIndexRef SKIndexCreateWithURL (
    CFURLRef            inURL,
    CFStringRef         inIndexName,
    SKIndexType         inIndexType,
    CFDictionaryRef     inAnalysisProperties
);
```

Parameters

inURL

The location of the index.

inIndexName

The name of the index. If you call this function with inIndexName set to NULL, Search Kit assigns the index the default index name IADefaultIndex. If you then attempt to create a second index in the same file without assigning a name, no second index is created and this function returns NULL. Search Kit does not currently support retrieving index names from an index.

inIndexType

The index type. See "SKIndexType" (page 62).

inAnalysisProperties

The text analysis properties dictionary, which optionally sets the minimum term length, stopwords, term substitutions, maximum unique terms to index, and proximity support (for phrase-based searches) when creating the index. See "Text Analysis Keys" (page 58). To get the analysis properties of an index, use the SKIndexGetAnalysisProperties (page 31) function. The inAnalysisProperties parameter can be NULL, in which case Search Kit applies the default dictionary, which is NULL.

Return Value
A unique reference to the newly created index.

Discussion
SKIndexCreateWithURL creates an index in a file. Search Kit indexes are initially empty. Use this function when your application needs persistent storage of an index. To create a memory-based, nonpersistent index, use SKIndexCreateWithMutableData (page 27).

A file can contain more than one index. To add a new index to an existing file, use the same value for inURL and supply a new name for inIndexName.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

When your application no longer needs the index, dispose of it by calling SKIndexClose (page 20).

Special Considerations

You cannot use CFMakeCollectable with SKIndex objects.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexDocumentIteratorCopyNext

Obtains the next document URL object from an index using a document iterator.

```
SKDocumentRef SKIndexDocumentIteratorCopyNext (
    SKIndexDocumentIteratorRef  inIterator
);
```

Parameters

inIterator

The index-based document iterator. See SKIndexDocumentIteratorCreate (page 29) for information on creating an document iterator, and SKIndexDocumentIteratorRef (page 55) for more about iterators.

Return Value
The next document URL object (SKDocumentRef) in the index.

Discussion
This function returns NULL when there are no more document URL objects (SKDocumentRefs) in the index. When finished iterating, your application must call CFRelease on all retrieved document URL objects that are non-NULL.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexDocumentIteratorCreate

Creates an index-based iterator for document URL objects owned by a parent document URL object.

```
SKIndexDocumentIteratorRef SKIndexDocumentIteratorCreate (
    SKIndexRef      inIndex,
    SKDocumentRef   inParentDocument
);
```

Parameters

inIndex

The index you want to iterate across.

inParentDocument

The document URL object (SKDocumentRef) that is the parent of the document URL objects you want to examine. Pass NULL to get the top item in an index. See SKDocumentRef (page 55) for a discussion of how to get the full URL for a document URL object.

Return Value
An index-based document iterator.

Discussion
When you want to iterate across all the documents represented in an index, use this function to create an iterator and then call `SKIndexDocumentIteratorCopyNext` (page 29) in turn for each document URL object (SKDocumentRef) in the index.

Document iterators iterate over a single level of an index. Your code is responsible for descending through a hierarchy of documents in an index.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

When your application no longer needs the iterator, dispose of it by calling `CFRelease`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexDocumentIteratorGetTypeID

Gets the type identifier for Search Kit document iterators.

```
CFTypeID SKIndexDocumentIteratorGetTypeID (void);
```

Return Value
A CFTypeID object containing the type identifier for the SKIndexDocumentIterator opaque type.

Discussion
Search Kit represents document iterators with the `SKIndexDocumentIteratorRef` (page 55) opaque type. If your code needs to determine whether a particular data type is a document iterator, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Never hard-code the document iterator type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexFlush

Invokes all pending updates associated with an index and commits them to backing store.

```
Boolean SKIndexFlush (
    SKIndexRef  inIndex
);
```

Parameters

inIndex

The index you want to update and commit to backing store.

Return Value
A Boolean value of `true` on success, or `false` on failure.

Discussion
An on-disk or memory-based index becomes stale when your application updates it by adding or removing a document entry. A search on an index in such a state won't have access to the nonflushed updates. The solution is to call this function before searching. `SKIndexFlush` flushes index-update information and commits memory-based index caches to disk, in the case of an on-disk index, or to a memory object, in the case of a memory-based index. In both cases, calling this function makes the state of the index consistent.

Before searching an index, always call `SKIndexFlush`, even though the flush process may take up to several seconds. If there are no updates to commit, a call to `SKIndexFlush` does nothing and takes minimal time.

A new Search Kit index does not have term IDs until it is flushed.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
SeeMyFriends

Declared In
`SKIndex.h`

SKIndexGetAnalysisProperties

Gets the text analysis properties of an index.

```
CFDictionaryRef SKIndexGetAnalysisProperties (
    SKIndexRef      inIndex
);
```

Parameters

inIndex

The index whose text-analysis properties you want to get.

Return Value
A CFDictionary object containing the index's text-analysis properties. On failure, returns `NULL`.

Discussion
The text analysis properties of an index determine how searches behave when querying the index. You set the analysis properties when creating an index with the `SKIndexCreateWithURL` (page 28) or `SKIndexCreateWithMutableData` (page 27) functions. For more information on text-analysis properties, see "`Text Analysis Keys`" (page 58).

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetDocumentCount

Gets the total number of documents represented in an index.

```
CFIndex SKIndexGetDocumentCount (
    SKIndexRef      inIndex
);
```

Parameters

inIndex

The index whose document URL objects (SKDocumentRefs) you want to count.

Return Value
A CFIndex object containing the number of document URL objects in the index. On failure, returns 0.

Discussion
Document URL objects (SKDocumentRefs) added to an index have an indexing state of kSKDocumentStateIndexed. See the "SKDocumentIndexState" (page 60) enumeration.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Special Considerations

In the current implementation of Search Kit, SKIndexGetDocumentCount returns the number of documents represented in the on-disk index. If your application has added document URL objects to the index but has not yet called SKIndexFlush (page 30), the document count may not be correct.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetDocumentID

Gets the ID of a document URL object in an index.

```
SKDocumentID SKIndexGetDocumentID (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument
);
```

Parameters

inIndex

The index containing the text of the document whose document URL object (SKDocumentRef) ID you want.

inDocument

The document URL object whose ID you want.

Return Value
A document ID object.

Discussion
The document ID identifies a document URL object (SKDocumentRef) in an index. The ID is available as soon as you add a document URL object to an index using `SKIndexAddDocumentWithText` (page 19) or `SKIndexAddDocument` (page 18).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexGetDocumentState

Gets the current indexing state of a document URL object in an index.

```
SKDocumentIndexState SKIndexGetDocumentState (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument
);
```

Parameters

inIndex

The index containing the document URL object (SKDocumentRef) whose indexing state you want.

inDocument

The document URL object whose indexing state you want.

Return Value
A value indicating the document URL object's indexing state.

Discussion
A document URL object (SKDocumentRef) can be in one of four states, as defined by the "`SKDocumentIndexState`" (page 60) enumeration: not indexed, indexed, not in the index but will be added after the index is flushed or closed, and in the index but will be deleted after the index is flushed or closed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexGetDocumentTermCount

Gets the number of terms for a document in an index.

```
CFIndex SKIndexGetDocumentTermCount (
    SKIndexRef      inIndex,
    SKDocumentID    inDocumentID
);
```

Parameters

inIndex

The index containing the text of the document whose term count you want.

inDocumentID

The ID of the document URL object (SKDocumentRef) whose term count you want. Obtain a document ID by calling `SKIndexGetDocumentID` (page 32).

Return Value

A CFIndex object containing the number of terms in a document.

Version Notes

versions of Mac OS X prior to Mac OS X v10.4, the parameter type for *inDocumentID* was `CFIndex`. In Mac OS X v10.4 and later, the parameter type is `SKDocumentID`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SKIndex.h`

SKIndexGetDocumentTermFrequency

Gets the number of occurrences of a term in a document.

```
CFIndex SKIndexGetDocumentTermFrequency (
    SKIndexRef      inIndex,
    SKDocumentID    inDocumentID,
    CFIndex         inTermID
);
```

Parameters

inIndex

The index containing the text of the document whose term count you are interested in.

inDocumentID

The ID of the document URL object whose associated term count you are interested in. Obtain a document ID by calling `SKIndexGetDocumentID` (page 32).

inTermID

The ID of the term whose number of occurrences you want.

Return Value

A CFIndex object containing the number of occurrences of a term in a document.

Version Notes

In versions of Mac OS X prior to Mac OS X v10.4, the parameter type for `inDocumentID` was `CFIndex`. In Mac OS X v10.4 and later, the parameter type is `SKDocumentID`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetIndexType

Gets the category of an index.

```
SKIndexType SKIndexGetIndexType (
    SKIndexRef      inIndex
);
```

Parameters

inIndex

The index whose category you want to know.

Return Value
The category of the index. See the "SKIndexType" (page 62) enumeration for a list of the various index categories. On failure, returns a value of kSKIndexUnknown.

Discussion
As described in "SKIndexType" (page 62), Search Kit offers four categories of index, each optimized for one or more types of searching.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetMaximumBytesBeforeFlush

Not recommended. Gets the memory size limit for updates to an index, measured in bytes.

```
CFIndex SKIndexGetMaximumBytesBeforeFlush (
    SKIndexRef      inIndex
);
```

Special Considerations

This function is rarely needed and is likely to be deprecated. Apple recommends using the SKIndexFlush (page 30) function along with the default memory size limit for index updates. Refer to the SKIndexSetMaximumBytesBeforeFlush function for more information.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetMaximumDocumentID

Gets the highest-numbered document ID in an index.

```
SKDocumentID SKIndexGetMaximumDocumentID (
    SKIndexRef  inIndex
);
```

Parameters

inIndex

An index.

Return Value

A document ID object containing the highest-numbered document ID in the index.

Discussion

Use this function with `SKIndexGetDocumentCount` (page 32) to determine whether an index is fragmented and in need of compaction. See `SKIndexCompact` (page 21).

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Version Notes

In versions of Mac OS X prior to Mac OS X v10.4, the return type for `SKIndexGetMaximumDocumentID` was `CFIndex`. The return type in Mac OS X v10.4 and later is `SKDocumentID`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SKIndex.h`

SKIndexGetMaximumTermID

Gets the highest-numbered term ID in an index.

```
CFIndex SKIndexGetMaximumTermID (
    SKIndexRef  inIndex
);
```

Parameters

inIndex

An index.

Return Value

A CFIndex object containing the highest-numbered term ID in an index.

Discussion

A new Search Kit index does not have term IDs until it is flushed.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability

Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetTermDocumentCount

Gets the number of documents containing a given term represented in an index.

```
CFIndex SKIndexGetTermDocumentCount (
    SKIndexRef      inIndex,
    CFIndex         inTermID
);
```

Parameters

inIndex

The index containing the text of the documents you want to examine.

inTermID

The terms whose occurrences you want to know.

Return Value
A CFIndex object containing the number of documents represented in an index that contain a given term.

Discussion
If you want to know in which documents a term appears across multiple indexes, call this function separately on each index. Before querying each index, get the index-specific term ID using `SKIndexGetTermIDForTermString` (page 37).

To ensure that this function takes into account document URL objects (SKDocumentRefs) recently added to indexes, call `SKIndexFlush` (page 30) on each index before calling this function.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexGetTermIDForTermString

Gets the ID for a term in an index.

```
CFIndex SKIndexGetTermIDForTermString (
    SKIndexRef      inIndex,
    CFStringRef     inTermString
);
```

Parameters

inIndex

The index you want to examine.

inTermString

The term string whose corresponding ID you want.

Return Value
A CFIndex object containing the term ID for a given term in an index. If the term isn't found, this function returns a value of `kCFNotFound`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexGetTypeID

Gets the type identifier for Search Kit indexes.

```
CFTypeID SKIndexGetTypeID (void);
```

Return Value
A CFTypeID object containing the type identifier for the SKIndex opaque type.

Discussion
Search Kit represents indexes with the `SKIndexRef` (page 56) opaque type. If your code needs to determine whether a particular data type is an index, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Never hard-code the index type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexMoveDocument

Changes the parent of a document URL object in an index.

```
Boolean SKIndexMoveDocument (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument,
    SKDocumentRef   inNewParent
);
```

Parameters

inIndex
The index containing the document URL object (SKDocumentRef) you want to move.

inDocument
The document URL object you want to move.

inNewParent
The new parent document URL object for the document URL object you want to move.

Return Value
A Boolean value of `true` for a successful move, or `false` on failure.

Discussion
When your application moves a document, use this function to update the index to reflect the change.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexOpenWithData

Opens an existing, named index for searching only.

```
SKIndexRef SKIndexOpenWithData (
    CFDataRef       inData,
    CFStringRef     inIndexName
);
```

Parameters

inData

The index to open.

inIndexName

The name of the index. Can be NULL, in which case this function attempts to open the index with the default name of IADefaultIndex.

Return Value
The named index, or NULL on failure.

Discussion
An index opened by SKIndexOpenWithData can be searched but not updated. To open an index for updating, use SKIndexOpenWithMutableData (page 39).

If inIndexName is NULL and inData does not contain an index with the default name, this function returns NULL.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

A call to SKIndexOpenWithData retains the opened index. When your application no longer needs the index, dispose of it by calling SKIndexClose (page 20).

Special Considerations

You cannot use CFMakeCollectable with SKIndex objects.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexOpenWithMutableData

Opens an existing, named index for searching and updating.

```
SKIndexRef SKIndexOpenWithMutableData (
    CFMutableDataRef    inData,
    CFStringRef         inIndexName
);
```

Parameters

inData

The index to open.

inIndexName

The name of the index. Can be `NULL`, in which case this function attempts to open the index with the default name of `IADefaultIndex`.

Return Value

The named index, or `NULL` on failure.

Discussion

An index opened by `SKIndexOpenWithMutableData` may be searched or updated. To open an index for search only, use the `SKIndexOpenWithData` (page 39) function.

If `inIndexName` is `NULL` and `inData` does not contain an index with the default name, this function returns `NULL`.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

A call to `SKIndexOpenWithMutableData` retains the opened index. When your application no longer needs the index, dispose of it by calling `SKIndexClose` (page 20).

Special Considerations

You cannot use `CFMakeCollectable` with SKIndex objects.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SKIndex.h`

SKIndexOpenWithURL

Opens an existing, named index stored in a file whose location is specified with a CFURL object.

```
SKIndexRef SKIndexOpenWithURL (
    CFURLRef        inURL,
    CFStringRef     inIndexName,
    Boolean         inWriteAccess
);
```

Parameters

inURL

The location of the index.

inIndexName

The name of the index. Can be `NULL`.

inWriteAccess

A Boolean value indicating whether the index is open for updating. To open an index for searching only, pass `false` (`0` or `kCFBoolenFalse`). To open it for searching and updating, pass `true` (`1` or `kCFBooleanTrue`).

Return Value
The named index.

Discussion
A call to `SKIndexOpenWithURL` retains the opened index. When your application no longer needs the index, dispose of it by calling `SKIndexClose` (page 20).

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Special Considerations

You cannot use `CFMakeCollectable` with SKIndex objects.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexRemoveDocument

Removes a document URL object and its children, if any, from an index.

```
Boolean SKIndexRemoveDocument (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument
);
```

Parameters

inIndex

The index from which you want to remove the document URL object (SKDocumentRef).

inDocument

The document URL object to remove.

Return Value
A Boolean value of `true` on success, or `false` on failure.

Discussion
When your application deletes a document, use this function to update the index to reflect the change.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
SeeMyFriends

Declared In
SKIndex.h

SKIndexRenameDocument

Changes the name of a document URL object in an index.

```
Boolean SKIndexRenameDocument (
    SKIndexRef      inIndex,
    SKDocumentRef   inDocument,
    CFStringRef     inNewName
);
```

Parameters

inIndex
The index containing the document URL object (SKDocumentRef) whose name you want to change.

inDocument
The document URL object whose name you want to change.

inNewName
The new name for the document URL object.

Return Value
A Boolean value of `true` if the document URL object name was successfully changed, or `false` on failure.

Discussion
When your application changes the name of a document, use this function to update the index to reflect the change.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SKIndex.h

SKIndexSetDocumentProperties

Sets the application-defined properties of a document URL object.

```
void SKIndexSetDocumentProperties (
    SKIndexRef          inIndex,
    SKDocumentRef       inDocument,
    CFDictionaryRef     inProperties
);
```

Parameters

inIndex
An index containing the document URL object (SKDocumentRef) whose properties you want to set.

inDocument
The document URL object whose properties you want to set.

inProperties

A CFDictionary object containing the properties to apply to the document URL object.

Discussion
Search Kit document URL objects (SKDocumentRefs) can have an optional, application-defined properties dictionary to hold any information you'd like to associate with the document represented by a document URL object—such as timestamp, keywords, and so on.

Use `SKIndexSetDocumentProperties` to persistently set application-defined properties for a document URL object in an index. This function replaces a document URL object's existing properties dictionary with the new one. To obtain a copy of a document URL object's properties dictionary, use `SKIndexCopyDocumentProperties` (page 22).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexSetMaximumBytesBeforeFlush

Not recommended. Sets the memory size limit for updates to an index, measured in bytes.

```
void SKIndexSetMaximumBytesBeforeFlush (
    SKIndexRef  inIndex
    CFIndex     inBytesForUpdate
);
```

Discussion
This function is rarely needed and is likely to be deprecated. Search Kit keeps track of index updates that are not yet committed to disk. Apple recommends using the default memory size limit for index updates, which is currently 2 million bytes.

Special Considerations

Apple recommends use of the `SKIndexFlush` (page 30) function instead of `SKIndexSetMaximumBytesBeforeFlush`.

Version Notes
In Mac OS X v10.3, the default memory size limit for index updates was 1 million bytes.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKLoadDefaultExtractorPlugIns

Tells Search Kit to use the Spotlight metadata importers.

```
void SKLoadDefaultExtractorPlugIns (void);
```

Discussion
The Spotlight metadata importers determine the `kMDItemTextContent` property for each document passed to the `SKIndexAddDocument` (page 18) function. See http://developer.apple.com/macosx/tiger/spotlight.html.

Call the `SKLoadDefaultExtractorPlugIns` function once at application launch to tell Search Kit to use the Spotlight metadata importers. The function `SKIndexAddDocument` (page 18) will then use Spotlight's importers to extract the text from supported files and place that text into an index, leaving the markup behind.

Version Notes
In versions of Mac OS X prior to Mac OS X v10.4, Search Kit used its own set of default text extractor plug-ins. The file types supported by Search Kit's default text extractor plug-ins were:

- plaintext
- PDF
- HTML
- RTF
- Microsoft Word (.doc)

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKSearchCancel

Cancels an asynchronous search request.

```
void SKSearchCancel (
    SKSearchRef                 inSearch
);
```

Parameters

inSearch
The search object whose associated asynchronous search you want to cancel.

Discussion
Call this function when you want to cancel an asynchronous search that you initiated with `SKSearchCreate` (page 45). This function stops the search process if it is still in progress at the time. It does not dispose of the search object (SKSearchRef).

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSearch.h`

SKSearchCreate

Creates an asynchronous search object for querying an index, and initiates search.

```
SKSearchRef SKSearchCreate (
    SKIndexRef              inIndex,
    CFStringRef             inQuery,
    SKSearchOptions         inSearchOptions
);
```

Parameters

inIndex

The index to query.

inQuery

The query string to search for.

inSearchOptions

The search options. May be `NULL`. See the "`SKSearchOptions`" (page 61) enumeration for a description of the available options.

Return Value

A search object.

Discussion

This function creates an asynchronous search object for querying the document contents in an index. It also initiates the search on a separate thread.

After you create the search object, call `SKSearchFindMatches` (page 47) to retrieve results. You can call `SKSearchFindMatches` immediately. To cancel a search, call `SKSearchCancel` (page 44).

For normal (non-similarity-based) queries, Search Kit discerns the type of query—Boolean, prefix, phrase, and so on—from the syntax of the query itself. Moreover, Search Kit supports multiple query types within a single search. For example, the following query includes Boolean, prefix, and suffix searching:

```
appl* OR *ing
```

This query will return documents containing words that begin with "appl" as well as documents that contain words that end with "ing".

For similarity searches, specified with the `kSKSearchOptionFindSimilar` flag in the *inSearchOptions* parameter, `SKSearchCreate` ignores all query operators.

The query operators that `SKSearchCreate` recognizes for non-similarity searching are:

Table 1 Search Kit query operators for non-similarity searches

Operator	meaning
`AND`	Boolean `AND`
`&`	Boolean `AND`
`<space>`	Boolean `AND` by default when no other operator is present, or Boolean `OR` if specified by `kSKSearchOptionSpaceMeansOR`.
`OR`	Boolean inclusive `OR`

Operator	meaning
|	Boolean inclusive OR
NOT	Boolean NOT (see Special Considerations)
!	Boolean NOT (see Special Considerations)
*	Wildcard for prefix or suffix; surround term with wildcard characters for substring search. Ignored in phrase searching.
(	Begin logical grouping
)	End logical grouping
"	delimiter for phrase searching

The operators AND, OR, and NOT are case sensitive.

Search Kit performs Unicode normalization on query strings and on the text placed into indexes. It uses Unicode Normalization Form KC (NFKC, compatibility decomposition followed by canonical composition) as documented in Unicode Standard Annex #15. For example, the a-grave character, 'à', can be written as the two Unicode characters (0x0061, 0x0300) or as the single Unicode character 0x00E0. Search Kit will normalize (0x0061, 0x0300) to 0x00E0. For more information on Unicode normalization, see http://unicode.org/reports/tr15 .

Search Kit further normalizes query strings and indexes by stripping diacritical marks and by forcing characters to lowercase. For example, Search Kit normalizes each of the following characters to 'a': 'a', 'à', 'A', and 'À'.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

When your application no longer needs the search object, dispose of it by calling CFRelease.

Special Considerations

Search Kit supports logical exclusion. The NOT and ! operators behave as though they were EXCLUDE operators. For example, a search for 'red NOT blue' returns all documents that contain the word 'red' and do not contain the word 'blue'.

Unary Boolean operators, however, are not currently implemented in Search Kit. A search, for example, for 'NOT blue', returns zero documents no matter what their content.

You cannot use CFMakeCollectable with SKSearch objects. In a garbage-collected environment, you must use CFRelease to dispose of an SKSearch object.

Version Notes

Mac OS X version 10.4 uses a completely revised, and far more powerful, query approach than did earlier versions of Mac OS X. Refer to the Discussion in this function for details. Refer to SKSearchResultsCreateWithQuery (page 69) (deprecated) for a description of Search Kit's behavior in earlier versions of Mac OS X.

In versions of Mac OS X prior to version 10.4, Search Kit did not perform Unicode normalization, and did not remove diacritical marks.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
SeeMyFriends

Declared In
`SKSearch.h`

SKSearchFindMatches

Extracts search result information from a search object.

```
Boolean SKSearchFindMatches (
    SKSearchRef         inSearch,
    CFIndex             inMaximumCount,
    SKDocumentID        *outDocumentIDsArray,
    float               *outScoresArray,
    CFTimeInterval      maximumTime
    CFIndex             *outFoundCount
);
```

Parameters

inSearch

A reference to a search object (SKSearchRef) previously created with `SKSearchCreate`.

inMaximumCount

The maximum number of items to find. For each item found, `SKSearchFindMatches` places the associated document ID into the *outDocumentIDsArray* array. Specify an *inMaximumCount* of 0 to find as many items as possible within *maximumTime*.

outDocumentIDsArray

On input, a pointer to an array for document IDs. On output, points to points to the previously allocated array, which now contains the found document IDs. The size of this array must be equal to *inMaximumCount*.

outScoresArray

On input, a pointer to an array for scores. On output, points to the previously allocated array, which now contains relevance scores for the found items. The size of this array, if not `NULL`, must be equal to *inMaximumCount*. Can be `NULL` on input, provided that your application doesn't need this information. Search Kit does not normalize relevance scores, so they can be very large.

maximumTime

The maximum number of seconds before this function returns, whether or not *inMaximumCount* items have been found. Setting maximumTime to 0 tells the search to return quickly

outFoundCount

On input, a pointer to a CFIndex object that will hold the number of items found. On output, points to the CFIndex object that now contains the actual number of items found.

Return Value
A logical value indicating whether the search is still in progress. Returns false when the search is exhausted.

Discussion
The `SKSearchFindMatches` extracts results from a find operation initiated by a search object (SKSearchRef).

This function provides results to its output parameters simply in the order in which they are found. This reduces latency to support search-as-you-type functionality. Larger scores mean greater relevance.

You can call this function on a search object repeatedly to get additional sets of search results. For example, if you call this function twice with an *inMaximumCount* value of 10, the first call will put the first 10 items found into the output arrays and the second call will put the second 10 items found into the output arrays.

Applications are free to display relevance scores in any appropriate manner. One simple way is to divide each relevance score by the largest number returned to get relevance numbers scaled linearly from 0.0 to 1.0. Search Kit does not scale the relevance scores for you, because you may want to combine the scores from several calls on a search object or the scores from calls to more than one search object.

Search Kit is thread-safe. You can use separate indexing and searching threads. Your application is responsible for ensuring that no more than one process is open at a time for writing to an index.

Before invoking a search, call `SKIndexFlush` (page 30) on all indexes you will query to ensure that updates to the indexes have been flushed to disk.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
SeeMyFriends

Declared In
`SKSearch.h`

SKSearchGetTypeID

Gets the type identifier for Search Kit search objects.

```
CFTypeID SKSearchGetTypeID (void);
```

Return Value
A CFTypeID object containing the type identifier for the SKSearch opaque type.

Discussion
Search Kit represents searches with search objects (`SKSearchRef` (page 56) opaque types). If your code needs to determine whether a particular data type is a search object, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Never hard-code the search type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSearch.h`

SKSummaryCopyParagraphAtIndex

Gets a specified paragraph from the text in a summarization object.

```
CFStringRef SKSummaryCopyParagraphAtIndex (
    SKSummaryRef        summary,
    CFIndex             i,
);
```

Parameters

summary

The summarization object containing the text from which you want a paragraph.

i

The ordinal number of the paragraph in the original text, with the first paragraph designated by zero (this function uses zero-based indexing).

Return Value
A CFString object containing the specified paragraph, or `NULL` on failure.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryCopyParagraphSummaryString

Gets a text string consisting of a summary with, at most, the requested number of paragraphs.

```
CFStringRef SKSummaryCopyParagraphSummaryString (
    SKSummaryRef        summary,
    CFIndex             numParagraphs
);
```

Parameters

summary

The summarization object containing the text from which you want a summarization.

numParagraphs

The maximum number of paragraphs you want in the summary.

Return Value
A CFString object containing the requested summary.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryCopySentenceAtIndex

Gets a specified sentence from the text in a summarization object.

```
CFStringRef SKSummaryCopySentenceAtIndex (
    SKSummaryRef        summary,
    CFIndex             i,
);
```

Parameters

summary

The summarization object containing the text from which you want a sentence.

i

The ordinal number of the sentence in the original text, with the first sentence designated by zero (this function uses zero-based indexing).

Return Value
A CFString object containing the specified sentence, or `NULL` on failure.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryCopySentenceSummaryString

Gets a text string consisting of a summary with, at most, the requested number of sentences.

```
CFStringRef SKSummaryCopySentenceSummaryString (
    SKSummaryRef        summary,
    CFIndex             numSentences
);
```

Parameters

summary

The summarization object containing the text from which you want a summarization.

numSentences

The maximum number of sentences you want in the summary.

Return Value
A CFString object containing the requested summary.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryCreateWithString

Creates a summary object based on a text string.

```
SKSummaryRef SKSummaryCreateWithString (
    CFStringRef     inString
);
```

Parameters

inString

The text string that you want to summarize.

Return Value
Returns a summarization object, or `NULL` on failure.

Discussion
The `SKSummaryCreateWithString` function creates a summarization object that pre-analyzes a text string to support fast summarization. When your application no longer needs the summarization object, dispose of it by calling `CFRelease`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryGetParagraphCount

Gets the number of paragraphs in a summarization object.

```
CFIndex SKSummaryGetParagraphCount (
    SKSummaryRef    summary
);
```

Parameters

summary

The summarization object whose paragraphs you want to count.

Return Value
A CFIndex object containing the number of paragraphs in the summarization object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryGetParagraphSummaryInfo

Gets detailed information about a body of text for constructing a custom paragraph-based summary string.

```
CFIndex SKSummaryGetParagraphSummaryInfo (
    SKSummaryRef            summary,
    CFIndex                 numParagraphsInSummary,
    CFIndex                 *outRankOrderOfParagraphs,
    CFIndex                 *outParagraphIndexOfParagraphs
);
```

Parameters

summary

The summarization object containing the text from which you want to build a summary.

numParagraphsInSummary

The maximum number of paragraphs you want in the summary.

outRankOrderOfParagraphs

On input, a pointer to an array of CFIndex objects. On output, points to the previously allocated array, which now lists the summarization relevance rank of each paragraph in the original text. The most important paragraph gets a rank of 1. The array size must equal *numParagraphsInSummary*, or else be `NULL` if you don't want to get the relevance ranks.

outParagraphIndexOfParagraphs

On output, points to an array containing the ordinal number for each paragraph in the original text. Use the `SKSummaryCopyParagraphAtIndex` (page 48) function with one of these numbers to get the corresponding paragraph. The array size must equal *numParagraphsInSummary*, or else be `NULL` if you don't want to get the ordinal numbers of the paragraphs.

Return Value
The number of paragraphs in the summary.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryGetSentenceCount

Gets the number of sentences in a summarization object.

```
CFIndex SKSummaryGetSentenceCount (
    SKSummaryRef    summary
);
```

Parameters

summary

The summarization object whose sentences you want to count.

Return Value
A CFIndex object containing the number of sentences in the summarization object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

SKSummaryGetSentenceSummaryInfo

Gets detailed information about a body of text for constructing a custom sentence-based summary string.

```
CFIndex SKSummaryGetSentenceSummaryInfo (
    SKSummaryRef            summary,
    CFIndex                 numSentencesInSummary,
    CFIndex                 *outRankOrderOfSentences,
    CFIndex                 *outSentenceIndexOfSentences,
    CFIndex                 *outParagraphIndexOfSentences
);
```

Parameters

summary

The summarization object containing the text from which you want to build a summary.

numSentencesInSummary

The maximum number of sentences you want in the summary.

outRankOrderOfSentences

On input, a pointer to an array of CFIndex objects. On output, points to the previously allocated array, which now lists the summarization relevance rank of each sentence in the original text. The most important sentence gets a rank of 1. The array size must equal *numSentencesInSummary*, or else be `NULL` if you don't want to get the rank orders.

outSentenceIndexOfSentences

On input, a pointer to an array of CFIndex objects. On output, points to the previously allocated array, which now contains the ordinal number for each sentence in the original text. Use the `SKSummaryCopySentenceAtIndex` (page 49) function with one of these numbers to get the corresponding sentence. The array size must equal *numSentencesInSummary*, or else be `NULL` if you don't want to get the ordinal numbers of the sentences.

outParagraphIndexOfSentences

On input, a pointer to an array of CFIndex objects. On output, points to the previously allocated array, which now contains the ordinal number for the paragraph that each corresponding sentence, referenced in *outSentenceIndexOfSentences*, appears in. The array size must equal *numSentencesInSummary*, or else be `NULL` if you don't want to get the ordinal numbers of the sentences.

Return Value

The number of sentences in the summary.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`SKSummary.h`

SKSummaryGetTypeID

Gets the type identifier for Search Kit summarization objects.

```
CFTypeID SKSummaryGetTypeID (void);
```

Return Value

A CFTypeID object, or `NULL` on failure.

Discussion
Search Kit represents summarization results with summarization objects (`SKSummaryRef` (page 57) opaque types). If your code needs to determine whether a particular data type is a summary, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Never hard-code the summarization type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SKSummary.h`

Callbacks

Developers should avoid using the callbacks listed in this section; instead, use `SKSearchCreate` (page 45) and `SKSearchFindMatches` (page 47).

SKSearchResultsFilterCallBack

Deprecated. Use `SKSearchCreate` and `SKSearchFindMatches` instead, which do not use a callback.

```
typedef Boolean (SKSearchResultsFilterCallBack) (
    SKIndexRef     inIndex,
    SKDocumentRef     inDocument,
    void      *inContext
```

If you name your function `MySearchResultsFilter`, you would declare it like this:

```
Boolean MySearchResultsFilter (
    SKIndexRef     inIndex,
    SKDocumentRef     inDocument,
    void      *inContext
);
```

Parameters

`inIndex`
The index you are searching.

`inDocument`
The document URL object within the index you are searching.

`inContext`
An application-specified context which you set when calling `SKSearchResultsCreateWithQuery` (page 69) or `SKSearchResultsCreateWithDocuments` (page 67).

Return Value
A Boolean value of `true` for a successful search hit, or `false` otherwise.

Discussion
Deprecated. Defines a pointer to a search-results filtering callback function for hit testing and processing during a search. Use this callback function to perform custom filtering on the search hits returned by the `SKSearchResultsCreateWithQuery` (page 69) and `SKSearchResultsCreateWithDocuments` (page 67) functions. Return `true` to keep this document URL object (SKDocumentRef) in the results, `false` to filter it out.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKSearch.h`

Data Types

SKDocumentRef

Defines an opaque data type representing a document's URL.

```
typedef struct __SKDocument *SKDocumentRef;
```

Discussion
A document URL object is a generic location specification for a document. It is built from a document scheme, a parent document, and a document name. You can convert back and forth between document URL objects and `CFURL` objects using Search Kit's `SKDocumentCreateWithURL` (page 15) and `SKDocumentCopyURL` (page 14) functions.

To create a Search Kit document URL object, use `SKDocumentCreateWithURL` (page 15) when you can provide a complete URL, or use `SKDocumentCreate` (page 14) when you want to specify document location indirectly using a parent document URL object. For other operations on documents, see "Working with Documents and Terms" (page 11).

If you create document URL objects with indirect locations using the `SKDocumentCreate` (page 14) function, you can resolve the locations by assembling them piece by piece, starting with a document URL object and going up step by step, parent to parent.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKDocument.h`

SKIndexDocumentIteratorRef

Defines an opaque data type representing an index-based document iterator.

```
typedef struct __SKIndexDocumentIterator *SKIndexDocumentIteratorRef;
```

Discussion
A Search Kit document iterator lets your application loop through all the document URL objects owned by a given parent document URL object. To create an iterator, use `SKIndexDocumentIteratorCreate` (page 29). To get a copy of the next document in the set owned by the iterator, use `SKIndexDocumentIteratorCopyNext` (page 29).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKIndexRef

Defines an opaque data type representing an index.

```
typedef struct __SKIndex *SKIndexRef;
```

Discussion
A Search Kit index object contains the textual contents of one or more documents, as well as document URL objects (SKDocumentRefs) representing those documents' locations.

To create a new disk-based Search Kit index object, use `SKIndexCreateWithURL` (page 28). To create a memory-based index, use `SKIndexCreateWithMutableData` (page 27). For other operations on indexes, see "Creating, Opening, and Closing Indexes" (page 9) and "Managing Indexes" (page 10). Also see"Fast Asynchronous Searching" (page 12).

Special Considerations

You cannot use `CFMakeCollectable` with SKIndex objects. In a garbage-collected environment, you must use `SKIndexClose` (page 20) to dispose of an SKIndex object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKIndex.h`

SKSearchRef

Defines an opaque data type representing a an asynchronous search.

```
typedef struct __SKSearch *SKSearchRef;
```

Discussion
A search object is created when you call the `SKSearchCreate` (page 45) function.

Special Considerations

You cannot use `CFMakeCollectable` with SKSearch objects. In a garbage-collected environment, you must use `CFRelease` to dispose of an SKSearch object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
SKSearch.h

SKSummaryRef

Defines an opaque data type representing summarization information.

```
typedef struct __SKSummary *SKSummaryRef;
```

Discussion
A summarization object contains summarization information, including summary text.

Availability
Available in Mac OS X v10.4 and later.

Declared In
SKSummary.h

SKDocumentID

Defines an opaque data type representing a lightweight document identifier.

```
typedef CFIndex SKDocumentID;
```

Discussion
Use document IDs rather than document URL objects (SKDocumentRefs) whenever possible. Using document IDs results in faster searching.

You can work with document IDs using a variety of Search Kit functions. See SKIndexGetMaximumDocumentID (page 36), SKIndexCopyDocumentForDocumentID (page 21), SKIndexCopyInfoForDocumentIDs (page 25), SKIndexCopyDocumentRefsForDocumentIDs (page 23), SKIndexCopyDocumentURLsForDocumentIDs (page 24), SKIndexCopyDocumentIDArrayForTermID (page 22), and SKIndexCopyTermIDArrayForDocumentID (page 25).

Availability
Available in Mac OS X v10.4 and later.

Declared In
SKIndex.h

SKSearchResultsRef

Deprecated. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.

```
typedef struct __SKSearchResults *SKSearchResultsRef;
```

Discussion
Defines an opaque data type representing the result of a search. To perform a query and generate search results, use `SKSearchResultsCreateWithQuery` (page 69) or `SKSearchResultsCreateWithDocuments` (page 67). To examine the result of a search, use `SKSearchResultsGetInfoInRange` (page 70). For other operations on search results, see "Legacy Support for Synchronous Searching" (page 13).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKSearch.h`

SKSearchGroupRef

Deprecated. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.

```
typedef struct __SKSearchGroup *SKSearchGroupRef;
```

Discussion
Defines an opaque data type representing a search group.

A search group is a group of one or more indexes to be searched. To create a search group, use `SKSearchGroupCreate` (page 65). For other operations with search groups, see "Fast Asynchronous Searching" (page 12).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SKSearch.h`

Constants

Text Analysis Keys

Each of these constants is an optional key in a Search Kit index's text analysis properties dictionary. The constant descriptions describe the corresponding values for each of these keys. These keys are declared in the `Analysis.h` header file.

```
const CFStringRef kSKMinTermLength;
const CFStringRef kSKStopWords;
const CFStringRef kSKSubstitutions;
const CFStringRef kSKMaximumTerms;
const CFStringRef kSKProximityIndexing;
const CFStringRef kSKTermChars;
const CFStringRef kSKStartTermChars;
const CFStringRef kSKEndTermChars;
```

Constants

`kSKMinTermLength`

The minimum term length to index. Specified as a CFNumber object. If this optional key is not present, Search Kit indexing defaults to a minimum term length of 1.

Available in Mac OS X v10.3 and later.

Declared in `SKAnalysis.h`.

`kSKStopWords`

A set of stopwords—words not to index. Specified as a CFSet object. There is no default stopword list. You must supply your own.

Available in Mac OS X v10.3 and later.

Declared in `SKAnalysis.h`.

`kSKSubstitutions`

A dictionary of term substitutions—terms that differ in their character strings but that match during a search. Specified as a CFDictionary object.

Available in Mac OS X v10.3 and later.

Declared in `SKAnalysis.h`.

`kSKMaximumTerms`

The maximum number of number unique terms to index in each document. Specified as a CFNumber object.

Search Kit indexes from the beginning of a document. When it has indexed the first n unique terms, it stops.

The default number of maximum terms, which applies if you do not provide a number, is 2000.

To tell Search Kit to index all the terms in each document without limit, specify a value of 0.

Available in Mac OS X v10.4 and later.

Declared in `SKAnalysis.h`.

`kSKProximityIndexing`

A Boolean flag indicating whether or not Search Kit should use proximity indexing. The flag can be a `0` or `kCFBoolenFalse` value (for false) or a `1` or `kCFBooleanTrue` value (for true). Proximity indexing is available only for inverted indexes—that is, indexes of type `kSKIndexInverted` (page 62).

Use proximity indexing to support phrase searching. If this key is not present in an index's text analysis properties dictionary, Search Kit defaults to not adding proximity information to the index.

Available in Mac OS X v10.4 and later.

Declared in `SKAnalysis.h`.

`kSKTermChars`

Additional valid starting-position "word" characters for indexing and querying. The corresponding value, a CFString object, specifies the additional valid "word" characters that you want to be considered as valid starting characters of terms for indexing and querying. "Word" characters are contrasted with nonword characters, such as spaces.

The value of `kSKStartTermChars`, if this key is present, overrides the value of `kSKTermChars` for the first character of a term.

By default, Search Kit considers alphanumeric characters as valid starting characters for terms, and considers all others (including the underscore character) to be nonword characters.

Available in Mac OS X v10.4 and later.

Declared in `SKAnalysis.h`.

`kSKStartTermChars`

Additional valid starting-position "word" characters for indexing and querying. The corresponding value, a CFString object, specifies the additional valid "word" characters that you want to be considered as valid starting characters of terms for indexing and querying. "Word" characters are contrasted with nonword characters, such as spaces.

The value of `kSKStartTermChars`, if this key is present, overrides the value of `kSKTermChars` for the first character of a term.

By default, Search Kit considers alphanumeric characters as valid starting characters for terms, and considers all others (including the underscore character) to be nonword characters.

Available in Mac OS X v10.4 and later.

Declared in `SKAnalysis.h`.

`kSKEndTermChars`

Additional valid last-position "word" characters for indexing and querying. The corresponding value, a CFString object, specifies the additional valid "word" characters that you want to be considered as valid ending characters of terms for indexing and querying. "Word" characters are contrasted with nonword characters, such as spaces.

The value of `kSKEndTermChars`, if this key is present, overrides the value of `kSKTermChars` for the last character of a term.

By default, Search Kit considers alphanumeric characters as valid ending characters for terms, and considers all others (including the underscore character) to be nonword characters.

Available in Mac OS X v10.4 and later.

Declared in `SKAnalysis.h`.

SKDocumentIndexState

The indexing state of a document.

```
enum SKDocumentIndexState {
    kSKDocumentStateNotIndexed  = 0,
    kSKDocumentStateIndexed     = 1,
    kSKDocumentStateAddPending  = 2,
    kSKDocumentStateDeletePending= 3
};
```

Constants

`kSKDocumentStateNotIndexed`

Specifies that the document is not indexed.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

`kSKDocumentStateIndexed`

Specifies that the document is indexed.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

`kSKDocumentStateAddPending`

Specifies that the document is not in the index but will be added after the index is flushed or closed.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

`kSKDocumentStateDeletePending`

Specifies that the document is in the index but will be deleted after the index is flushed or closed.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

Declared In

`SKIndex.h`

SKSearchOptions

Specifies the search options available for the `SKSearchCreate` (page 45) function.

```
typedef UInt32 SKSearchOptions;
enum SKSearchType {
    kSKSearchOptionDefault = 0,
    kSKSearchOptionNoRelevanceScores = 1L << 0,
    kSKSearchOptionSpaceMeansOR = 1L << 1
    kSKSearchOptionFindSimilar = 1L << 2
};
```

Constants

`kSKSearchOptionDefault`

Default search options include:

- Relevance scores will be computed
- Spaces in a query are interpreted as Boolean `AND` operators.
- Do not use similarity searching.

Available in Mac OS X v10.4 and later.

Declared in `SKSearch.h`.

`kSKSearchOptionNoRelevanceScores`
This option saves time during a search by suppressing the computation of relevance scores.

Available in Mac OS X v10.4 and later.

Declared in `SKSearch.h`.

`kSKSearchOptionSpaceMeansOR`
This option alters query behavior so that spaces are interpreted as Boolean `OR` operators.

Available in Mac OS X v10.4 and later.

Declared in `SKSearch.h`.

`kSKSearchOptionFindSimilar`
This option alters query behavior so that Search Kit returns references to documents that are similar to an example text string. When this option is specified, Search Kit ignores all query operators.

Available in Mac OS X v10.4 and later.

Declared in `SKSearch.h`.

Declared In
`SKSearch.h`

SKIndexType

Specifies the category of an index.

```
enum SKIndexType {
    kSKIndexUnknown         = 0,
    kSKIndexInverted        = 1,
    kSKIndexVector          = 2,
    kSKIndexInvertedVector  = 3
};
```

Constants
`kSKIndexUnknown`
Specifies an unknown index type.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

`kSKIndexInverted`
Specifies an inverted index, mapping terms to documents.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

`kSKIndexVector`
Specifies a vector index, mapping documents to terms.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

`kSKIndexInvertedVector`
Specifies an index type with all the capabilities of an inverted and a vector index.

Available in Mac OS X v10.3 and later.

Declared in `SKIndex.h`.

Declared In
`SKIndex.h`

Deprecated Text Analysis Keys

Search Kit ignores the `kSKLanguageTypes` constant. It determines language directly by document content.

```
const CFStringRef kSKLanguageTypes;
```

Constants

`kSKLanguageTypes`

Deprecated—Search Kit ignores this constant.

In releases of Mac OS X previous to version 10.4, each string in this key's corresponding value specifies a language to use for indexing. Each such string is a two character ISO 639-1 code. For example, `'en'` for English, `'ja'` for Japanese, and so on. If this key is not present, Search Kit uses the Mac OS X preferences system to determine the primary language from the user's locale.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared in `SKAnalysis.h`.

Version Notes

In releases of Mac OS X prior to version 10.4, the `kSKLanguageTypes` constant was an optional key in an index's text analysis properties dictionary. Starting in Mac OS X v10.4, Search Kit ignores this constant and determines language directly by the document content. A document may use multiple languages.

Deprecated Search Keys

Search Kit ignores the constants in this group. Use asynchronous searching with `SKSearchCreate` instead, which uses query syntax to determine search type.

```
enum SKSearchType {
    kSKSearchRanked         = 0,
    kSKSearchBooleanRanked  = 1,
    kSKSearchRequiredRanked = 2,
    kSKSearchPrefixRanked   = 3
};
```

Constants

`kSKSearchRanked`

Deprecated. Specifies a basic ranked search.

Available in Mac OS X v10.3 and later.

Declared in `SKSearch.h`.

`kSKSearchBooleanRanked`

Deprecated. Specifies a query that can include Boolean operators including `'|'`, `'&'`, `'!'`, `'('`, and `')'`.

Available in Mac OS X v10.3 and later.

Declared in `SKSearch.h`.

`kSKSearchRequiredRanked`

Deprecated. Specifies a query that can include required (`'+'`) or excluded (`'-'`) terms.

Available in Mac OS X v10.3 and later.

Declared in `SKSearch.h`.

`kSKSearchPrefixRanked`

Deprecated. Specifies a prefix-based search, which matches terms that begin with the query string.

Available in Mac OS X v10.3 and later.

Declared in `SKSearch.h`.

Version Notes

In releases of Mac OS X prior to version 10.4, these constants specify the category of search to perform. Starting with Mac OS X v10.4, use asynchronous searching with `SKSearchCreate` instead, which uses query syntax to determine search type.

In older versions of Mac OS X, these constants specify the various search types you can use with `SKSearchResultsCreateWithQuery`. Each of these specifies a set of ranked search hits. The `kSKSearchRanked` and `kSKSearchPrefixRanked` constants can be used for all index types. The `kSKSearchBooleanRanked` and `kSKSearchRequiredRanked` constants cannot be used for vector indexes.

Declared In

`SKSearch.h`

Deprecated Search Kit Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

SKSearchGroupCopyIndexes

Obtains the indexes for a search group. (Deprecated in Mac OS X v10.4. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.)

```
CFArrayRef SKSearchGroupCopyIndexes (
    SKSearchGroupRef    inSearchGroup
);
```

Parameters

inSearchGroup

The search group whose indexes you want to copy.

Return Value

A CFArray object containing the indexes in the search group.

Discussion

Although the search functions `SKSearchResultsCreateWithQuery` (page 69) and `SKSearchResultsCreateWithDocuments` (page 67) operate directly on search groups, many Search Kit functions, such as `SKIndexCompact` (page 21), operate on one index at a time. When you want to examine or manage all the indexes in a search group, use `SKSearchGroupCopyIndexes` to get the search group's list of indexes.

Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared In

`SKSearch.h`

SKSearchGroupCreate

Creates a search group as an array of references to indexes. (Deprecated in Mac OS X v10.4. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups.)

```
SKSearchGroupRef SKSearchGroupCreate (
    CFArrayRef      inArrayOfInIndexes
);
```

Parameters

inArrayOfInIndexes

A CFArray object containing the indexes to put into the search group.

Return Value
An SKSearchGroup opaque type.

Discussion
Creates a search group as an array of references to indexes.

You create a search group to search one or more indexes, and then typically use the resulting `SKSearchGroupRef` opaque type with `SKSearchResultsCreateWithQuery` (page 69) or `SKSearchResultsCreateWithDocuments` (page 67).

When your application no longer needs the search group, dispose of it by calling `CFRelease`.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
`SKSearch.h`

SKSearchGroupGetTypeID

Deprecated. Use asynchronous searching with SKSearchCreate instead, which does not employ search groups. (Deprecated in Mac OS X v10.4.)

```
CFTypeID SKSearchGroupGetTypeID (void);
```

Return Value
A CFTypeID object containing the type identifier for the SKSearchGroup opaque type.

Discussion
Gets the type identifier for Search Kit search groups.

Search Kit represents search groups with the `SKSearchGroupRef` (page 58) opaque type. If your code needs to determine whether a particular data type is a search group, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Never hard-code the search group type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
`SKSearch.h`

SKSearchResultsCopyMatchingTerms

Obtains the terms in a document that match a query. (Deprecated in Mac OS X v10.4. Use SKSearchCreate (page 45) instead.)

```
CFArrayRef SKSearchResultsCopyMatchingTerms (
    SKSearchResultsRef  inSearchResults,
    CFIndex             inItem
);
```

Parameters

inSearchResults

The search results to examine.

inItem

An integer that corresponds to a document URL object (SKDocumentRef) in the search results. A value of '1' identifies the first document URL object in the search results, a value of '2' identifies the second, and so on.

If you've created the search results using SKSearchResultsCreateWithQuery (page 69), the document URL objects are sorted in ranking order with the top-ranked one first. See SKSearchResultsGetInfoInRange (page 70) for a description of how to get a particular document URL object, or set of them, from a search result.

Return Value

A CFArray object containing term IDs.

Discussion

When using a prefix search, or a search for which the user entered more than one word, there may be multiple terms that match the query. This function returns an array of the term IDs corresponding to these matches.

For example, a user could enter 'App' when performing a prefix search. If a document represented in the search group contains the words 'Apple,' 'application,' and 'appendectomy,' the IDs for all of these terms would then appear in the CFArray object that SKSearchResultsCopyMatchingTerms returns.

See SKSearchResultsCreateWithQuery (page 69) for a description of how to perform a search and get search results. See SKSearchResultsGetInfoInRange (page 70) for how to extract information, including document URL objects, from a search result. See "Deprecated Search Keys" (page 63) for a description of the various categories of search.

Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared In

SKSearch.h

SKSearchResultsCreateWithDocuments

Finds documents similar to given example documents. (Deprecated in Mac OS X v10.4. Use SKSearchCreate (page 45) instead.)

```
SKSearchResultsRef SKSearchResultsCreateWithDocuments (
    SKSearchGroupRef                inSearchGroup,
    CFArrayRef                      inExampleDocuments,
    CFIndex                         inMaxFoundDocuments,
    void                            *inContext,
    SKSearchResultsFilterCallBack   inFilterCallBack
);
```

Parameters

inSearchGroup

A search group containing the indexes which, in turn, contain the document URL objects (SKDocumentRefs) representing the documents you want to search by similarity. The search group must also contains the indexes that contain the textual content of the example documents.

inExampleDocuments

An array of document URL objects (SKDocumentRefs), each representing an example document.

inMaxFoundDocuments

The maximum number of found items to return. Your application must pass in a positive value.

inContext

An application-specified context for use by the SKSearchResultsFilterCallBack (page 54) callback function. Can be NULL.

inFilterCallBack

A callback function for hit testing during searching—see SKSearchResultsFilterCallBack (page 54). In a similarity search, your application would typically use this function to exclude the example documents from the search results. This parameter can be NULL, in which case your application receives the returned results directly and without any custom postprocessing.

Return Value

A search results object containing a list of document URL objects (SKDocumentRefs) representing documents similar to the example documents.

Discussion

This function searches the on-disk indexes in a search group for document URL objects (SKDocumentRefs) representing documents similar to those provided as examples. Build the search group in three steps:

1. Collect the index IDs from the search groups you want to search: for each search group, call the SKSearchGroupCopyIndexes (page 65) function.
2. Add the document URL objects representing the example documents to a memory-based index (if they're not already in an index) by calling SKIndexCreateWithMutableData (page 27), and get that index's ID.
3. Create a new search group that contains the indexes to search, and also containing the example-documents index, using SKSearchGroupCreate (page 65).

Before invoking a search, call SKIndexFlush (page 30) on all indexes in the search group to ensure that changes to the indexes have been written to disk.

Once you've obtained the results of a search, get the specifics—including which documents match the user's similarity query, and the ranking scores for each document—by calling SKSearchResultsGetInfoInRange (page 70).

When your application no longer needs the search result, dispose of it by calling CFRelease.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
SKSearch.h

SKSearchResultsCreateWithQuery

Queries the indexes in a search group. (Deprecated in Mac OS X v10.4. Use SKSearchCreate (page 45) instead.)

```
SKSearchResultsRef SKSearchResultsCreateWithQuery (
    SKSearchGroupRef        inSearchGroup,
    CFStringRef             inQuery,
    SKSearchType            inSearchType,
    CFIndex                 inMaxFoundDocuments,
    void                    *inContext,
    SKSearchResultsFilterCallBackinFilterCallBack
);
```

Parameters

inSearchGroup
The search group to query.

inQuery
The query string to search for.

inSearchType
The category of search to perform. See the "Deprecated Search Keys" (page 63) enumeration for options.

inMaxFoundDocuments
The maximum number of found items to return. Your application must pass in a positive integer value.

inContext
An application-specified context for use by the SKSearchResultsFilterCallBack (page 54). Can be NULL, but if you want to use the callback you must supply a context.

inFilterCallBack
A callback function for hit testing during searching. Can be NULL, in which case your application receives the returned results directly and without any custom postprocessing. If non-NULL, you must supply a context. See SKSearchResultsFilterCallBack (page 54).

Return Value
A search results object.

Discussion
This function searches the on-disk indexes in a search group. Before invoking a search, call SKIndexFlush (page 30) on all indexes in the search group to ensure that changes to the indexes have been flushed to disk.

Once you've obtained the results of a search, get the specifics—including which documents match the user's query, and the ranking scores for each document—by calling SKSearchResultsGetInfoInRange (page 70). You can extract other information by calling SKSearchResultsCopyMatchingTerms (page 67) and SKSearchResultsGetCount (page 70).

When your application no longer needs the search result, dispose of it by calling `CFRelease`.

Special Considerations

This deprecated function performs searches synchronously. Apple recommends using the asynchronous `SKSearchCreate` function instead.

In the current implementation of Search Kit, unary Boolean operators are not implemented. A search, for example, for 'not blue', returns zero documents no matter what their content.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
`SKSearch.h`

SKSearchResultsGetCount

Gets the total number of found items in a search. (Deprecated in Mac OS X v10.4. Use `SKSearchCreate` (page 45) instead.)

```
CFIndex SKSearchResultsGetCount (
    SKSearchResultsRef  inSearchResults
);
```

Parameters

inSearchResults
A search results object containing the results of a query.

Return Value
A CFIndex object containing the total number of found items in a search.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
`SKSearch.h`

SKSearchResultsGetInfoInRange

Extracts information from a Search Kit query result. (Deprecated in Mac OS X v10.4. Use `SKSearchCreate` (page 45) instead.)

```
CFIndex SKSearchResultsGetInfoInRange (
    SKSearchResultsRef  inSearchResults,
    CFRange             inRange,
    SKDocumentRef       *outDocumentsArray,
    SKIndexRef          *outIndexesArray,
    float               *outScoresArray
);
```

Parameters

inSearchResults

The search results whose information you want to extract.

inRange

The starting ranking and total number of found items to obtain, specified as `(Location, Length)`. 'Location' specifies the starting item by ranking, with the top-ranked item having a location of 0. 'Length' specifies the total number of items to include in the results. For example, (0,1) indicates the first item, which is also the highest-ranking item. (1,1) indicates the second item, which is also the second-highest-ranking item. (0,5) means to get the first 5 items.

outDocumentsArray

On output, points to an array of found document URL objects (SKDocumentRefs).

outIndexesArray

On output, points to an array of indexes in which the found document URL objects reside. Can be `NULL` on input, provided that your application doesn't need this information.

outScoresArray

On output, points to an array of correspondence scores for found items. Can be `NULL` on input, provided that your application doesn't need this information.

Return Value

The number of items returned—usually the same number as specified by the length item in the `inRange` parameter.

Discussion

This function provides results to its output parameters in the order in which they are found, to reduce latency and to support search-as-you-type functionality.

Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared In

`SKSearch.h`

SKSearchResultsGetTypeID

Gets the type identifier for Search Kit search results. (Deprecated in Mac OS X v10.4. Use `SKSearchCreate` (page 45) instead.)

```
CFTypeID SKSearchResultsGetTypeID (void);
```

Return Value

A CFTypeID object containing the type identifier for the SKSearchResults opaque type.

Discussion
Search Kit represents search results with search results objects (`SKSearchResultsRef` (page 57) opaque types). If your code needs to determine whether a particular data type is a search result, you can use this function along with the `CFGetTypeID` function and perform a comparison.

Never hard-code the search result type ID because it can change from one release of Mac OS X to another.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
`SKSearch.h`

Document Revision History

This table describes the changes to *Search Kit Reference*.

Date	Notes
2010-03-24	Improved the description for the `kSKProximityIndexing` (page 59) text analysis key.
2009-05-06	Added notes that SKSearch and SKIndex objects cannot be used with CFMakeCollectable.
2006-07-24	Minor updates and corrections.
2006-07-14	Minor updates and corrections.
2006-03-08	Minor updates and corrections.
2005-12-06	Minor updates and corrections.
2005-08-11	Clarified descriptions of `SKIndexSetMaximumBytesBeforeFlush` (page 43) and `SKIndexGetMaximumBytesBeforeFlush` (page 35) functions.
2005-06-04	Reorganized Constants section and clarified constant descriptions and abstracts.
	Corrected description of the "`Text Analysis Keys`" (page 58) constants in regard to the underscore character. Also added type descriptions to the "`Text Analysis Keys`" constants.
	Clarified behavior of `AND`, `OR`, `NOT`, and `*` operators in the `SKSearchCreate` (page 45) function.
	Added explanation of Unicode normalization and other normalizations to discussion for the `SKSearchCreate` (page 45) function.
	Corrected descriptions of the `SKSearchCreate` (page 45), `SKSearchFindMatches` (page 47), and `SKSearchCancel` (page 44) functions in regard to initiating and canceling search operations.
2005-04-29	Reorganized document around new asynchronous searching. Renamed function group for searching to "Fast Asynchronous Searching" (page 12). Created legacy function group for deprecated functions related to synchronous search: "Legacy Support for Synchronous Searching" (page 13). Changed group name for callback to "Search Kit Legacy Callbacks" (page 54). Added deprecation information to data types and constants: `SKSearchResultsRef` (page 57), `SKSearchGroupRef` (page 58), `kSKLanguageTypes` (page 63), and "`Deprecated Search Keys`" (page 63). Added new function group for summarization: "Working With Summarization" (page 12).

Date	Notes
	Updated introduction. Added thread safety information throughout document.
	Added documentation for new functions available in Mac OS X v10.4: SKIndexClose (page 20), SKIndexCopyInfoForDocumentIDs (page 25), SKIndexCopyDocumentRefsForDocumentIDs (page 23), SKIndexCopyDocumentURLsForDocumentIDs (page 24), SKSearchCreate (page 45), SKSearchFindMatches (page 47), SKSearchCancel (page 44), SKSearchGetTypeID (page 48), SKSummaryCreateWithString (page 50), SKSummaryGetSentenceSummaryInfo (page 53), SKSummaryGetParagraphSummaryInfo (page 51), SKSummaryGetSentenceCount (page 52), SKSummaryGetParagraphCount (page 51), SKSummaryCopySentenceAtIndex (page 49), SKSummaryCopyParagraphAtIndex (page 48), SKSummaryCopySentenceSummaryString (page 50), SKSummaryCopyParagraphSummaryString (page 49), and SKSummaryGetTypeID (page 53).
	Updated documentation for all other functions.
	Added documentation for new types and constants available in Mac OS X v10.4: SKSearchRef (page 56) and "SKSearchOptions" (page 61).
2004-10-11	Added note on thread safety to indexing and searching functions.
	Added deprecation information to SKSearchGroupCreate (page 65), SKSearchGroupCopyIndexes (page 65), SKSearchGroupGetTypeID (page 66), SKSearchResultsCreateWithDocuments (page 67), SKSearchResultsGetInfoInRange (page 70), SKSearchResultsCopyMatchingTerms (page 67), SKSearchResultsGetCount (page 70), SKSearchResultsGetTypeID (page 71), and SKSearchResultsCreateWithQuery (page 69).
	Updated introduction. Clarified abstract and discussion for SKIndexFlush (page 30) and SKIndexCompact (page 21).
2004-06-28	Added descriptions for Mac OS X v10.4 APIs including SKIndexClose (page 20), SKIndexCopyInfoForDocumentIDs (page 25), SKIndexCopyDocumentRefsForDocumentIDs (page 23), SKIndexCopyDocumentURLsForDocumentIDs (page 24), SKSearchCreate (page 45), SKSearchCancel (page 44), SKSearchFindMatches (page 47), SKSearchGetTypeID (page 48), SKSearchRef (page 56), kSKProximityIndexing (page 59), kSKMaximumTerms (page 59), kSKTermChars (page 60), kSKStartTermChars (page 60), kSKEndTermChars (page 60), and "SKSearchOptions" (page 61).
	Updated introduction. Clarified abstract and discussion for SKIndexFlush (page 30) and SKIndexCompact (page 21).

Date	Notes
2004-05-20	Reorganized function groups to better reflect use of functions. Clarified descriptions of `SKSearchResultsCreateWithQuery` (page 69), `SKIndexAddDocumentWithText` (page 19), `SKIndexAddDocument` (page 18), `SKIndexCreateWithURL` (page 28), `SKIndexCreateWithMutableData` (page 27), `SKIndexOpenWithData` (page 39), `SKIndexOpenWithMutableData` (page 39), and others. Changed descriptions of the SKDocumentRef opaque data type from "document reference" to "document URL object."
2003-11-12	Minor additions and corrections. Clarified description for `SKSearchResultsCopyMatchingTerms` (page 67).
2003-10-17	First publication of Search Kit Reference.

Document Revision History