Dictionary Manager Reference

(Not Recommended)

Carbon > Text & Fonts

2007-12-11

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

Simultaneously published in the United States and Canada.

Contents

Dictionary Manager Reference (Not Recommended)

Framework:	ApplicationServices/ApplicationServices.h
Declared in	Dictionary.h

Overview

Important: The Dictionary Manager is deprecated as of Mac OS X v10.5 and is not available to 64-bit applications. Instead, use the APIs presented by Dictionary Services, which are described in *Dictionary Services Programming Guide* and *Dictionary Services Reference*.

The Dictionary Manager facilitates the use of dictionary files by such programs as spelling checkers and input methods. The Dictionary Manager separates dictionary data from the code that accesses the data.

The Dictionary Manager uses access method plug-ins to mediate access to the dictionary's data. The use of access-method plug-ins allows the Dictionary Manager to support a variety of data and does not require the internal structure of a dictionary to conform to a fixed format.

Dictionary Manager functions with the prefix "DCM" are Carbon-compliant. However, these functions are available only on systems that have the Japanese language kit installed.

Functions by Task

Obtaining the Version Number

`DCMLibraryVersion` (page 57) Deprecated in Mac OS X v10.5
Obtains the version number of the Dictionary Manager.

Working With a List of Dictionaries

You can use the functions in this section to obtain a list of the dictionaries available on the system. You can create a list of dictionaries, count the available dictionaries, and obtain the dictionary ID for each available dictionary.

`DCMCountObjectIterator` (page 33) Deprecated in Mac OS X v10.5
Obtains the number of dictionaries in a list.

`DCMCreateDictionaryIterator` (page 35) Deprecated in Mac OS X v10.5
Obtains a list of the dictionaries available on the system.

`DCMDisposeObjectIterator` (page 40) Deprecated in Mac OS X v10.5
Disposes of a iterator.

`DCMIterateObject` (page 57) Deprecated in Mac OS X v10.5
Obtains the object ID for a dictionary from a list of available dictionaries.

`DCMResetObjectIterator` (page 62) Deprecated in Mac OS X v10.5
Resets an iterator to the start of the dictionary list.

Obtaining Access Method Information

`DCMCreateAccessMethodIterator` (page 35) Deprecated in Mac OS X v10.5
Obtains a list of the available access methods.

`DCMGetAccessMethodIDFromName` (page 42) Deprecated in Mac OS X v10.5
Obtains the ID for access method.

Working With a Dictionary File

`DCMCloseDictionary` (page 32) Deprecated in Mac OS X v10.5
Closes a dictionary.

`DCMCompactDictionary` (page 33) Deprecated in Mac OS X v10.5
Compacts a dictionary.

`DCMCountRecord` (page 34) Deprecated in Mac OS X v10.5
Return number of records in a dictionary.

`DCMDeleteDictionary` (page 37) Deprecated in Mac OS X v10.5
Deletes a dictionary.

`DCMDeriveNewDictionary` (page 38) Deprecated in Mac OS X v10.5
Create a new dictionary based on an existing dictionary.

`DCMGetDictionaryIDFromFile` (page 44) Deprecated in Mac OS X v10.5
Obtains the ID associated with a dictionary file.

`DCMGetDictionaryIDFromRef` (page 44) Deprecated in Mac OS X v10.5
Obtains the dictionary ID associated with a dictionary reference.

`DCMGetFileFromDictionaryID` (page 51) Deprecated in Mac OS X v10.5
Obtains the file specification associated with a dictionary ID.

`DCMNewDictionary` (page 58) Deprecated in Mac OS X v10.5
Creates a new dictionary.

`DCMOpenDictionary` (page 59) Deprecated in Mac OS X v10.5
Opens a dictionary.

`DCMRegisterDictionaryFile` (page 60) Deprecated in Mac OS X v10.5
Registers a dictionary.

`DCMReorganizeDictionary` (page 62) Deprecated in Mac OS X v10.5
Reorganizes a dictionary.

`DCMUnregisterDictionary` (page 65) Deprecated in Mac OS X v10.5
Unregisters a dictionary.

Changing Access Privileges

`DCMGetDictionaryWriteAccess` (page 46) Deprecated in Mac OS X v10.5
Obtains write access for an open dictionary.

`DCMReleaseDictionaryWriteAccess` (page 61) Deprecated in Mac OS X v10.5
Releases write access to a dictionary.

Getting and Setting Dictionary Properties

`DCMGetDictionaryProperty` (page 45) Deprecated in Mac OS X v10.5
Obtains the data associated with a property tag.

`DCMGetDictionaryPropertyList` (page 46) Deprecated in Mac OS X v10.5
Obtains a list of property tags from a dictionary.

`DCMSetDictionaryProperty` (page 63) Deprecated in Mac OS X v10.5
Sets a property for a dictionary. Set the properties to a dictionary.

Working With Dictionary Records

`DCMAddRecord` (page 31) Deprecated in Mac OS X v10.5
Add a new record to a dictionary.

`DCMCountRecordIterator` (page 34) Deprecated in Mac OS X v10.5
Returns the number of records contained in a list of search results.

`DCMDeleteRecord` (page 38) Deprecated in Mac OS X v10.5
Delete specified record from the dictionary.

`DCMDisposeRecordIterator` (page 40) Deprecated in Mac OS X v10.5
Disposes of a list of search results.

`DCMFindRecords` (page 41) Deprecated in Mac OS X v10.5
Obtains a list of dictionary records that meet specified criteria.

`DCMGetNextRecord` (page 52) Deprecated in Mac OS X v10.5
Obtains the next specified record.

`DCMGetNthRecord` (page 53) Deprecated in Mac OS X v10.5
Return records in a specified order within the dictionary.

`DCMGetPrevRecord` (page 54) Deprecated in Mac OS X v10.5
Obtains the previous record. Return the record previous to the specified record.

`DCMGetRecordSequenceNumber` (page 55) Deprecated in Mac OS X v10.5
Obtains the sequence number for the specified record in a dictionary.

`DCMIterateFoundRecord` (page 56) Deprecated in Mac OS X v10.5
Retrieves one record from a list of search results.

Working With Fields in a Single Record

`DCMCreateFieldInfoRecord` (page 36) Deprecated in Mac OS X v10.5
Creates a field information record.

`DCMGetDictionaryFieldInfo` (page 43) Deprecated in Mac OS X v10.5
Obtains field information for a specified field in a dictionary record.

`DCMGetFieldAttributes` (page 47) Deprecated in Mac OS X v10.5
Obtains the field attributes for a field information record.

`DCMGetFieldData` (page 48) Deprecated in Mac OS X v10.5
Obtains data from one or more fields in a specified record.

`DCMGetFieldDefaultData` (page 49) Deprecated in Mac OS X v10.5
Obtains default data for a field information record.

`DCMGetFieldFindMethods` (page 49) Deprecated in Mac OS X v10.5
Obtains the search methods for a field information record.

`DCMGetFieldMaxRecordSize` (page 50) Deprecated in Mac OS X v10.5
Obtains the maximum data size for a field in a dictionary record.

`DCMGetFieldTagAndType` (page 51) Deprecated in Mac OS X v10.5
Obtains the field tag and type associated with a field information record.

`DCMSetFieldData` (page 64) Deprecated in Mac OS X v10.5
Set the data to a specific field of a specified record.

Callbacks

DCMProgressFilterProcPtr

Displays the progress state of a reorganization or compaction operation.

```
typedef Boolean (*DCMProgressFilterProcPtr)
(
    Boolean determinateProcess,
    UInt16 percentageComplete,
    UInt32 callbackUD
);
```

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

```
Boolean DCMProgressFilterProcPtr (
    Boolean determinateProcess,
    UInt16 percentageComplete,
    UInt32 callbackUD
);
```

Discussion

You supply this callback as a parameter to the `DCMReorganizeDictionary` (page 62) and `DCMCompactDictionary` (page 33) functions.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

Data Types

DCMAccessMethodID

Represents an access method ID.

```
typedef DCMObjectID DCMAccessMethodID;
```

Discussion
See DCMGetAccessMethodIDFromName (page 42).

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
Dictionary.h

DCMAccessMethodIterator

Represents a list of access methods.

```
typedef DCMObjectIterator DCMAccessMethodIterator;
```

Discussion
See DCMCreateAccessMethodIterator (page 35).

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
Dictionary.h

DCMDictionaryHeader

Contains header information for a dictionary.

```
struct DCMDictionaryHeader {
    FourCharCode headerSignature;
    UInt32 headerVersion;
    ByteCount headerSize;
    Str63 accessMethod;
};
typedef struct DCMDictionaryHeader DCMDictionaryHeader;
```

Fields
headerSignature
The header signature must be 'dict'.

`headerVersion`
: The version of header information. The current version is specified by the constant `kDCMDictionaryHeaderVersion`.

`headerSize`
: The size of the header information. The current size is 76 bytes.

`accessMethod`
: The library name of the access method.

Discussion
The internal structure of dictionaries used by the Dictionary Manager can be broadly divided into three structures: dictionary header section, dictionary property section, and data record section. However, internal structures apart from the dictionary header section rely on the access method and can be freely defined, so you must be aware that this is strictly a structural model for dictionaries viewed from outside.

Dictionary files that are managed by the Dictionary Manager have a dictionary header at the start of its data fork in a defined format, and this header must contain a signature which indicates that it is a dictionary supported by the Dictionary Manager. The file must also contain basic information such as the name of the access method which can manage this dictionary. The dictionary header is the only information in the dictionary that can be accessed without the use of an access method.

The dictionary property section contains information about the dictionary as a whole, such as access restrictions (whether it is write enabled, whether its content can be downloaded). Properties are managed by tags, and it is possible to define and save different types of information.

The data record section contains registered data records. Each data record is further divided into fields," and these are managed using tags which represent the meaning of the data—for example, 'yomi' (read) and 'hins' (part of speech). At least one field in a record must be a key field. The key field is used as an index, and tag and key data for this field are used to find records. When searching through records, you can specify other field tags in addition to the key field, thereby enabling a variety of data to be obtained at once.

Each field in a record has field attributes that specify the field's data type, maximum data length, and default data The key data field also has an attribute that specifies the search methods it supports.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMDictionaryID

Represents the ID associated with a specific dictionary.

```
typedef DCMObjectID DCMDictionaryID;
```

Discussion
When you call the function `DCMRegisterDictionaryFile` (page 60) to register a dictionary, the Dictionary Manager assigned a unique ID that you need to use in subsequent calls to the Dictionary Manager. The `DCMDictionaryID` value is not persistent across system restarts, so you must not save this value for future use. Each time your application launches or the system starts up you need to obtain the newly-assigned `DCMDictionaryID` value.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMDictionaryIterator

Represents a list of dictionaries.

```
typedef DCMObjectIterator DCMDictionaryIterator;
```

Discussion
See `DCMCreateDictionaryIterator` (page 35).

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMDictionaryRef

Represents a reference to a dictionary.

```
typedef DCMObjectRef DCMDictionaryRef;
```

Discussion
You can obtain a dictionary reference by calling the function `DCMOpenDictionary` (page 59). You need this reference to operate on the dictionary.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMFieldTag

Represents a field inside a dictionary.

```
typedef DescType DCMFieldTag;
```

Discussion
A field tag is a 4-byte value used to identify a fields. A field must be unique within a dictionary. See "Field Data Tags" (page 21) for more information.

Availability
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMFieldType

Represents the data type of the data stored in a field.

```
typedef DescType DCMFieldType;
```

Discussion
A field tag is a 4-byte value used to specify the data type contained in a field. The basic definition is the data type defined by Apple Event Manager. "Field Data Types" (page 21).

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMFoundRecordIterator

Represents a reference to an opaque list of search results.

```
typedef struct OpaqueDCMFoundRecordIterator * DCMFoundRecordIterator;
```

Discussion
See `DCMFindRecords` (page 41) for more information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMObjectID

Represents a reference to an opaque dictionary ID.

```
typedef struct OpaqueDCMObjectID * DCMObjectID;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMObjectIterator

Defines a reference to an opaque dictionary object iterator.

```
typedef struct OpaqueDCMObjectIterator * DCMObjectIterator;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMObjectRef

Defines a reference to an opaque dictionary reference.

```
typedef struct OpaqueDCMObjectRef * DCMObjectRef;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMProgressFilterUPP

Defines a universal procedure pointer (UPP) to a progress filter callback.

```
typedef DCMProgressFilterProcPtr DCMProgressFilterUPP;
```

Discussion
For more information see `DCMProgressFilterProcPtr` (page 10). You pass a progress filter callback UPP to the functions `DCMReorganizeDictionary` (page 62) and `DCMCompactDictionary` (page 33).

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMUniqueID

Represents the unique ID of a record in a dictionary.

```
typedef UInt32 DCMUniqueID;
```

Discussion
This ID is used in many functions. For example, see `DCMGetNextRecord` (page 52),`DCMGetPrevRecord` (page 54), and `DCMGetNthRecord` (page 53).

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

Constants

Access Method Features

Specify features associated with an access method.

```
enum {
    kDCMCanUseFileDictionaryMask = 0x00000001,
    kDCMCanUseMemoryDictionaryMask = 0x00000002,
    kDCMCanStreamDictionaryMask = 0x00000004,
    kDCMCanHaveMultipleIndexMask = 0x00000008,
    kDCMCanModifyDictionaryMask = 0x00000010,
    kDCMCanCreateDictionaryMask = 0x00000020,
    kDCMCanAddDictionaryFieldMask = 0x00000040,
    kDCMCanUseTransactionMask = 0x00000080
};
typedef OptionBits DCMAccessMethodFeature;
```

Constants

`kDCMCanUseFileDictionaryMask`
Specifies the file dictionary mask.
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.
Declared in `Dictionary.h`.

`kDCMCanUseMemoryDictionaryMask`
Specifies the memory dictionary mask
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.
Declared in `Dictionary.h`.

`kDCMCanStreamDictionaryMask`
Specifies t the stream dictionary mask
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.
Declared in `Dictionary.h`.

`kDCMCanHaveMultipleIndexMask`
Specifies the multiple index mask
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.
Declared in `Dictionary.h`.

`kDCMCanModifyDictionaryMask`
: Specifies the modify dictionary mask

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMCanCreateDictionaryMask`
: Specifies the create dictionary mask

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMCanAddDictionaryFieldMask`
: Specifies to use the add dictionary field mask

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMCanUseTransactionMask`
: Specifies to use the use transaction dictionary mask

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Discussion
The Dictionary Manager does not operate directly on a dictionary. Instead it accesses dictionaries using the access method specified for the dictionary. The access method mediates between the dictionary and the Dictionary Manager, so that the Dictionary Manager does not need to know anything about the physical format of the dictionary. As a result, the dictionary can have an free-form internal structure. You can also to use an existing dictionary as long as the dictionary has its own access method.

Normally (except when you create a dictionary) you do not need to recognize the existence of the access method.

Dictionary Classes

Specify dictionary classes associated with the property `pDCMClass`.

```
enum {
    kDCMUserDictionaryClass = 0,
    kDCMSpecificDictionaryClass = 1,
    kDCMBasicDictionaryClass = 2
};
```

Constants
`kDCMUserDictionaryClass`
: Indicates a user dictionary.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMSpecificDictionaryClass`
Indicates a specific dictionary.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMBasicDictionaryClass`
Indicates a basic dictionary.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Dictionary Information Constants

Specify various data in a dictionary.

```
enum {
    kDictionaryFileType = 'dict',
    kDCMDictionaryHeaderSignature = 'dict',
    kDCMDictionaryHeaderVersion = 2
};
```

Constants

`kDictionaryFileType`
Specify the file type.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMDictionaryHeaderSignature`
Specify the header in a dictionary.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMDictionaryHeaderVersion`
Specify the header version.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Dictionary Properties

Specify standard dictionary properties.

```
enum {
    pDCMAccessMethod = 'amtd',
    pDCMPermission = 'perm',
    pDCMListing = 'list',
    pDCMMaintenance = 'mtnc',
    pDCMLocale = 'locl',
    pDCMClass = pClass,
    pDCMCopyright = 'info'
};
```

Constants

`pDCMAccessMethod`

Specifies an access method; the associated data type is `typeChar` and is read-only. This property is required.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`pDCMPermission`

Specifies a permission level; the associated data type is `typeUInt16`. This property is required.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`pDCMListing`

Specifies whether or not to allow data to be downloaded from the dictionary; the associated data type is `typeUInt16`. This property is optional.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`pDCMMaintenance`

This property is obsolete.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`pDCMLocale`

Specifies a script code for the dictionary contents; the associated data type is `typeUInt32`. The default is `kLocaleIdentifierWildCard`. This property is optional.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`pDCMClass`

Specifies a dictionary class; the associated data type is `typeUInt16`. The possible values are `kDCMUserDictionaryClass`, `kDCMSpecificDictionaryClass`), and `kDCMBasicDictionaryClass`. This property is optional.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`pDCMCopyright`
: Specifies copyright information; the associated data type is `typeChar` and is read-only.

  Available in Mac OS X v10.0 and later.

  Not available to 64-bit applications.

  Declared in `Dictionary.h`.

Discussion
Of the properties listed here, the only property that must always be supported is the `pDCMPermission` property. If a existing property is set by calling the function `DCMSetDictionaryProperty` (page 63) the property data is overwritten, except if the property is defined as read-only property. However, some read-write properties can be restricted so they can be modified only by an access method.

Functions that obtain data from a dictionary cannot be used on any dictionary whose `pDCMListing` property is set to `kDCMProhibitListing`. In this case, the function trying to obtain the data returns the result `dcmPermissionErr`. This means the dictionary data is protected because it is impossible to output the contents of the dictionary as text.

Field Attributes

Specify attributes for fields in a dictionary.

```
enum {
    kDCMIndexedFieldMask = 0x00000001,
    kDCMRequiredFieldMask = 0x00000002,
    kDCMIdentifyFieldMask = 0x00000004,
    kDCMFixedSizeFieldMask = 0x00000008,
    kDCMHiddenFieldMask = 0x80000000
};
typedef OptionBits DCMFieldAttributes;
```

Constants
`kDCMIndexedFieldMask`
: Specifies a key field that can be used in search.

  Available in Mac OS X v10.0 and later.

  Not available to 64-bit applications.

  Declared in `Dictionary.h`.

`kDCMRequiredFieldMask`
: Specifies an essential field in which data must always be provided when adding a record.

  Available in Mac OS X v10.0 and later.

  Not available to 64-bit applications.

  Declared in `Dictionary.h`.

`kDCMIdentifyFieldMask`
: Specifies a field that can be used to identify the same record.

  Available in Mac OS X v10.0 and later.

  Not available to 64-bit applications.

  Declared in `Dictionary.h`.

`kDCMFixedSizeFieldMask`

Specifies a fixed-size field.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMHiddenFieldMask`

Specfies a hidden field that is not returned by the function that obtains the field list.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Discussion

The records contained in a dictionary have a variety of fields. For example, a record in a dictionary for kana-kanji conversion, typically has a "read" key field which stores hiragana character strings that are found, as well as a "part of speech" and a "notation" fields that refers to the search results. These respective fields also require attributes such as data type, maximum length of stored data, and fixed length/variable length.

When you create a new dictionary using the Dictionary Manager, you can specify the fields to include in the dictionary and the attributes associated with each of those field.

Field Data Tags

Specify field tags in an Apple Japanese dictionary.

```
enum {
    kDCMJapaneseYomiTag = 'yomi',
    kDCMJapaneseHyokiTag = 'hyok',
    kDCMJapaneseHinshiTag = 'hins',
    kDCMJapaneseWeightTag = 'hind',
    kDCMJapanesePhoneticTag = 'hton',
    kDCMJapaneseAccentTag = 'acnt',
    kDCMJapaneseOnKunReadingTag = 'OnKn',
    kDCMJapaneseFukugouInfoTag = 'fuku'
};
```

Field Data Types

Specify the data types for the values associated with field tags.

```
enum {
    kDCMJapaneseYomiType = typeUnicodeText,
    kDCMJapaneseHyokiType = typeUnicodeText,
    kDCMJapaneseHinshiType = 'hins',
    kDCMJapaneseWeightType = typeShortInteger,
    kDCMJapanesePhoneticType = typeUnicodeText,
    kDCMJapaneseAccentType = 'byte',
    kDCMJapaneseOnKunReadingType = typeUnicodeText,
    kDCMJapaneseFukugouInfoType = 'fuku'
};
```

Constants

`kDCMJapaneseYomiType`

Specifies the data type is Unicode text.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapaneseHyokiType`

Specifies the data type is Unicode text.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapaneseHinshiType`

Specifies the data type is `'hins'`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapaneseWeightType`

Specifies the data type is short integer.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapanesePhoneticType`

Specifies the data type is Unicode text.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapaneseAccentType`

Specifies the data type is `byte`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapaneseOnKunReadingType`
Specifies the data type is Unicode text.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMJapaneseFukugouInfoType`
Specifies the data type is `'fuku'`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Field Info Record Entries

Specifies tags for the entries in a field information record.

```
enum {
    keyDCMFieldTag = 'ftag',
    keyDCMFieldType = 'ftyp',
    keyDCMMaxRecordSize = 'mrsz',
    keyDCMFieldAttributes = 'fatr',
    keyDCMFieldDefaultData = 'fdef',
    keyDCMFieldName = 'fnam',
    keyDCMFieldFindMethods = 'ffnd'
};
```

Constants

`keyDCMFieldTag`
The data type of the associated data is `typeEnumeration`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`keyDCMFieldType`
The data type of the associated data is `typeEnumeration`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`keyDCMMaxRecordSize`
The data type of the associated data is `typeMagnitude`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`keyDCMFieldAttributes`
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`keyDCMFieldDefaultData`

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`keyDCMFieldName`

The data type of the associated data is `typeChar`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`keyDCMFieldFindMethods`

The data associated with this field is a list (`typeAEList`) of `typeDCMFindMethod` values.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Field Info Record Types

Specify special types for a field information record.

```
enum {
    typeDCMFieldAttributes = 'fatr',
    typeDCMFindMethod = 'fmth'
};
```

Constants

`typeDCMFieldAttributes`

Specifies a data type for field attributes.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`typeDCMFindMethod`

Specifies a data type for search methods.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Listing Permissions

Specifies whether to allow or prohibit a dictionary from being listed.

```
enum {
    kDCMAllowListing = 0,
    kDCMProhibitListing = 1
};
```

Constants

`kDCMAllowListing`

Specifies to allow listing.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMProhibitListing`

Specifies to prohibit listing.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Discussion

A dictionary that is prohibited from being listed is available only to the application that created the dictionary.

Permission Levels

Specify permission levels for a dictionary.

```
enum {
    kDCMReadOnlyDictionary = 0,
    kDCMReadWriteDictionary = 1
};
```

Constants

`kDCMReadOnlyDictionary`

Specifies the dictionary is read-only.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMReadWriteDictionary`

Specifies the dictionary has read-write permission.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Search Methods

Specify search criteria.

```
enum {
    kDCMFindMethodExactMatch = kAEEquals,
    kDCMFindMethodBeginningMatch = kAEBeginsWith,
    kDCMFindMethodContainsMatch = kAEContains,
    kDCMFindMethodEndingMatch = kAEEndsWith,
    kDCMFindMethodForwardTrie = 'ftri',
    kDCMFindMethodBackwardTrie = 'btri'
};
typedef OSType DCMFindMethod;
```

Constants

`kDCMFindMethodExactMatch`

Specifies an exact match.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMFindMethodBeginningMatch`

Specifies the beginning must match. For example, cat matches catch and catalog.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMFindMethodContainsMatch`

Specifies the match can be contained in a string. (rat matches crater, decorate)

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMFindMethodEndingMatch`

Specifies the end must match. For example, bat matches combat and acrobat.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMFindMethodForwardTrie`

Specifies partial character string from the front. For example, theme matches the, them, and theme. Used for morphological analysis.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMFindMethodBackwardTrie`

Specfies partial character string from the back. For example, flash matches ash, lash, and flash. Used for morphological analysis.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Wild Card Values

Represent any field tag or field type.

```
enum {
    kDCMAnyFieldTag = typeWildCard,
    kDCMAnyFieldType = typeWildCard
};
```

Constants

`kDCMAnyFieldTag`

Specifies any field.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

`kDCMAnyFieldType`

Specifies any type.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `Dictionary.h`.

Result Codes

The most common result codes returned by Dictionary Manager are listed below.

Result Code	Value	Description
`dcmParamErr`	-7100	Bad parameter. Available in Mac OS X v10.0 and later.
`dcmNotDictionaryErr`	-7101	Not a dictionary. Available in Mac OS X v10.0 and later.
`dcmBadDictionaryErr`	-7102	Invalid dictionary. Available in Mac OS X v10.0 and later.
`dcmPermissionErr`	-7103	Invalid permission. Available in Mac OS X v10.0 and later.
`dcmDictionaryNotOpenErr`	-7104	Dictionary not opened. Available in Mac OS X v10.0 and later.
`dcmDictionaryBusyErr`	-7105	Dictionary is busy. Available in Mac OS X v10.0 and later.
`dcmBlockFullErr`	-7107	Dictionary block is full. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
dcmNoRecordErr	-7108	No such record. Available in Mac OS X v10.0 and later.
dcmDupRecordErr	-7109	Same record already exists. Available in Mac OS X v10.0 and later.
dcmNecessaryFieldErr	-7110	Missing the required field. Available in Mac OS X v10.0 and later.
dcmBadFieldInfoErr	-7111	Incomplete field. Available in Mac OS X v10.0 and later.
dcmBadFieldTypeErr	-7112	No such field type supported. Available in Mac OS X v10.0 and later.
dcmNoFieldErr	-7113	No such field exists. Available in Mac OS X v10.0 and later.
dcmBadKeyErr	-7115	Bad key information. Available in Mac OS X v10.0 and later.
dcmTooManyKeyErr	-7116	Too many key fields. Available in Mac OS X v10.0 and later.
dcmBadDataSizeErr	-7117	Data size too large. Available in Mac OS X v10.0 and later.
dcmBadFindMethodErr	-7118	Search method not supported. Available in Mac OS X v10.0 and later.
dcmBadPropertyErr	-7119	No such property exists. Available in Mac OS X v10.0 and later.
dcmProtectedErr	-7121	Need a keyword to use the dictionary. Available in Mac OS X v10.0 and later.
dcmNoAccessMethodErr	-7122	No such access method exists. Available in Mac OS X v10.0 and later.
dcmBadFeatureErr	-7124	Invalid access method feature. Available in Mac OS X v10.0 and later.
dcmIterationCompleteErr	-7126	Iteration complete; no more items in the iterator. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`dcmBufferOverflowErr`	-7127	Data is larger than the buffer size. Available in Mac OS X v10.0 and later.

Deprecated Dictionary Manager Reference (Not Recommended) Functions

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

Deprecated in Mac OS X v10.5

DCMAddRecord

Add a new record to a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMAddRecord (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   Boolean checkOnly,
   const AEDesc *dataList,
   DCMUniqueID *newUniqueID
);
```

Parameters

dictionaryRef

A reference to the dictionary to which you want to add a record. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag

The field tag that specifies the record you want to add.

keySize

The size of the `keyData` parameter.

keyData

The key data of the record you want to add. You are responsible for allocating this buffer.

checkOnly

If true is specified, only a duplication check of records is carried out, and actual addition of records is not carried out.

dataList

A pointer to the data contained in the record you want to add. The `AEDesc` data structure contains two fields: A four-character code that specifies the type of data in the structure and an opaque storage type that points to the storage for the descriptor data. Each of the data structures in the `dataList` array specifies a field name (the four-character code) in the dictionary record and the data associated with that field. You must first call the function `DCMCreateFieldInfoRecord` (page 36) to create this array of data structures. When you create the data list, you must include all of the fields specified by the masks `kDCMIndexedFieldMask`, `kDCMRequiredFieldMask`, and `kDCMIdentifyFieldMask` as attributes of the field properties of the applicable dictionary.

newUniqueID

On output, the unique ID of the added record. If the added record is a duplicate record, the function returns the result `dcmDupRecordErr` and the unique ID of the existing record is returned in the `newUniqueID` parameter.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
When `checkOnly` is specified as `true`, if duplicate records do not exist, `noErr` is returned, and the `newUniqueID` value is undefined. When `checkOnly` is specified as `false`, if the same record exists, only the field data within that record is overwritten, and the unique ID of that record is returned as `newUniqueID`, and `dcmDupRecordErr` is returned as the return value. If the same record does not exist, the record is added, and the uniqueID of the added record is returned as `newUniqueID`, and `noErr` is returned as the return value.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMCloseDictionary

Closes a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMCloseDictionary (
   DCMDictionaryRef dictionaryRef
);
```

Parameters

dictionaryRef

A reference to the dictionary you want to close. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMCompactDictionary

Compacts a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMCompactDictionary (
   DCMDictionaryID dictionaryID,
   DCMProgressFilterUPP progressProc,
   UInt32 userData
);
```

Parameters

dictionaryID

The ID of the dictionary you want to compact. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

progressProc

A universal procedure pointer (UPP) to your progress callback function. This callback is not supported.

userData

Data needed by your progress callback function.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion

The function `DCMCompactDictionary` organizes the contents of a dictionary to reduce the size of the dictionary as much as possible. You cannot add additional records to a dictionary after you have compacted it unless you call the function `DCMReorganizeDictionary` (page 62) to expand the capacity of the dictionary.

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMCountObjectIterator

Obtains the number of dictionaries in a list. (Deprecated in Mac OS X v10.5.)

```
ItemCount DCMCountObjectIterator (
   DCMObjectIterator iterator
);
```

Parameters

iterator

The list of available dictionaries. You obtain this list by calling the function `DCMCreateDictionaryIterator` (page 35) or `DCMCreateAccessMethodIterator` (page 35).

Return Value

The number of dictionaries contained in the list specified by the `iterator` parameter.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMCountRecord

Return number of records in a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMCountRecord (
   DCMDictionaryID dictionaryID,
   ItemCount *count
);
```

Parameters

dictionaryID
The ID of dictionary whose records you want to count. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

count
On output, the number of records in the dictionary.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMCountRecordIterator

Returns the number of records contained in a list of search results. (Deprecated in Mac OS X v10.5.)

```
ItemCount DCMCountRecordIterator (
   DCMFoundRecordIterator recordIterator
);
```

Parameters

recordIterator
A reference to list of search results. You obtain a list of search results by calling the function `DCMFindRecords` (page 41).

Return Value
The number of records in the list specified by the `recordIterator` parameter.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMCreateAccessMethodIterator

Obtains a list of the available access methods. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMCreateAccessMethodIterator (
   DCMAccessMethodIterator *accessMethodIterator
);
```

Parameters

`accessMethodIterator`
On output, a list of access methods.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
You can operate on the list of access methods created by the function `DCMCreateAccessMethodIterator` by calling the functions `DCMCountObjectIterator` (page 33), `DCMIterateObject` (page 57), `DCMResetObjectIterator` (page 62), and `DCMDisposeObjectIterator` (page 40). You use this function along with the function `DCMGetAccessMethodIDFromName` (page 42) to obtain the `accessMethodID` that you supply to the function `DCMNewDictionary` (page 58) to create a new dictionary.

Normally you should not need to call this function because the Dictionary Manager handles the mapping of a dictionary to its access method.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMCreateDictionaryIterator

Obtains a list of the dictionaries available on the system. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMCreateDictionaryIterator (
   DCMDictionaryIterator *dictionaryIterator
);
```

Parameters

dictionaryIterator

On output, a reference to the list of available dictionaries. You are responsible for disposing of this list when you no longer need it.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion

The function `DCMCreateDictionaryIterator` scans the dictionary directories, registers the dictionaries found, and returns a list of dictionaries. In Mac OS X, the Dictionary Manager searches `(User's home)/Library/Dictionaries/`, `/Library/Dictionaries/`, and their subdirectories. In Mac OS 9, it searches the Extensions folder, Preferences folder, and their subfolders.

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMCreateFieldInfoRecord

Creates a field information record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMCreateFieldInfoRecord (
   DescType fieldTag,
   DescType fieldType,
   ByteCount maxRecordSize,
   DCMFieldAttributes fieldAttributes,
   AEDesc *fieldDefaultData,
   ItemCount numberOfFindMethods,
   DCMFindMethod findMethods[],
   AEDesc *fieldInfoRecord
);
```

Parameters

fieldTag

The tag for the field you want to create.

fieldType

The data type of the field.

maxRecordSize

The maximum size of the data in the field.

fieldAttributes

The attributes associated with the field. See "Field Attributes" (page 20) for more information.

fieldDefaultData
On input, points to the default data for the field.

numberOfFindMethods
The number of search methods associated with the field.

findMethods
On input, an array of search methods associated with the field. See "Search Methods" (page 25) for more information.

fieldInfoRecord
On output, points to the field information record for the newly-created field.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
You can add multiple fields in the form of an `AEDescList` data structure by repeatedly calling the function `DCMCreateFieldInfoRecord`. You use the field information record (`fieldInfoRecord`) when you call the function `DCMNewDictionary`. However, when calling the function for the first time, you must set the `descriptorType` of the `fieldInfoRecord` to `typeNull`.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMDeleteDictionary

Deletes a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMDeleteDictionary (
   DCMDictionaryID dictionaryID
);
```

Parameters
dictionaryID
The ID of dictionary you want to delete. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27). Returns the result `dcmDictionaryBusyErr` if another application has this dictionary open.

Discussion
The function `DCMDeleteDictionary` deletes the dictionary specified by the `dictionaryID` parameter.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMDeleteRecord

Delete specified record from the dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMDeleteRecord (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMUniqueID uniqueID
);
```

Parameters

dictionaryRef
A reference to the dictionary from which you want to delete a record. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
The field tag that specifies the record you want to delete.

keySize
The size of the `keyData` parameter.

keyData
The key data of the record you want to delete. You are responsible for allocating this buffer.

uniqueID
The unique ID of the record you want to delete.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMDeriveNewDictionary

Create a new dictionary based on an existing dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMDeriveNewDictionary (
   DCMDictionaryID srcDictionary,
   const FSSpec *newDictionaryFile,
   ScriptCode scriptTag,
   Boolean invisible,
   ItemCount recordCapacity,
   DCMDictionaryID *newDictionary
);
```

Parameters

srcDictionary

The ID of dictionary from which you want to derive a dictionary. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

newDictionaryFile

A pointer to an `FSSpec` structure that specifies the file name and location for the newly-created dictionary. This is an input parameter.

scriptTag

The script code of the file specified by the `newDictionaryFile` parameter.

invisible

A Boolean value that specifies whether the dictionary is available through the Dictionary Manager. Pass `true` if you do not want the dictionary to be available, `false` otherwise. If you set invisible to `true`, that dictionary is no longer seen by such functions as `DCMCreateDictionaryIterator`, so it becomes a dictionary that cannot be accessed from any application other than the application that created the dictionary.

recordCapacity

The number of records that can be stored in the dictionary. You can supply an approximate value if you do not know the exact number.

newDictionary

On output, points to the ID of the newly-created dictionary.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion

The function `DCMDeriveNewDictionary` creates an new dictionary file based on an existing dictionary specified in the `srcDictionary` parameter, with the same field configuration and properties, but does not contain the data from the source. The new dictionary is created with the name and at the location specified by the `newDictionaryFile` parameter. The newly-created dictionary is read-write enabled even if the source dictionary is read-only.

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMDisposeObjectIterator

Disposes of a iterator. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMDisposeObjectIterator (
   DCMObjectIterator iterator
);
```

Parameters

iterator

The list of available dictionaries that you want to dispose of.You obtain this list by calling the function `DCMCreateDictionaryIterator` (page 35) or `DCMCreateAccessMethodIterator` (page 35).

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion

You must dispose of a dictionary iterator when you no longer need it by calling the function `DCMDisposeObjectIterator` (page 40).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMDisposeRecordIterator

Disposes of a list of search results. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMDisposeRecordIterator (
   DCMFoundRecordIterator recordIterator
);
```

Parameters

recordIterator

A reference to the list of search results you want to dispose of. You obtain a list of search results by calling the function `DCMFindRecords` (page 41).

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMFindRecords

Obtains a list of dictionary records that meet specified criteria. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMFindRecords (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMFindMethod findMethod,
   ItemCount preFetchedDataNum,
   DCMFieldTag preFetchedData[],
   ItemCount skipCount,
   ItemCount maxRecordCount,
   DCMFoundRecordIterator *recordIterator
);
```

Parameters

dictionaryRef

A reference to the dictionary you want to search. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag

A tag that specifies the field to search through. See "Field Data Tags" (page 21) for a list of the fields you can specify for an Apple Japanese dictionary.

keySize

The length of the keyword specified by the `keyData` parameter.

keyData

The string for which you want to search.

findMethod

The search method to use. See "Search Methods" (page 25) for a description of the methods you can supply.

preFetchedDataNum

The number of items in the `preFetchedData` array.

preFetchedData

An array of the tags obtained during the search. See "Field Data Tags" (page 21) for a list of the field tags that can be obtained for an Apple Japanese dictionary.

skipCount

The number of records you want to skip during the search. You can use this value along with the `maxRecordCount` parameter to search through a dictionary in chunks. For example, if you want to obtain 10 matches at a time, the first time you search you should set the `skipCount` parameter to 0 and the `maxRecordCount` to 10. The second time you search you set `skipCount` to 10 and `maxRecordCount` to10. Each subsequent time you search, you increment `skipCount` by 10, keeping `maxRecordCount` set to10.

maxRecordCount

The maximum number of results to return. Pass 0 if you want all matching records returned. If the number of matching records is smaller than the maximum number of results to return, the search is terminated and the matching records returned in the `recordIterator` parameter. See the description of the `skipCount` parameter for information on how to use `maxRecordCount` to search through a dictionary in chunks.

recordIterator

On return, a reference to list of search results.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).If a match is not found, the function returns `dcmNoRecordErr`, and the `recordIterator` value is undefined.

Discussion
The function `DCMFindRecords` uses the specified key and search method to return a list of matching records. Search results are returned as a reference to a list of matching records. After you obtain this list, you can call the function `DCMGetFieldData` to retrieve each result in the list.

You can use the `preFetchedData` parameter to obtain a list of tags during the search. Data specified as pre-fetched are actually retrieved at the same time as the search key is retrieved, so you can specify data tags the need to be accessed fast and immediately after searching. In other words, you can avoid accessing and loading data that are not going to be used immediately by omitting those tags from the `preFetchedData` list. (You can retrieve those data later by calling the function `DCMGetFieldData` (page 48).) For example: An application that searches pictures by date, displays the found titles in the list, and shows the picture only if the title is double-clicked, the key is "date", the pre-fetched data is "title", and the "picture" is retrieved later if needed by calling the function `DCMGetFieldData` (page 48).

You pass the record iterator (`recordIterator`) as a parameter to the function DCM `DCMCountRecordIterator` (page 34) to obtain the number of items in the list. You can obtain the individual items in the list by passing the record iterator to the function `DCMIterateFoundRecord` (page 56). Your application is responsible for disposing of the record iterator when you no longer need it by calling the function `DCMDisposeRecordIterator` (page 40).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetAccessMethodIDFromName

Obtains the ID for access method. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetAccessMethodIDFromName (
   ConstStr63Param accessMethodName,
   DCMAccessMethodID *accessMethodID
);
```

Parameters

`accessMethodName`
The name of access method whose ID you want to obtain.

`accessMethodID`
On output, the ID for the access method specified by the `accessMethodName` parameter.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
You can use this function to obtain the `accessMethodID` that you supply to the function `DCMNewDictionary` (page 58) to create a new dictionary.

Normally you should not need to call this function because the Dictionary Manager handles the mapping of a dictionary to its access method.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetDictionaryFieldInfo

Obtains field information for a specified field in a dictionary record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetDictionaryFieldInfo (
   DCMDictionaryID dictionaryID,
   DCMFieldTag fieldTag,
   AEDesc *fieldInfoRecord
);
```

Parameters

`dictionaryID`
The ID of dictionary from which you want to obtain field information. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

`fieldTag`
The tag of the field whose field information you want to obtain. If you pass `'****'` (`typeWildCard`) as the field tag, all of the field information contained in the specified dictionary is returned in the `fieldInfoRecord` parameter.

`fieldInfoRecord`
On return, points to the field information for the specified field. You are responsible for disposing of this structure by calling the Apple Event Manager function `AEDisposeDesc`.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetDictionaryIDFromFile

Obtains the ID associated with a dictionary file. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetDictionaryIDFromFile (
   const FSSpec *fileRef,
   DCMDictionaryID *dictionaryID
);
```

Parameters

fileRef

The file specification for the dictionary whose ID you want to obtain.

dictionaryID

On output, points to the ID for the dictionary specified by the `fileRef` parameter.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27). If the file is not a dictionary, the result `dcmNotDictionaryErr` is returned. If the dictionary is not yet registered, the result `dcmBadDictionaryErr` is returned.

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMGetDictionaryIDFromRef

Obtains the dictionary ID associated with a dictionary reference. (Deprecated in Mac OS X v10.5.)

```
DCMDictionaryID DCMGetDictionaryIDFromRef (
   DCMDictionaryRef dictionaryRef
);
```

Parameters

dictionaryRef

A reference to the dictionary whose ID you want to obtain. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

Return Value

On return, the ID of the dictionary specified by the `dictionaryRef` parameter. If `dictionaryRef` is invalid, the result `kDCMInvalidObjectID` is returned. See page for a description of the `DCMDictionaryID` data type.

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetDictionaryProperty

Obtains the data associated with a property tag. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetDictionaryProperty (
   DCMDictionaryID dictionaryID,
   DCMFieldTag propertyTag,
   ByteCount maxPropertySize,
   ByteCount *actualSize,
   LogicalAddress propertyValue
);
```

Parameters

dictionaryID
The ID of dictionary whose property you want to obtain. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

propertyTag
The property tag whose data you want to obtain.

maxPropertySize
The size of the data specified by the `propertyValue` parameter.

actualSize
On output, the actual size of the data specified by the `propertyValue` parameter.

propertyValue
On output, points to the property data.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27). Returns the result `dcmBadPropertyErr` if the property tag does not exist.

Discussion
If you don't know the size of the property whose data you want to obtain, you need to call this function twice as follows:

- The first time you call the function `DCMGetProperty`, pass the dictionary ID, the property tag, `0` for the `maxPropertySize` parameter and `Null` for `propertyValue`. Then allocate a `propertyValue` buffer of the size returned by the `actualSize` parameter.
- The second time you call the function `DCMGetProperty`, pass the dictionary ID, the property tag, the correct size for the `maxPropertySize` parameter, and the `propertyValue` buffer of the appropriate size.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetDictionaryPropertyList

Obtains a list of property tags from a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetDictionaryPropertyList (
   DCMDictionaryID dictionaryID,
   ItemCount maxPropertyNum,
   ItemCount *numProperties,
   DCMFieldTag propertyTag[]
);
```

Parameters

dictionaryID
The ID of dictionary whose list of property tags you want to obtain. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

maxPropertyNum
The maximum number of property tags in the list.

numProperties
On output, the number of properties actually contained in the dictionary.

propertyTag
On output, an array of the property tags contained in the dictionary. You are responsible for allocating an array of the appropriate size.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
The function `DCMGetDictionaryPropertyList` returns a list of property tags in the specified dictionary. You need to call this function twice. The first time you call the function to get the number of properties. Then you must allocate a `propertyTag` array of the appropriate size. You call the function a second time to get the actual list of tags.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetDictionaryWriteAccess

Obtains write access for an open dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetDictionaryWriteAccess (
   DCMDictionaryRef dictionaryRef,
   Duration timeOutDuration
);
```

Parameters

dictionaryRef

A reference to the dictionary for which you want to obtain write access. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

timeOutDuration

The maximum amount of time to wait for write access. This parameter is currently not used.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27). Returns the result `dcmPermissionErr` if the dictionary is read-only dictionary or if another application has write access.

Discussion

When you call the function `DCMOpenDictionary` (page 59), the dictionary opens with read-only access. You must call the function `DCMGetDictionaryWriteAccess` to obtain write privileges. You can obtain write access only if a dictionary is already opened and no other application has write access to that dictionary. You should release write privileges as soon as you no longer need write access, by calling the function `DCMReleaseDictionaryWriteAccess` (page 61).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMGetFieldAttributes

Obtains the field attributes for a field information record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFieldAttributes (
   const AEDesc *fieldInfoRecord,
   DCMFieldAttributes *attributes
);
```

Parameters

fieldInfoRecord

On input, points to the field information record whose attributes you want to obtain.

attributes

On output, points to the attributes obtained from the field information record. See "Field Attributes" (page 20)for more information.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetFieldData

Obtains data from one or more fields in a specified record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFieldData (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMUniqueID uniqueID,
   ItemCount numOfData,
   const DCMFieldTag dataTag[],
   AEDesc *dataList
);
```

Parameters

dictionaryRef
A reference to the dictionary that contains the field data you want to obtain. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
A field tag that specifies the data you want to obtain.

keySize
The size of the `keyData` parameter.

keyData
The key data of the record you whose field you want to obtain. You are responsible for allocating this buffer.

uniqueID
The unique ID of the record whose field you want to obtain.

numOfData
The number of data fields tags in the `dataTag` array.

dataTag
A list of the data field to obtain.

dataList
On return, points to a list of obtained data. The data obtained is returned in t as an `AERecord` data structure. You can retrieve data from specific field using the Apple Event Manager function `AEGetKeyPtr`. You are responsible for disposing of the `dataList` array by calling the Apple Event Manager function `AEDisposeDesc`.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetFieldDefaultData

Obtains default data for a field information record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFieldDefaultData (
   const AEDesc *fieldInfoRecord,
   DescType desiredType,
   AEDesc *fieldDefaultData
);
```

Parameters

fieldInfoRecord
On input, points to the field information record whose default data you want to obtain.

desiredType
The data type of the default data.

fieldDefaultData
On output, points to the default data obtained from the field information record.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetFieldFindMethods

Obtains the search methods for a field information record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFieldFindMethods (
   const AEDesc *fieldInfoRecord,
   ItemCount findMethodsArrayMaxSize,
   DCMFindMethod findMethods[],
   ItemCount *actualNumberOfFindMethods
);
```

Parameters

fieldInfoRecord

On input, points to the field information record whose search methods you want to obtain.

findMethodsArrayMaxSize

The number of elements in the `findMethods` array.

findMethods

On output, an array of search methods obtained from the field information record. See "Search Methods" (page 25) for more information.

actualNumberOfFindMethods

On output, the actual number of search methods obtained from the field information array.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMGetFieldMaxRecordSize

Obtains the maximum data size for a field in a dictionary record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFieldMaxRecordSize (
   const AEDesc *fieldInfoRecord,
   ByteCount *maxRecordSize
);
```

Parameters

fieldInfoRecord

On input, points to the field information record whose maximum data size you want to obtain.

maxRecordSize

On output, points to the maximum data size obtained from the field information record.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetFieldTagAndType

Obtains the field tag and type associated with a field information record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFieldTagAndType (
   const AEDesc *fieldInfoRecord,
   DCMFieldTag *fieldTag,
   DCMFieldType *fieldType
);
```

Parameters

fieldInfoRecord
On input, points to the field information record whose field tag and type you want to obtain.

fieldTag
On output, points to the field tag obtained from the field information record.

fieldType
On output, points to the field tag type obtained from the field information record.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetFileFromDictionaryID

Obtains the file specification associated with a dictionary ID. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetFileFromDictionaryID (
   DCMDictionaryID dictionaryID,
   FSSpec *fileRef
);
```

Parameters

dictionaryID
The ID of the dictionary whose file specification you want to obtain. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

fileRef
On output, points to the file specification for the dictionary specified by the `dictionaryID` parameter.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetNextRecord

Obtains the next specified record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetNextRecord (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMUniqueID uniqueID,
   ByteCount maxKeySize,
   ByteCount *nextKeySize,
   LogicalAddress nextKeyData,
   DCMUniqueID *nextUniqueID
);
```

Parameters
dictionaryRef
A reference to the dictionary whose record you want to obtain. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
The field tag that specifies the data you want to obtain.

keySize
The size of the `keyData` parameter. If you pass `0`, the first record in the dictionary is returned.

keyData
The key data of the reference record.

uniqueID
The unique ID of the reference record.

maxKeySize
The size of the buffer for the `nextKeyData` parameter.

nextKeySize
On output, the actual size of the buffer for the `nextKeyData` parameter.

nextKeyData
On output, points to the next key of the specified record. You must allocate this buffer.

nextUniqueID
On output, the unique ID of the found record.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetNthRecord

Return records in a specified order within the dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetNthRecord (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ItemCount serialNum,
   ByteCount maxKeySize,
   ByteCount *keySize,
   LogicalAddress keyData,
   DCMUniqueID *uniqueID
);
```

Parameters
dictionaryRef
A reference to the dictionary whose record you want to obtain. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
The field tag that specifies the data you want to obtain.

serialNum
A value that specifies the location of the record within the dictionary. You can obtain this value by calling the function `DCMGetRecordSequenceNumber` (page 55).

maxKeySize
The maximum size of the `keyData` parameter.

keySize
On output, the size of the `keyData` parameter.

keyData
The key data of the record you want to obtain. You are responsible for allocating this buffer.

uniqueID
On output, the unique ID of the found record.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetPrevRecord

Obtains the previous record. Return the record previous to the specified record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetPrevRecord (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMUniqueID uniqueID,
   ByteCount maxKeySize,
   ByteCount *prevKeySize,
   LogicalAddress prevKeyData,
   DCMUniqueID *prevUniqueID
);
```

Parameters

dictionaryRef
A reference to the dictionary whose record you want to obtain. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
The field tag that specifies the data you want to obtain.

keySize
The size of the `keyData` parameter. If you pass `0`, the last record in the dictionary is returned.

keyData
The key data of the reference record.

uniqueID
The unique ID of the record reference record.

maxKeySize
The size of the buffer for the `prevKeyData` parameter.

prevKeySize
On output, the actual size of the buffer for the `prevKeyData` parameter.

prevKeyData
On output, points to the previous key of the specified record. You must allocate this buffer.

prevUniqueID
On output, the unique ID of the found record.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMGetRecordSequenceNumber

Obtains the sequence number for the specified record in a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMGetRecordSequenceNumber (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMUniqueID uniqueID,
   ItemCount *sequenceNum
);
```

Parameters

dictionaryRef
A reference to the dictionary whose record sequence number you want to obtain. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
The field tag that specifies the data you want to obtain.

keySize
The size of the `keyData` parameter.

keyData
The key data of the record whose sequence number you want to obtain.

uniqueID
The unique ID of the record whose sequence number you want to obtain.

sequenceNum
On output, a value that specifies the order of the record in the dictionary. The first record in a dictionary has the value 1. Subsequent records are numbered sequentially.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMIterateFoundRecord

Retrieves one record from a list of search results. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMIterateFoundRecord (
   DCMFoundRecordIterator recordIterator,
   ByteCount maxKeySize,
   ByteCount *actualKeySize,
   LogicalAddress keyData,
   DCMUniqueID *uniqueID,
   AEDesc *dataList
);
```

Parameters

recordIterator

A reference to list of search results. You obtain a list of search results by calling the function DCMFindRecords (page 41).

maxKeySize

The size of the keyData parameter.

actualKeySize

On output, the actual size of the buffer needed for the data specified by the keyData parameter.

keyData

On output, the key of the retrieved data.

uniqueID

On output, the unique ID of the retrieved record. This value is guaranteed to be unique among records with the same key data; but it is not unique among all records in the dictionary. You can use the unique ID in conjunction with the retrieved key data to specify individual records within the dictionary when you call the functions DCMGetNextRecord (page 52), DCMGetPrevRecord (page 54), DCMGetRecordSequenceNumber (page 55), DCMDeleteRecord (page 38), DCMGetFieldData (page 48), and DCMSetFieldData (page 64).

dataList

On output, the data associated with the key specified by the keyData parameter.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27). The function returns the result dcmIterationCompleteErr when the final record is retrieved.

Discussion

The function DCMIterateFoundRecord retrieves one record from a list of search results referenced by a record iterator. You can retrieve all of the records referenced by a record iterator by repeatedly calling the DCMIterateFoundRecord function.

The data associated with the fields specified in the preFetchedData parameter is returned to dataList in the form of an AERecord. (See DCMFindRecords (page 41) for more information on pre-fetched data). It is possible to retrieve data by specifying the field tag and data type, and use the AEGetKeyPtr and so forth of the Apple Event Manager. You application is responsible for disposing of dataList by calling the Apple Event Manager function AEDisposeDesc.

Only pre-fetched data can be retrieved here since these data are already retrieved. Other data can be retrieved later by calling the function DCMGetFieldData (page 48).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMIterateObject

Obtains the object ID for a dictionary from a list of available dictionaries. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMIterateObject (
   DCMObjectIterator iterator,
   DCMObjectID *objectID
);
```

Parameters

iterator

The list of available dictionaries. You obtain this list by calling the function `DCMCreateDictionaryIterator` (page 35) or `DCMCreateAccessMethodIterator` (page 35).

objectID

On output, the object ID of the dictionary.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
The first time you call the function `DCMIterateObject` it retrieves the object ID of the first dictionary in the list. The next time you call the function, it retrieves the object ID of the next dictionary in the list. You can obtain all object IDs by repeatedly calling this function. If you call the function after you obtain the object ID for the last dictionary, the function returns the result `dcmIterationCompleteErr`.

You can reset the iterator to the first dictionary in the list by calling the function `DCMResetObjectIterator` (page 62). When you no longer need the iterator, you must dispose of it by calling the function `DCMDisposeObjectIterator` (page 40).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMLibraryVersion

Obtains the version number of the Dictionary Manager. (Deprecated in Mac OS X v10.5.)

```
UInt32 DCMLibraryVersion (
   void
);
```

Parameters

Return Value
The library version number.

Discussion
The function `DCMLibraryVersion` returns the version of the installed Dictionary Manager in the same format as the `'vers'` resource. That is, the version number is returned in Binary-Coded Decimal (BCD) format in the high-order word, and the release stage information is returned in the low-order word. For example, the version 1.1.1 library returns 0x01118000.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMNewDictionary

Creates a new dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMNewDictionary (
   DCMAccessMethodID accessMethodID,
   const FSSpec *newDictionaryFile,
   ScriptCode scriptTag,
   const AEDesc *listOfFieldInfoRecords,
   Boolean invisible,
   ItemCount recordCapacity,
   DCMDictionaryID *newDictionary
);
```

Parameters

accessMethodID
The ID of access method to use for the dictionary. You can obtain an access method ID by calling the function `DCMGetAccessMethodIDFromName` (page 42).

newDictionaryFile
On output, a pointer to an `FSSpec` structure that specifies the dictionary file to be created.

scriptTag
The code of the script system in which the filename of the dictionary file is to be displayed.

listOfFieldInfoRecords

A pointer to an array of `AEDesc` data structures. The `AEDesc` data structure contains two fields: A four-character code that specifies the type of data in the structure and an opaque storage type that points to the storage for the descriptor data. Each of the data structures in the `listOfFieldInfoRecords` array specifies a field name (the four-character code) in the dictionary record and the data associated with that field. You must first call the function `DCMCreateFieldInfoRecord` (page 36) to create this array of data structures.

invisible

A Boolean value that specifies whether the dictionary is available through the Dictionary Manager. Pass `true` if you do not want the dictionary to be available, `false` otherwise. If you set invisible to `true`, that dictionary is no longer seen by such functions as `DCMCreateDictionaryIterator`, so it becomes a dictionary that cannot be accessed from any application other than the application that created the dictionary.

recordCapacity

The number of records that can be stored in the dictionary. You can supply an approximate value if you do not know the exact number.

newDictionary

On output, points to the ID of the newly-created dictionary.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
The function `DCMNewDictionary` creates a new dictionary file and registers that dictionary with the Dictionary Manager. You need to specify the access method to use in creating the dictionary. The Dictionary Manager does not operate directly on a dictionary. Instead it accesses dictionaries using the access method specified for the dictionary. The access method mediates between the dictionary and the Dictionary Manager, so that the Dictionary Manager does not need to know anything about the physical format of the dictionary. As a result, the dictionary can have an free-form internal structure. You can also to use an existing dictionary as long as the dictionary has its own access method.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMOpenDictionary

Opens a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMOpenDictionary (
   DCMDictionaryID dictionaryID,
   ByteCount protectKeySize,
   ConstLogicalAddress protectKey,
   DCMDictionaryRef *dictionaryRef
);
```

Parameters

dictionaryID

The ID of the dictionary you want to open. You obtain a dictionary ID when you register a dictionary by calling the function `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

protectKeySize

The size of the keyword specified by the `protectKey` parameter. This parameter is optional. Pass `0` if you do not plan to provide a password.

protectKey

The keyword to use when opening the dictionary. An access method can use the `protectKey` parameter as a password to restrict access to the dictionary. This parameter is optional. Pass `NULL` if you do not plan to provide a password.

dictionaryRef

On output, a reference to the opened dictionary.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion

The function `DCMOpenDictionary` opens the dictionary specified by the dictionary ID and obtains a reference to the dictionary (`dictionaryRef`). You can pass this reference as a parameter to other Dictionary Manager functions to access the records in the dictionary.

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMRegisterDictionaryFile

Registers a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMRegisterDictionaryFile (
   const FSSpec *dictionaryFile,
   DCMDictionaryID *dictionaryID
);
```

Parameters

dictionaryFile

The file specification for the dictionary you want to register.

dictionaryID

On output, points to the ID of the registered dictionary. A dictionary ID is not persistent across system restarts.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
You should use this function only when the target dictionary is not in default location (see`DCMCreateDictionaryIterator` (page 35)). Otherwise, dictionaries should be already registered and you can obtain the `dictionaryID` by calling the function `DCMGetDictionaryIDFromFile` (page 44) or `DCMCreateDictionaryIterator` (page 35).

You can only use registered dictionaries. You pass the dictionary ID obtained from the function `DCMRegisterDictionaryFile` when you call other Dictionary Manager functions.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMReleaseDictionaryWriteAccess

Releases write access to a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMReleaseDictionaryWriteAccess (
   DCMDictionaryRef dictionaryRef,
   Boolean commitTransaction
);
```

Parameters

dictionaryRef

A reference to the dictionary for which you want to release write access. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

commitTransaction

A Boolean value that specifies whether or not to write changed to the dictionary. Pass `true` to write changes to the dictionary. Pass `false` to cancel changes. The dictionary must support transaction processing for this parameter to have an effect. This parameter is currently not used.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMReorganizeDictionary

Reorganizes a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMReorganizeDictionary (
   DCMDictionaryID dictionaryID,
   ItemCount extraCapacity,
   DCMProgressFilterUPP progressProc,
   UInt32 userData
);
```

Parameters

dictionaryID
The ID of dictionary you want to reorganize. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

extraCapacity
The number of additional records you want to add. This number can be approximate.

progressProc
A universal procedure pointer (UPP) to a progress callback function. This callback is not supported.

userData
Data needed by your progress callback function.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion
The function `DCMReorganizeDictionary` reorganizes the contents of the dictionary specified by the `dictionaryID` parameter, expanding the dictionary to allow for the additional number of records specified by the `extraCapacity` parameter.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMResetObjectIterator

Resets an iterator to the start of the dictionary list. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMResetObjectIterator (
   DCMObjectIterator iterator
);
```

Parameters

iterator

The list of available dictionaries. You obtain this list by calling the function `DCMCreateDictionaryIterator` (page 35) or `DCMCreateAccessMethodIterator` (page 35).

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27).

Discussion

If you want to retrieve the object ID for a dictionary in the list, call the function `DCMIterateObject` (page 57). When you no longer need the iterator, you must dispose of it by calling the function `DCMDisposeObjectIterator` (page 40).

Availability

Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Dictionary.h`

DCMSetDictionaryProperty

Sets a property for a dictionary. Set the properties to a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMSetDictionaryProperty (
   DCMDictionaryID dictionaryID,
   DCMFieldTag propertyTag,
   ByteCount propertySize,
   ConstLogicalAddress propertyValue
);
```

Parameters

dictionaryID

The ID of dictionary whose property you want to set. You obtain a dictionary ID when you call the functions `DCMRegisterDictionaryFile` (page 60) or `DCMGetDictionaryIDFromFile` (page 44).

propertyTag

The property tag whose data you want to set.

propertySize

The size of data pointed to by the `propertyValue` parameter.

propertyValue

A pointer to the property data to be set.

Return Value

A result code. See "Dictionary Manager Result Codes" (page 27). Returns the result `dcmPermissionErr` if the property already exists and it is a read-only property.

Discussion
If the specified properties already exists and it is a writable property, the property data is replaced. If the property does not exist, it is created.

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMSetFieldData

Set the data to a specific field of a specified record. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMSetFieldData (
   DCMDictionaryRef dictionaryRef,
   DCMFieldTag keyFieldTag,
   ByteCount keySize,
   ConstLogicalAddress keyData,
   DCMUniqueID uniqueID,
   const AEDesc *dataList
);
```

Parameters

dictionaryRef
A reference to the dictionary that contains the field data you want to set. You obtain a dictionary reference when you call the function `DCMOpenDictionary` (page 59).

keyFieldTag
Tag of applicable key field.

keySize
The size of the `keyData` parameter.

keyData
The key data of the record you whose field you want to set.

uniqueID
The unique ID of the record whose field you want to obtain.

dataList
A pointer to the list of data you want to set. The `AEDesc` data structure contains two fields: A four-character code that specifies the type of data in the structure and an opaque storage type that points to the storage for the descriptor data. Each of the data structures in the `dataList` array specifies a field name (the four-character code) in the dictionary record and the data associated with that field. You must first call the function `DCMCreateFieldInfoRecord` (page 36) to create this array of data structures.

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

DCMUnregisterDictionary

Unregisters a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus DCMUnregisterDictionary (
   DCMDictionaryID dictionaryID
);
```

Parameters
dictionaryID

Return Value
A result code. See "Dictionary Manager Result Codes" (page 27). Returns `dcmDictionaryBusyErr` if the dictionary is in use by another client.

Discussion
You should use this function only for dictionaries that you registered by calling the function `DCMRegisterDictionaryFile` (page 60).

Availability
Available in CarbonLib 1.0 and later when running Japanese Mac OS 8.5 or later, or other Mac OS 8.5 or later with the Japanese Language Kit.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Dictionary.h`

Document Revision History

This table describes the changes to *Dictionary Manager Reference*.

Date	Notes
2007-12-11	Deprecated document, adding to introduction a deprecation statement and pointers to documentation for replacement technology.
2003-02-17	First version of this document. A preliminary version was published previously under the title *Programming With Dictionary Manager*.

Document Revision History

Index

A

D

F

K

L

P

S

T

W