Translation Manager Reference

Carbon > File Management

2006-07-12

Contents

Translation Manager Reference

Framework:	Carbon/Carbon.h
Declared in	Translation.h TranslationExtensions.h

Overview

You can use the Translation Manager to direct the translation of documents from one format to another. For example, Macintosh Easy Open uses the Translation Manager to provide

- automatic translation of a document, if the application that created the document is not available
- automatic translation of documents drop-launched onto an application
- enhanced file-opening dialog boxes and (when necessary) automatic translation of documents the user selects in those dialog boxes
- batch desktop translation of documents
- automatic translation of data pasted from the clipboard

These services allow your application to open documents created by other applications (possibly running on other operating systems) and to import data from other applications with better fidelity than previously possible.

Macintosh Easy Open does not do any translating itself, and it does not have any knowledge of translation data models. Instead, it delegates these functions to translation extensions or to applications with built-in translation capability. Translation extensions and application translation capabilities operate as "black boxes" to Macintosh Easy Open. At system startup (or whenever new translation extensions become available), Macintosh Easy Open catalogs the translation capability of each translation extension and each application, and then invokes each as needed. Macintosh Easy Open can support multiple translation systems.

Carbon supports the Translation Manager in Mac OS 9, with the exception of the functions declared in `TranslationExtensions.h`.

Important: The Translation Manager was deprecated in Mac OS X v10.3. In Mac OS X, Carbon includes the Translation Manager headers but does not implement any of the functionality. If you call the functions declared in the API, they do nothing. You should use Translation Services instead.

Callbacks

DoGetFileTranslationListProcPtr

Defines a pointer to a get file translation list callback function that returns a list of the file types which your extension can translate.

Unsupported

```
typedef ComponentResult (*DoGetFileTranslationListProcPtr)
(
    ComponentInstance self,
    FileTranslationListHandle translationList
);
```

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

```
ComponentResult DoGetFileTranslationListProcPtr
(
    ComponentInstance self,
    FileTranslationListHandle translationList
);
```

Parameters

self

A component instance that identifies the component containing the translation extension.

translationList

On entry to your function, this parameter contains a handle to a structure of type `FileTranslationList` (page 16). If your translation extension can translate any files at all, your function should resize that handle and fill the block with a list of the file types it can translate. If the translation list whose handle you return in this parameter has the `groupCount` field set to 0, Macintosh Easy Open assumes that your extension cannot translate any file types.

For improved performance, Macintosh Easy Open remembers each translation extension's most recently returned file translation list and passes that list to your get file translation list callback function in this parameter. If you determine that the list hasn't changed, you should simply return the same handle to Macintosh Easy Open.

Return Value

If successful, your function should return `noErr`. Otherwise, your function should return an appropriate result code. The Component Manager requires this function to return a value of type `ComponentResult` to simplify dispatching. See the Component Manager documentation for a description of the `ComponentResult` data type.

Discussion

A file translation extension must respond to the `kTranslateGetFileTranslationList` request code. Whenever it first notices the extension, Macintosh Easy Open calls your extension with this request code to obtain a list of the file types that the extension can translate. You can handle this request by calling the Component Manager function `CallComponentFunctionWithStorage` and passing it a pointer to a your get file translation list callback function.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

DoGetScrapTranslationListProcPtr

Defines a pointer to a get-scrap-translation-list callback function that returns a list of the scrap types that your extension can translate.

Unsupported

```
typedef ComponentResult (*DoGetScrapTranslationListProcPtr)
(
    ComponentInstance self,
    ScrapTranslationListHandle list
);
```

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

```
ComponentResult DoGetScrapTranslationListProcPtr
(
    ComponentInstance self,
    ScrapTranslationListHandle list
);
```

Parameters

`self`

A component instance that identifies the component containing the translation extension.

`list`

A handle to a `ScrapTranslationList` (page 18). Your function should return, through this parameter, a handle to a list of the scrap types from and into which your translation extension can translate. If your translation extension can translate any scrap types at all, your function should resize this handle and fill the block with a list of the scrap types it can translate. If the translation list whose handle you return in this parameter has the `groupCount` field set to 0, Macintosh Easy Open assumes that your extension cannot translate any scrap types.

On entry to your function, this parameter contains a handle to a structure of type `ScrapTranslationList`. When it first becomes aware of your extension, Macintosh Easy Open calls your translation extension's get scrap translation list function. For improved performance, Macintosh Easy Open remembers each translation extension's most recently returned scrap translation list and passes that list to your function in this parameter. If you determine that the list hasn't changed, you should simply return the same handle to Macintosh Easy Open.

Return Value

If successful, your function should return `noErr`. Otherwise, your function should return an appropriate result code. The Component Manager requires this function to return a value of type `ComponentResult` to simplify dispatching. See the Component Manager documentation for a description of the `ComponentResult` data type.

Discussion

A scrap translation extension must respond to the `kTranslateGetScrapTranslationList` request code. At system startup time, the Translation Manager calls your extension with this code. You can handle this request by calling the Component Manager function `CallComponentFunctionWithStorage` and passing it a pointer to your get scrap translation list callback function.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

DoGetTranslatedFilenameProcPtr

Defines a pointer to a get-translated-filename callback function.

Unsupported

```
typedef ComponentResult (*DoGetTranslatedFilenameProcPtr)
(
    ComponentInstance self,
    FileType dstType,
    long dstTypeHint,
    FSSpec * theDocument
);
```

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

```
ComponentResult DoGetTranslatedFilenameProcPtr
```

```
(
    ComponentInstance self,
    FileType dstType,
    long dstTypeHint,
    FSSpec * theDocument
);
```

Parameters

self

dstType

theDocument

Return Value

See the Component Manager documentation for a description of the `ComponentResult` data type.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

DoIdentifyFileProcPtr

Defines a pointer to a file identification callback function that identifies a file as having a format which your extension can translate.

Unsupported

```
typedef ComponentResult (*DoIdentifyFileProcPtr)
(
    ComponentInstance self,
    const FSSpec * theDocument,
    FileType * docType
);
```

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

```
ComponentResult DoIdentifyFileProcPtr (
    ComponentInstance self,
    const FSSpec * theDocument,
    FileType * docType
);
```

Parameters

self

A component instance that identifies the component containing the translation extension.

theDocument

A pointer to a file system specification structure that specifies the document that the translation extension must identify.

docType

Your function should return, in this parameter, the file format type of the file specified in the `theDocument` parameter.

Your function should not return `'TEXT'` as a file type unless you determine that the document consists solely of a plain, unformatted stream of ASCII characters.

Return Value

If successful, your function should return `noErr`. Otherwise, your function should return an appropriate result code. If your translation extension does not recognize the type of the specified file, your function should return the result code `noTypeErr`. The Component Manager requires this function to return a value of type `ComponentResult` to simplify dispatching. See the Component Manager documentation for a description of the `ComponentResult` data type.

Discussion

A file translation extension must respond to the `kTranslateIdentifyFile` request code. The Translation Manager uses this request code to allow the translation extension to identify a file as having a format that the extension can translate. You can handle this request by calling the Component Manager function `CallComponentFunctionWithStorage` and passing it a pointer to your file identification callback function.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

DoIdentifyScrapProcPtr

Defines a pointer to a scrap identification callback function that identifies a scrap as one that your scrap translation extension can translate.

Unsupported

```
typedef ComponentResult (*DoIdentifyScrapProcPtr)
(
    ComponentInstance self,
    const void * dataPtr,
    Size dataLength,
    ScrapType * dataFormat
);
```

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

```
ComponentResult DoIdentifyScrapProcPtr
(
    ComponentInstance self,
    const void * dataPtr,
    Size dataLength,
    ScrapType * dataFormat
);
```

Parameters

`self`

A component instance that identifies the component containing the translation extension.

`dataPtr`

A pointer to the scrap to translate.

`dataLength`

The size of the scrap to translate.

`dataFormat`

On entry, the type of the scrap format. Your function returns, through this parameter, the type of the scrap format of the scrap specified by the `dataPtr` parameter, as recognized by your translation extension.

In general, the scrap that your `DoIdentifyScrap` function is asked to identify is always in one of the formats listed among the source formats in the translation groups contained in your extension's scrap translation list. Your scrap translation extension therefore needs only to verify that the indicated scrap is of the specified format.

Return Value

If successful, your function should return `noErr`. Otherwise, your function should return an appropriate result code. If your translation extension does not recognize the type of the specified scrap, your function should return the result code `noTypeErr`. The Component Manager requires this function to return a value of type `ComponentResult` to simplify dispatching. See the Component Manager documentation for a description of the `ComponentResult` data type.

Discussion

A scrap translation extension must respond to the `kTranslateIdentifyScrap` request code. You can handle this request by calling the Component Manager function `CallComponentFunctionWithStorage` and passing it a pointer to a your scrap identification callback function.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

DoTranslateFileProcPtr

Defines a pointer to a file translation callback function that translates a document from one format into another.

Unsupported

```
typedef ComponentResult (*DoTranslateFileProcPtr)
(
    ComponentInstance self,
    TranslationRefNum refNum,
    const FSSpec * sourceDocument,
    FileType srcType,
    long srcTypeHint,
    const FSSpec * dstDoc,
    FileType dstType,
    long dstTypeHint
);
```

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

```
ComponentResult DoTranslateFileProcPtr
(
    ComponentInstance self,
    TranslationRefNum refNum,
    const FSSpec * sourceDocument,
    FileType srcType,
    long srcTypeHint,
    const FSSpec * dstDoc,
    FileType dstType,
    long dstTypeHint
);
```

Parameters

self

A component instance that identifies the component containing your translation extension.

refNum

The translation reference number for this translation.

Macintosh Easy Open assigns this reference number to the translation. Each translation is assigned a unique number to distinguish the translation from any other translations that might occur. You need to pass this reference number to any Macintosh Easy Open functions you call from within the file translation extension; for instance, if by calling the `SetTranslationAdvertisement` function you display the progress dialog box, you'll pass that reference number in the `refNum` parameter.

sourceDocument

A file system specification structure that specifies the document to translate.

srcType

The format of the file to be translated.

srcTypeHint

The value in the `hint` field of the source document's file type specification.

dstDoc

A file system specification structure that specifies the destination document.

Your function should put the translated document into the file specified by this parameter. The data fork of the destination file already exists by the time your function is called. In addition, if the `flags` field in the appropriate destination file type specification in your extension's file translation list has the `taDstDocNeedsResourceFork` bit set, the destination file already contains a resource fork. Your function should open the destination file and fill its data or resource fork (or both) with the appropriate translated data.

dstType

The format into which to translate the source document.

dstTypeHint

The value in the `hint` field of the destination document's file type specification.

Return Value

If successful, your function should return `noErr`. Otherwise, your function should return an appropriate result code. If it cannot translate the source file, your function should return a result code different from `noErr`. In that case, Macintosh Easy Open will automatically delete the destination file. The Component Manager requires this function to return a value of type `ComponentResult` to simplify dispatching. See the Component Manager documentation for a description of the `ComponentResult` data type.

Discussion

A file translation extension must respond to the `kTranslateTranslateFile` request code. You can handle this request by calling the Component Manager function `CallComponentFunctionWithStorage` and passing it a pointer to your file translation function.

Your file translation function can translate the source file itself or rely upon external translators.

Your translation extension should call the `SetTranslationAdvertisement` (page 30) function to display the progress dialog box and the `UpdateTranslationProgress` (page 33) function to update the dialog box periodically.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

DoTranslateScrapProcPtr

Defines a pointer to a scrap translation callback function that translates a scrap from one format into another.

Unsupported

```
typedef ComponentResult (*DoTranslateScrapProcPtr)
(
    ComponentInstance self,
    TranslationRefNum refNum,
    const void * srcDataPtr,
    Size srcDataLength,
    ScrapType srcType,
    long srcTypeHint,
    Handle dstData,
    ScrapType dstType,
    long dstTypeHint
);
```

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

```
ComponentResult DoTranslateScrapProcPtr
(
    ComponentInstance self,
    TranslationRefNum refNum,
    const void * srcDataPtr,
    Size srcDataLength,
    ScrapType srcType,
    long srcTypeHint,
    Handle dstData,
    ScrapType dstType,
    long dstTypeHint
);
```

Parameters

self

A component instance that identifies the component containing the translation extension.

refNum

The translation reference number for this translation.

Macintosh Easy Open assigns this reference number to the translation. Each translation is assigned a unique number to distinguish the translation from any other translations that might be occurring. You need to pass this reference number to any Macintosh Easy Open functions you call from within the scrap translation extension; for instance, if you display the progress dialog box by calling the `SetTranslationAdvertisement` function, you'll pass that reference number in the `refNum` parameter.

srcDataPtr

A pointer to the scrap to translate.

srcDataLength

The size of the scrap to translate.

srcType

The format of the scrap to translate.

srcTypeHint

The value in the `hint` field of the source document's scrap type specification.

dstData

A handle to the destination to be filled in. Your function should put the translated data into the block specified here, resizing it as necessary.

dstType

The format into which to translate the source scrap.

dstTypeHint

The value in the `hint` field of the destination document's scrap type specification.

Return Value

If successful, your function should return `noErr`. Otherwise, your function should return an appropriate result code. The Component Manager requires this function to return a value of type `ComponentResult` to simplify dispatching. See the Component Manager documentation for a description of the `ComponentResult` data type.

Discussion
A scrap translation extension must respond to the `kTranslateTranslateScrap` request code. You can handle this request by calling the Component Manager function `CallComponentFunctionWithStorage` and passing it a pointer to your function. Your scrap translation callback function can translate the source file itself or rely upon external translators.

Your translation extension should call the `SetTranslationAdvertisement` (page 30) function to display the progress dialog box and the `UpdateTranslationProgress` (page 33) function to update the dialog box periodically.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

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

Declared In
`TranslationExtensions.h`

GetScrapDataProcPtr

Defines a pointer to a source-data fetching callback.

```
typedef OSErr (*GetScrapDataProcPtr)
(
    ScrapType requestedFormat,
    Handle dataH,
    void * srcDataGetterRefCon
);
```

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

```
OSErr GetScrapDataProcPtr (
    ScrapType requestedFormat,
    Handle dataH,
    void * srcDataGetterRefCon
);
```

Parameters

`requestedFormat`
Format of data that `TranslateScrap` (page 32) needs.

`dataH`
The handle in which to put the requested data.

Return Value
A result code. See "Translation Manager Result Codes" (page 20).

Discussion
The first time this function is call ed, you should resize and fill in the handle with a list all the formats that you have available to be translated, and the length of each. When called again, you should supply the data in one of the formats in the list.

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

Declared In
`Translation.h`

Data Types

FileTranslationList

Defines a list of file formats an extension can translate.

```
struct FileTranslationList {
  unsigned long modDate;
  unsigned long groupCount;
};
typedef struct FileTranslationList  FileTranslationList;
typedef FileTranslationList *       FileTranslationListPtr;
typedef FileTranslationListPtr *    FileTranslationListHandle;
```

Fields

`modDate`
: The creation date of the file translation list. If your extension uses external translators, you might set this field to the modification date of a folder containing those translators.

`groupCount`
: The number of translation groups that follow.

Discussion
The Translation Manager uses the file translation list that it gets from each translation system to create a master database of format translations it can direct. A file translation list consists of a field indicating the modification date of the list and a count of the number of groups that follow those two fields.

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

Declared In
`TranslationExtensions.h`

FileTranslationSpec

Defines a file translation method supported by the Translation Manager.

```
struct FileTranslationSpec {
  OSType          componentSignature;
  const void *    translationSystemInfo;
  FileTypeSpec    src;
  FileTypeSpec    dst;
};
typedef struct FileTranslationSpec      FileTranslationSpec;
typedef FileTranslationSpec *           FileTranslationSpecArrayPtr;
typedef FileTranslationSpecArrayPtr *   FileTranslationSpecArrayHandle;
```

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

Declared In
`Translation.h`

FileType

Defines the translation file type of a document.

```
typedef OSType FileType;
```

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

Declared In
`TranslationExtensions.h`

FileTypeSpec

Defines a descriptor for the source or destination file type used in a translation method.

```
struct FileTypeSpec {
    FileType format;
    long hint;
    TranslationAttributes flags;
    OSType catInfoType;
    OSType catInfoCreator;
};
typedef struct FileTypeSpec FileTypeSpec;
```

Fields

`format`
: The translation file type of the document. Macintosh Easy Open uses this field as the canonical way to describe the format of a file for translation purposes.

`hint`
: A 4-byte value reserved for use by your translation extension.

`flags`
: A 4-byte value consisting of bit flags that specify how to control the translation. This field is used only for destination file types; you should set it to 0 for all source file type specifications. Currently 2 bits are defined; all other bits should be cleared to 0.

`catInfoType`
: The type of the file as contained in the volume's catalog file.

`catInfoCreator`

The creator of the file as contained in the volume's catalog file.

Discussion

The `FileTranslationList` (page 16) structure uses file type specifications to describe document formats. A file type specification is defined by the `FileTypeSpec` data structure.

The interpretation of some of the fields of a file type specification depends on whether the specification occurs in the list of source document types or in the list of destination document types:

In file type specifications occurring in the list of source document types in a file translation list, Macintosh Easy Open uses the `format` and `catInfoCreator` fields to determine the kind string displayed in the "From" format specification of the translation progress dialog box.

In file type specifications occurring in the list of destination document types in a file translation list, Macintosh Easy Open uses the `format` and `catInfoCreator` fields to determine the kind string displayed in the "To" format specification in the translation progress dialog box. The `format` and `catInfoCreator` fields are also used to get the information displayed in the Document Converter dialog box. However, Macintosh Easy Open uses the `catInfoType` and `catInfoCreator` fields to set the catalog type and creator of the destination file.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TranslationExtensions.h`

ScrapTranslationList

Defines the scrap formats an extension can translate.

```
struct ScrapTranslationList {
  unsigned long   modDate;
  unsigned long   groupCount;
};
typedef struct ScrapTranslationList     ScrapTranslationList;
typedef ScrapTranslationList *          ScrapTranslationListPtr;
typedef ScrapTranslationListPtr *       ScrapTranslationListHandle;
```

Fields

`modDate`

The creation date of the scrap translation list. If your extension uses external translators, you might set this field to the modification date of a folder containing those translators.

`groupCount`

The number of translation groups that follow. The size of the translation list prepared by an extension is variable, depending upon the number of groups, the scrap specification structure size, and the number of scrap types that the extension knows about.

Discussion

The Translation Manager uses the scrap translation list that it gets from each translation system to create a master database of its translation capability. A scrap translation list consists of a field indicating the modification date of the list and a count of the number of groups that follow those two fields.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`TranslationExtensions.h`

ScrapType

Defines the scrap type in a Translation Manager scrap format.

```
typedef ResType ScrapType;
```

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

Declared In
`TranslationExtensions.h`

ScrapTypeSpec

Describes a specific scrap format.

```
struct ScrapTypeSpec {
    ScrapType format;
    long hint;
};
typedef struct ScrapTypeSpec ScrapTypeSpec;
```

Fields

`format`

The type of the specified scrap.

`hint`

A 4-byte value reserved for use by your translation extension.

Discussion
The `ScrapTypeSpec` data structure is used by the `ScrapTranslationList` (page 18) structure.

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

Declared In
`TranslationExtensions.h`

TypesBlock

Defines a null-terminated array of `OSType` or `FileType` elements.

```
    typedef OSType      TypesBlock[64];
    typedef OSType *    TypesBlockPtr;
```

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

Declared In
`Translation.h`

Constants

DocOpenMethod

Specifies the ways a document can be opened.

```
enum {
  domCannot = 0,
  domNative = 1,
  domTranslateFirst = 2,
  domWildcard = 3
};
typedef short DocOpenMethod;
```

Constants

`domCannot`

Indicates that the application cannot open the document.

Available in Mac OS X v10.0 and later.

Declared in `Translation.h`.

`domNative`

Indicates that the application can open the document natively.

Available in Mac OS X v10.0 and later.

Declared in `Translation.h`.

`domTranslateFirst`

Indicates that the application can open the document only after it's been translated.

Available in Mac OS X v10.0 and later.

Declared in `Translation.h`.

`domWildcard`

Indicates that the application has the file type '****' in its list of the file types that it can open and hence can open any type of document.

Available in Mac OS X v10.0 and later.

Declared in `Translation.h`.

Discussion

The `CanDocBeOpened` (page 23) function uses the following constants to specify the method for opening a given document.

Result Codes

The most common result codes returned by the Translation Manager are listed in the table below.

Result Code	Value	Description
`invalidTranslationPathErr`	-3025	Source type to destination type not a valid path. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`couldNotParseSourceFileErr`	-3026	Source document does not contain source type. Available in Mac OS X v10.0 and later.
`noTranslationPathErr`	-3030	Application cannot open document. Available in Mac OS X v10.0 and later.
`badTranslationSpecErr`	-3031	Translation path is invalid. Available in Mac OS X v10.0 and later.
`noPrefAppErr`	-3032	No translation preference available. Available in Mac OS X v10.0 and later.

Gestalt Constants

You can check for version and feature availability information by using the Translation Manager selectors defined in the Gestalt Manager. For more information, see *Inside Mac OS X: Gestalt Manager Reference*.

Deprecated Translation Manager Functions

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

Deprecated in Mac OS X v10.3

CanDocBeOpened

Determines whether a specified application can open a particular document. (Deprecated in Mac OS X v10.3. Use Launch Services to determine whether a document can be opened. See *Launch Services Programming Guide*.)

Not Recommended

```
OSErr CanDocBeOpened (
   const FSSpec *targetDocument,
   short appVRefNumHint,
   OSType appSignature,
   const FileType *nativeTypes,
   Boolean onlyNative,
   DocOpenMethod *howToOpen,
   FileTranslationSpec *howToTranslate
);
```

Parameters

targetDocument
: A pointer to the document to check.

appVRefNumHint
: The volume reference number of the volume containing the application. The search for the specified application begins on this volume if the application isn't found there, the search continues to other mounted volumes.

appSignature
: The signature of the application.

nativeTypes
: A pointer to the zero-terminated list of file types that the application can open without translation; if this parameter contains `NULL`, the default list of file types returned by the `GetFileTypesThatAppCanNativelyOpen` (page 27) function is used.

onlyNative
: If `TRUE`, determine only whether the application can open the document without translation; otherwise, determine whether the application can open the document after translation.

howToOpen

On return, a pointer to a constant indicating the method of opening the document. This field contains a meaningful value only if the function returns `noErr` (indicating that the specified document can be opened). See "DocOpenMethod" (page 20) for a description of the values you can use here.

howToTranslate

On return, if the document needs to be translated before it can be opened, a pointer to a buffer of information (in a private format) indicating how to translate the document. You pass the information returned in this parameter to the `TranslateFile` (page 31) function.

Return Value

A result code. If the application can open the document, the function returns `noErr`.

Discussion

A preference must have already been set (using the Document Converter tool) on how to open the document.

Special Considerations

This function might cause memory to be moved or purged; you should not call it at interrupt time.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

DisposeGetScrapDataUPP

Disposes of a universal procedure pointer (UPP) to a source-data fetcher callback. (Deprecated in Mac OS X v10.3. There is no replacement function. You should adopt Translation Services instead.)

```
void DisposeGetScrapDataUPP (
   GetScrapDataUPP userUPP
);
```

Parameters

userUPP

The universal procedure pointer.

Discussion

See the callback `GetScrapDataProcPtr` (page 15) for more information.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Declared In

`Translation.h`

ExtendFileTypeList

Creates a list of file types that can be translated into a type in a given list. (Deprecated in Mac OS X v10.3. Use the Translation Services function `TranslationCreateWithSourceArray` instead.)

Not Recommended

```
OSErr ExtendFileTypeList (
   const FileType *originalTypeList,
   short numberOriginalTypes,
   FileType *extendedTypeList,
   short *numberExtendedTypes
);
```

Parameters

`originalTypeList`
: A pointer to a list of file types that can be opened.

`numberOriginalTypes`
: The number of file types in the `originalTypeList` parameter.

`extendedTypeList`
: On return, a pointer to a buffer filled with file types that can be translated into the types in `originalTypeList`.

`numberExtendedTypes`
: On input, a pointer to the maximum number of file types that can be put into the buffer passed in the `extendedTypeList` parameter. On return, a pointer to the actual number of file types put into the extended type list.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Discussion

Note that the number of types specified in the parameters `numberOriginalTypes` and `numberExtendedTypes` is limited only by available memory.

The Standard File Package calls this function internally your application probably won't need to use it.

Special Considerations

This function might cause memory to be moved or purged; you should not call it at interrupt time.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

GetDocumentKindString

Allows you to get a document kind string. (Deprecated in Mac OS X v10.3. There is no replacement function, but you can use Launch Services to obtain similar information. See *Launch Services Programming Guide*.)

Not Recommended

```
OSErr GetDocumentKindString (
   short docVRefNum,
   OSType docType,
   OSType docCreator,
   Str63 kindString
);
```

Parameters

docVRefNum

The volume containing the document. This is a hint to the Translation Manager. If it doesn't find the string on that volume, it will use an internal search path to look on other volumes for the string.

docType

The type code of the document you want to query.

docCreator

The creator code of the document you want to query.

kindString

Upon return, contains the kind string to display for the specified document type and creator.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

GetFileTranslationPaths

A low-level routine that allows you to get all the translation capabilities of the Translation Manager. (Deprecated in Mac OS X v10.3. Use the Translation Services function `TranslationCreateWithSourceArray` instead.)

Not Recommended

```
short GetFileTranslationPaths (
   const FSSpec *srcDocument,
   FileType dstDocType,
   unsigned short maxResultCount,
   FileTranslationSpecArrayPtr resultBuffer
);
```

Parameters

srcDocument

A source document (optional), or `NULL`.

dstDocType

The desired document type to which you would like `srcDocument` translated.

resultBuffer

The requested translation information.

Return Value

Returns the number of translation paths, or a result code if the value is negative.

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

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In
`Translation.h`

GetFileTypesThatAppCanNativelyOpen

Obtains a list of file types that an application can open by itself. (Deprecated in Mac OS X v10.3. There is no replacement function. Applications should use Translation Services instead.)

Not Recommended

```
OSErr GetFileTypesThatAppCanNativelyOpen (
   short appVRefNumHint,
   OSType appSignature,
   FileType *nativeTypes
);
```

Parameters

appVRefNumHint
: The volume reference number of the volume containing the application. The search for the specified application begins on this volume if the application isn't found there, the search continues to other mounted volumes.

appSignature
: The signature of the application.

nativeTypes
: On return, a pointer to an array of file types that the application can open without translation. If successful, the array contains up to 64 file types. If fewer than 64 types are returned, the end of the list is indicated by an entry whose value is 0.

Return Value
A result code. See "Translation Manager Result Codes" (page 20).

Special Considerations

This function might cause memory to be moved or purged; you should not call it at interrupt time.

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

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In
`Translation.h`

GetPathFromTranslationDialog

Displays the Translation Dialog, which attempts to generate a list of translation paths resulting in a document readable by the target application. (Deprecated in Mac OS X v10.3. There is no replacement function. However, applications can obtain this information using Launch Services, Translation Services, and uniform type identifiers.)

Not Recommended

```
OSErr GetPathFromTranslationDialog (
   const FSSpec *theDocument,
   const FSSpec *theApplication,
   TypesBlockPtr typeList,
   DocOpenMethod *howToOpen,
   FileTranslationSpec *howToTranslate
);
```

Parameters

`theDocument`
: The target file.

`theApplication`
: The target application.

`typeList`
: Specifies a list of file types into which to translate the target document.

`howToOpen`
: Upon return, contains the translation open method.

`howToTranslate`
: Upon return, contains the translation specification.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Special Considerations

For information about using Launch Services and uniform type identifiers, see *Launch Services Programming Guide* and *Uniform Type Identifiers Overview*.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

GetTranslationExtensionName

Finds the name of the extension performing the translation. (Deprecated in Mac OS X v10.3. There is no replacement function. However, you can obtain useful user-level information by calling the Translation Manager function `TranslationCopyDestinationType` and the uniform type identifiers function `UTTypeCopyDescription`.)

Not Recommended

```
OSErr GetTranslationExtensionName (
   const FileTranslationSpec *translationMethod,
   Str31 extensionName
);
```

Parameters

translationMethod

A file translation method obtained by calling `CanDocBeOpened` or `GetFileTranslationPaths`.

extensionName

Upon return, contains the name of the extension performing the translation.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

InvokeGetScrapDataUPP

Calls a source-data fetcher callback. (Deprecated in Mac OS X v10.3. There is no replacement function. You should adopt Translation Services instead.)

```
OSErr InvokeGetScrapDataUPP (
   ScrapType requestedFormat,
   Handle dataH,
   void *srcDataGetterRefCon,
   GetScrapDataUPP userUPP
);
```

Discussion

You should not need to use the function `InvokeGetScrapDataUPP`, as the system calls your source-data fetcher callback for you. See the callback `GetScrapDataProcPtr` (page 15) for more information.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Declared In

`Translation.h`

NewGetScrapDataUPP

Creates a new universal procedure pointer (UPP) to a source-data fetcher callback. (Deprecated in Mac OS X v10.3. There is no replacement function. You should adopt Translation Services instead.)

```
GetScrapDataUPP NewGetScrapDataUPP (
   GetScrapDataProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your source-data fetcher callback.

Return Value

On return, a UPP to the source-data fetcher callback.

Discussion

See the callback `GetScrapDataProcPtr` (page 15) for more information.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Declared In

`Translation.h`

SetTranslationAdvertisement

Allows your translation extension to install an advertisement into the upper portion of the progress dialog box. (Deprecated in Mac OS X v10.3. There is no replacement function. You should adopt Translation Services instead.)

Unsupported

```
OSErr SetTranslationAdvertisement (
   TranslationRefNum refNum,
   PicHandle advertisement
);
```

Parameters

refNum

A translation reference number. You should set this parameter to the translation reference number passed to your `DoTranslateFileProcPtr` (page 11) or `DoTranslateScrapProcPtr` (page 13) callback functions. The Translation Manager uses that number internally.

advertisement

A handle to a picture to display in the upper portion of the dialog box. If this parameter is `NULL`, no advertisement is displayed and the upper portion of the dialog box is removed before the box is displayed to the user. After the function installs the specified advertisement, it then displays the dialog box.

Your translation extension can read the picture data from its resource fork, but it should detach the resource from the resource fork (by calling the `DetachResource` function) and make the handle unpurgeable before calling this function. Because you'll usually load the picture data into the temporary heap provided for the translation extension, the picture data is automatically disposed of when that heap is destroyed. If your translation extension loads the picture data elsewhere in memory, you are responsible for disposing of it before returning from your `DoTranslateFile` or `DoTranslateScrap` callback function.

The size of the picture to display can be no larger than 280 by 50 pixels. If the picture you specify is smaller than that, it is automatically centered (both vertically and horizontally) in the available space.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Discussion

Your translation extension should call this function only in response to the `kTranslateTranslateFile` or `kTranslateTranslateScrap` request code (that is, you should only call this function in your `DoTranslateFile` or `DoTranslateScrap` callback function). Do not call this function in response to any other request code or from any code that isn't a translation extension.

You must call this function before you call the `UpdateTranslationProgress` (page 33) function for the first time.

Special Considerations

This function might cause memory to be moved or purged; you should not call it at interrupt time.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`TranslationExtensions.h`

TranslateFile

Translates a document from one format to another. (Deprecated in Mac OS X v10.3. Use the Translation Services function `TranslationPerformForFile` or `TranslationPerformForURL` instead.)

Not Recommended

```
OSErr TranslateFile (
   const FSSpec *sourceDocument,
   const FSSpec *destinationDocument,
   const FileTranslationSpec *howToTranslate
);
```

Parameters

sourceDocument

A pointer to the document to translate.

destinationDocument

A pointer to the file to put the translated document into. Note that your application only specifies the name and location for the file; the function creates the file and puts the translated data into it. The destination file must not exist before you call this function.

howToTranslate

A pointer to a buffer of information indicating how to translate the document. Usually, you'll get this information by calling the `CanDocBeOpened` (page 23) function.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Special Considerations

This function might cause memory to be moved or purged; you should not call it at interrupt time.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

TranslateScrap

Uses a client-specified callback function to get the source data to translate. (Deprecated in Mac OS X v10.3. Use the Translation Manager function `TranslationPerformForData` instead.)

Not Recommended

```
OSErr TranslateScrap (
   GetScrapDataUPP sourceDataGetter,
   void *sourceDataGetterRefCon,
   ScrapType destinationFormat,
   Handle destinationData,
   short progressDialogID
);
```

Parameters

sourceDataGetter

The callback that provides the data to translate.

sourceDataGetterRefCon

A generic pointer to private information for your callback.

destinationFormat
: The desired translation format.

destinationData
: A handle you provide. The Translation Extension will automatically re-size it as necessary during translation. Upon exit, if the routine successfully executes, it will contain the translated information.

progressDialogID
: This parameter should always be assigned the value `TranslationScrapProgressDialogID`.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Translation.h`

UpdateTranslationProgress

Allows your translation extension to update the progress dialog box that is displayed during file and scrap translation and give the user a chance to cancel. (Deprecated in Mac OS X v10.3. There is no replacement function. You should adopt Translation Services instead.)

Unsupported

```
OSErr UpdateTranslationProgress (
   TranslationRefNum refNum,
   short percentDone,
   Boolean *canceled
);
```

Parameters

refNum
: A translation reference number. You should set this parameter to the translation reference number passed to your `DoTranslateFileProcPtr` (page 11) or `DoTranslateScrapProcPtr` (page 13) callback function. The Translation Manager uses that number internally.

percentDone
: A value in the range 0–100 that indicates the percentage of the translation that has been completed (the approximate percentage of time elapsed until completion). When the translation is complete, you should call this function with this parameter set to 100 so that the user can see that the translation is complete.

canceled
: On return, a pointer to a value which indicates whether or not the user canceled the translation. `TRUE` if the user clicked the Cancel button in the progress dialog box, or typed Command-period while the box is displayed; otherwise, `FALSE`. When `TRUE`, you should stop the translation, and your `DoTranslateFile` or `DoTranslateScrap` callback function should return the result code `userCancelledErr`.

Return Value

A result code. See "Translation Manager Result Codes" (page 20).

Discussion

Your translation extension should call this function only in response to the `kTranslateTranslateFile` or `kTranslateTranslateScrap` request code (that is, you should only call this function in your `DoTranslateFile` or `DoTranslateScrap` callback function). Do not call this function in response to any other request code or from any code that isn't a translation extension.

You should already have called `SetTranslationAdvertisement` before calling `UpdateTranslationProgress`.

Special Considerations

This function might cause memory to be moved or purged; you should not call it at interrupt time.

Carbon Porting Notes

The functions contained in TranslationExtensions.h were originally written to be used only by someone implementing a Mac Easy Open translation component. Carbon, however, is for applications and not extensions. Therefore, this function is not supported.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`TranslationExtensions.h`

Document Revision History

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

Date	Notes
2006-07-12	Made minor formatting changes.
2006-07-24	Deprecated entire document.
2003-04-01	Added documentation for the functions `NewGetScrapDataUPP` (page 29), `InvokeGetScrapDataUPP` (page 29), and `DisposeGetScrapDataUPP` (page 24).
	Fixed formatting for callback abstracts.
2003-02-01	Updated to include Mac OS X availability information.

Index