Navigation Services Reference

Carbon > User Experience

2006-08-16

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

Simultaneously published in the United States and Canada.

Contents

Navigation Services Reference

Framework: Carbon/Carbon.h

Declared in Navigation.h

Overview

Navigation Services is an application programming interface that allows your application to provide a user interface for navigating, opening, and saving Mac OS file objects.

This reference describes the application programming interface for Navigation Services, as introduced with CarbonLib 1.1. Navigation Services establishes a new model for creating, displaying, and processing dialogs. This new functionality gives you the ability to create truly modeless dialogs and provides support for Unicode and Mac OS X sheets.

Navigation Services replaces the Standard File Package, which is not supported in Carbon.

Functions by Task

Creating Dialogs

`NavGetDefaultDialogCreationOptions` (page 33)
Determines the default attributes or behavior for dialogs.

Choosing Files, Folders and Volumes

`NavCreateChooseFileDialog` (page 17)
Creates a Choose File dialog, which prompts the user to select a single file as the target of an operation.

`NavCreateChooseFolderDialog` (page 18)
Creates a Choose Folder dialog, which prompts the user to select a folder as the target of an operation.

`NavCreateChooseVolumeDialog` (page 21)
Creates a Choose Volume dialog, which prompts the user to select a volume.

`NavCreateChooseObjectDialog` (page 19)
Creates a Choose Object dialog, which prompts the user to select a file. folder or volume.

`NavCreateGetFileDialog` (page 21)
Creates an Open dialog, which prompts the user to select a file or files to be opened.

`NavCreateNewFolderDialog` (page 23)
Creates a New Folder dialog.

Saving Files

`NavCreatePutFileDialog` (page 23)
Creates a Save dialog, which prompts the user for the name and location of a file to be saved.

`NavCreateAskSaveChangesDialog` (page 16)
Creates a dialog that asks the user whether to save changes.

`NavCreateAskReviewDocumentsDialog` (page 15)
Creates a Review Changes dialog, which notifies the user of multiple unsaved documents and gives the user the option to review them.

`NavCreateAskDiscardChangesDialog` (page 14)
Creates a dialog that asks the user whether to discard changes.

`NavDialogSetSaveFileName` (page 31)
Specifies the current value of the filename text field in a Save dialog.

`NavDialogSetSaveFileExtensionHidden` (page 31)
Sets the current state of extension hiding in a Save dialog.

`NavDialogGetSaveFileName` (page 27)
Obtains the current value of the filename text field in a Save dialog.

`NavDialogGetSaveFileExtensionHidden` (page 27)
Gets the current state of extension hiding in a Save dialog.

`NavCompleteSave` (page 13)
Completes a save operation and performs any needed translation on the file.

`NavCreatePreview` (page 85) Deprecated in Mac OS X v10.5
Creates a document preview in a specified file.

Customizing Dialogs

`NavCustomControl` (page 25)
Allows your application to control various settings in Navigation Services dialogs.

`NavDialogSetFilterTypeIdentifiers` (page 30)
Sets UTI filtering criteria for “get file” and “choose file” dialogs.

Running And Disposing of Dialogs

`NavDialogRun` (page 29)
Displays a previously created dialog.

`NavDialogDispose` (page 26)
Disposes of a dialog reference.

Obtaining Dialog Information

`NavDialogGetWindow` (page 29)
Obtains a window reference for a dialog.

`NavDialogGetUserAction` (page 28)
Reports the user action taken to dismiss a dialog.

`NavDialogGetReply` (page 26)
Reports the results of a dialog session (unless cancelled or programmatically terminated).

`NavDisposeReply` (page 32)
Releases the memory allocated for a `NavReplyRecord` structure after your application has finished using the structure.

Translating Files

`NavTranslateFile` (page 92) Deprecated in Mac OS X v10.5
Provides a means for files opened through Navigation Services to be read from different file formats.

Identifying Navigation Services Availability

`NavServicesAvailable` (page 34)
Reports whether the Navigation Services library is available on the user's system.

`NavLibraryVersion` (page 89) Deprecated in Mac OS X v10.5
Reports the currently installed version of the Navigation Services shared library.

Working With Universal Procedure Pointers

`NewNavEventUPP` (page 35)
Creates a new universal procedure pointer to your application-defined event–handling function.

`NewNavObjectFilterUPP` (page 36)
Creates a new universal procedure pointer to your application-defined filter function.

`NewNavPreviewUPP` (page 36)
Creates a new universal procedure pointer to your application-defined preview function.

`DisposeNavEventUPP` (page 10)
Disposes of a UPP to an application-defined event–handling function.

`DisposeNavObjectFilterUPP` (page 11)
Disposes of a UPP to an application-defined filter function.

`DisposeNavPreviewUPP` (page 11)
Disposes of a UPP to an application-defined preview function.

`InvokeNavEventUPP` (page 12)
Calls your application-defined event–handling function.

`InvokeNavObjectFilterUPP` (page 12)
Calls your application-defined filter function.

`InvokeNavPreviewUPP` (page 13)
Calls your application-defined preview function.

Deprecated Functions

`NavAskDiscardChanges` (page 77) Deprecated in Mac OS X v10.5
Displays an alert box that asks the user whether to discard changes to a particular document.

`NavAskSaveChanges` (page 78) Deprecated in Mac OS X v10.5
Displays a Save Changes alert box.

`NavChooseFile` (page 79) Deprecated in Mac OS X v10.5
Creates a simple dialog box that prompts the user to select a file.

`NavChooseFolder` (page 81) Deprecated in Mac OS X v10.5
Displays a dialog box that prompts the user to choose a folder or volume.

`NavChooseObject` (page 82) Deprecated in Mac OS X v10.5
Displays a dialog box that prompts the user to choose a file, folder, or volume.

`NavChooseVolume` (page 83) Deprecated in Mac OS X v10.5
Displays a dialog box that prompts the user to choose a volume.

`NavCustomAskSaveChanges` (page 86) Deprecated in Mac OS X v10.5
Displays a Save Changes alert box with a custom alert message.

`NavGetDefaultDialogOptions` (page 87) Deprecated in Mac OS X v10.5
Determines the default attributes or behavior for dialog boxes.

`NavGetFile` (page 87) Deprecated in Mac OS X v10.5
Displays a dialog box that prompts the user to select a file or files to be opened.

`NavNewFolder` (page 90) Deprecated in Mac OS X v10.5
Displays a dialog box that prompts the user to create a new folder.

`NavPutFile` (page 91) Deprecated in Mac OS X v10.5
Displays a Save dialog box.

Unsupported Functions

`NavLoad` (page 33)
Pre-loads the Navigation Services shared library.

`NavUnload` (page 35)
Unloads the Navigation Services shared library.

`NavServicesCanRun` (page 34)

Functions

DisposeNavEventUPP

Disposes of a UPP to an application-defined event–handling function.

```
void DisposeNavEventUPP (
   NavEventUPP userUPP
);
```

Parameters

userUPP

The UPP to dispose of.

Discussion

For more information on event–handling functions, see `NavEventProcPtr` (page 37).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Related Sample Code

QTMetaData

Declared In

`Navigation.h`

DisposeNavObjectFilterUPP

Disposes of a UPP to an application-defined filter function.

```
void DisposeNavObjectFilterUPP (
   NavObjectFilterUPP userUPP
);
```

Parameters

userUPP

The UPP to dispose of.

Discussion

For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

DisposeNavPreviewUPP

Disposes of a UPP to an application-defined preview function.

```
void DisposeNavPreviewUPP (
   NavPreviewUPP userUPP
);
```

Parameters

userUPP

The UPP to dispose of.

Discussion
For more information on preview functions, see `NavPreviewProcPtr` (page 39).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

InvokeNavEventUPP

Calls your application-defined event–handling function.

```
void InvokeNavEventUPP (
   NavEventCallbackMessage callBackSelector,
   NavCBRecPtr callBackParms,
   void *callBackUD,
   NavEventUPP userUPP
);
```

Discussion
You should not need to use the function `InvokeNavEventUPP`, as the system calls your event–handling function for you.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

InvokeNavObjectFilterUPP

Calls your application-defined filter function.

```
Boolean InvokeNavObjectFilterUPP (
   AEDesc *theItem,
   void *info,
   void *callBackUD,
   NavFilterModes filterMode,
   NavObjectFilterUPP userUPP
);
```

Discussion
You should not need to use the function `InvokeNavObjectFilterUPP`, as the system calls your filter function for you.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

InvokeNavPreviewUPP

Calls your application-defined preview function.

```
Boolean InvokeNavPreviewUPP (
   NavCBRecPtr callBackParms,
   void *callBackUD,
   NavPreviewUPP userUPP
);
```

Discussion
You should not need to use the function `InvokeNavPreviewUPP`, as the system calls your preview function for you.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

NavCompleteSave

Completes a save operation and performs any needed translation on the file.

```
OSErr NavCompleteSave (
   const NavReplyRecord *reply,
   NavTranslationOptions howToTranslate
);
```

Parameters

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavCompleteSave` call.

howToTranslate

A pointer to a structure of type `NavTranslationOptions`. Pass one of two values to specify how to perform any needed translation. For a description of the constants you can use to represent these values, see "Translation Options" (page 72). Translating in-place causes the source file to be replaced by the translation. Translating to a copy results in a file name followed by the string "(converted)" to avoid unwanted replacement. If you call the `NavCompleteSave` function in response to a Save a Copy command, you should pass the `kNavTranslateInPlace` constant in this parameter.

Return Value
A result code. See "Navigation Services Result Codes" (page 75). Since this function performs any needed translation, it may return a translation error.

Discussion
You should always call `NavCompleteSave` to complete any file saving operation performed with the `NavCreatePutFileDialog` function. `NavCompleteSave` performs any needed translation, so you do not have to use the function `NavTranslateFile` (page 92) when saving. If you wish to turn off automatic translation, set to `false` the value of the `translationNeeded` field of the `NavReplyRecord` structure you pass in the `reply` parameter of the `NavPutFile` function. If you turn off automatic translation, your application is responsible for any required translation.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTCarbonShell

Declared In
`Navigation.h`

NavCreateAskDiscardChangesDialog

Creates a dialog that asks the user whether to discard changes.

```
OSStatus NavCreateAskDiscardChangesDialog (
   const NavDialogCreationOptions *inOptions,
   NavEventUPP inEventProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog. You must supply a string in the `saveFileName` field of this structure; otherwise the function returns `paramErr`.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a Discard Changes dialog, A pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function

Return Value
A result code.

Discussion
This function creates a dialog that gives the user the option of discarding unsaved changes to a file or cancelling the operation. This dialog is most commonly used when the user wants to revert to the last saved version of a document.

Once you have successfully created the Discard Changes dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavAskDiscardChanges` function and adds support for Unicode and new window modalities.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

NavCreateAskReviewDocumentsDialog

Creates a Review Changes dialog, which notifies the user of multiple unsaved documents and gives the user the option to review them.

```
OSStatus NavCreateAskReviewDocumentsDialog (
   const NavDialogCreationOptions *inOptions,
   ItemCount inDocumentCount,
   NavEventUPP inEventProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions
A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inDocumentCount
The number of documents needing review. This number appears in the text presented to the user. If the total number of unsaved documents is unknown, specify 0; Navigation Services uses a general message. You should not specify 1; this alert should be used only when more than one document needs review. For more information, see *Inside Mac OS X: Aqua Human Interface Guidelines*.

inEventProc
A universal procedure pointer (UPP) to an application-defined event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inClientData
A pointer to an application-defined value that is passed back to all callback functions.

outDialog
Upon successful completion, a reference to the created dialog.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
The Review Changes dialog tells the user how many unsaved documents there are and asks the user to choose one of the following options:

- review the unsaved documents
- don't save any documents

- cancel

Use of this dialog is appropriate when an application is quitting and there is more than one unsaved document. It is supported only on Mac OS X; prior to Mac O X, this dialog is not part of the application quit sequence.

Upon successful creation, the dialog is not visible; to present and run the dialog, call the `NavDialogRun` (page 29) function. After the dialog is complete, dispose of it with the `NavDialogDispose` function. Upon dismissal of the dialog, the user's action is set to one of the following actions: `kNavUserActionReviewDocuments`, `kNavUserActionDiscardDocuments`, or `kNavUserActionCancel`. You can obtain this reply by calling the `NavDialogGetReply` (page 26).

Availability
Not available in CarbonLib.
Available in Mac OS X 10.1 and later.

Related Sample Code
QTCarbonShell

Declared In
`Navigation.h`

NavCreateAskSaveChangesDialog

Creates a dialog that asks the user whether to save changes.

```
OSStatus NavCreateAskSaveChangesDialog (
   const NavDialogCreationOptions *inOptions,
   NavAskSaveChangesAction inAction,
   NavEventUPP inEventProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions
A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inAction
A value indicating whether the user is closing a document or quitting the application and thereby determines the message displayed to the user. To provide a customized message for the dialog, specify a non-`NULL` value in the `message` field of the structure provided in the *inOptions* parameter.

inEventProc
A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inClientData
A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog
On successful creation of the Save Changes dialog, a pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value
A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion
This function creates a Save Changes dialog, which your application should display when the user attempts to close a document or quit the application with unsaved changes. The Save Changes dialog allows the user to choose one of the following options:

- save the changes
- discard the unsaved changes
- cancel the operation

Once you have successfully created the Save Changes dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, dispose of it by calling the `NavDialogDispose` (page 26) function.

If there is more than one document with unsaved changes when the user attempts to quit your application, you should display a Review Changes dialog instead. You can create a Review Changes dialog with the `NavCreateAskReviewDocumentsDialog` (page 15) function.

This function replaces the `NavAskSaveChanges` function and adds support for Unicode and new window modalities.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Related Sample Code
QTCarbonShell

Declared In
`Navigation.h`

NavCreateChooseFileDialog

Creates a Choose File dialog, which prompts the user to select a single file as the target of an operation.

```
OSStatus NavCreateChooseFileDialog (
   const NavDialogCreationOptions *inOptions,
   NavTypeListHandle inTypeList,
   NavEventUPP inEventProc,
   NavPreviewUPP inPreviewProc,
   NavObjectFilterUPP inFilterProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters
inOptions
A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inTypeList

A structure specifying a creator signature and a list of file types to show in the Choose File dialog.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inPreviewProc

A Universal Procedure Pointer (UPP) to your application's preview function. You may specify `NULL` if you don't need to register a preview function. For more information on preview functions, see `NavPreviewProcPtr` (page 39).

inFilterProc

A Universal Procedure Pointer (UPP) to your application's filter function. You may specify `NULL` if you don't need to register a filter function. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a Choose File dialog, A pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value
A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion
Once you have successfully created the Choose File dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavChooseFile` function and adds support for Unicode and new window modalities.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Related Sample Code
QTCarbonShell

Declared In
`Navigation.h`

NavCreateChooseFolderDialog

Creates a Choose Folder dialog, which prompts the user to select a folder as the target of an operation.

```
OSStatus NavCreateChooseFolderDialog (
   const NavDialogCreationOptions *inOptions,
   NavEventUPP inEventProc,
   NavObjectFilterUPP inFilterProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inFilterProc

A Universal Procedure Pointer (UPP) to your application's filter function. You may specify `NULL` if you don't need to register a filter function. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a Choose Folder dialog, A pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion

Once you have successfully created the Choose Folder dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavChooseFolder` function and adds support for Unicode and new window modalities.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavCreateChooseObjectDialog

Creates a Choose Object dialog, which prompts the user to select a file. folder or volume.

```
OSStatus NavCreateChooseObjectDialog (
   const NavDialogCreationOptions *inOptions,
   NavEventUPP inEventProc,
   NavPreviewUPP inPreviewProc,
   NavObjectFilterUPP inFilterProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inPreviewProc

A Universal Procedure Pointer (UPP) to your application's preview function. You may specify `NULL` if you don't need to register a preview function. For more information on preview functions, see `NavPreviewProcPtr` (page 39).

inFilterProc

A Universal Procedure Pointer (UPP) to your application's filter function. You may specify `NULL` if you don't need to register a filter function. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a Choose Object dialog, a pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion

Once you have successfully created the Choose Object dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavChooseObject` function and adds support for Unicode and new window modalities.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavCreateChooseVolumeDialog

Creates a Choose Volume dialog, which prompts the user to select a volume.

```
OSStatus NavCreateChooseVolumeDialog (
   const NavDialogCreationOptions *inOptions,
   NavEventUPP inEventProc,
   NavObjectFilterUPP inFilterProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inFilterProc

A Universal Procedure Pointer (UPP) to your application's filter function. You may specify `NULL` if you don't need to register a filter function. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a Choose Volume dialog, A pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion

Once you have successfully created the Choose Volume dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavChooseVolume` function and adds support for Unicode and new window modalities.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavCreateGetFileDialog

Creates an Open dialog, which prompts the user to select a file or files to be opened.

```
OSStatus NavCreateGetFileDialog (
   const NavDialogCreationOptions *inOptions,
   NavTypeListHandle inTypeList,
   NavEventUPP inEventProc,
   NavPreviewUPP inPreviewProc,
   NavObjectFilterUPP inFilterProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inTypeList

A structure specifying an application signature and a list of file types to show in the Open dialog.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inPreviewProc

A Universal Procedure Pointer (UPP) to your application's preview function. You may specify `NULL` if you don't need to register a preview function. For more information on creating a preview function, see `NavPreviewProcPtr` (page 39).

inFilterProc

A Universal Procedure Pointer (UPP) to your application's filter function. You may specify `NULL` if you don't need to register a filter function. For more information on creating a filter function, see `NavObjectFilterProcPtr` (page 38).

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of an Open dialog instance, this value specifies a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion

Once you have successfully created the Open dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavGetFile` function and adds support for Unicode and new window modalities.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavCreateNewFolderDialog

Creates a New Folder dialog.

```
OSStatus NavCreateNewFolderDialog (
   const NavDialogCreationOptions *inOptions,
   NavEventUPP inEventProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a New Folder dialog, a pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion

Once you have successfully created the New Folder dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function

Use the New Folder dialog to allow the user to create a new folder. Navigation Services creates the folder as specified by the user and returns a reference to the folder in the `selection` field of the reply record.

This function replaces the `NavNewFolder` function and adds support for Unicode and new window modalities.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavCreatePutFileDialog

Creates a Save dialog, which prompts the user for the name and location of a file to be saved.

```
OSStatus NavCreatePutFileDialog (
   const NavDialogCreationOptions *inOptions,
   OSType inFileType,
   OSType inFileCreator,
   NavEventUPP inEventProc,
   void *inClientData,
   NavDialogRef *outDialog
);
```

Parameters

inOptions

A pointer to a structure specifying options that control the appearance and behavior of the dialog.

inFileType

A four-character code specifying a file type for the file to be saved.

inFileCreator

A four-character code specifying a creator signature for the file to be saved. If you want to change or remove the top default item in the Format menu, pass `kNavGenericSignature`.

inEventProc

A Universal Procedure Pointer (UPP) to your application's event-handling function. You are strongly advised to create and register an event-handling function, as described in `NavEventProcPtr` (page 37). You must have an event-handling function in order to create modeless or window-modal (sheet) dialogs. Specify `NULL` in this parameter if you do not implement an event-handling function.

inClientData

A pointer to an application-defined value that is passed back to all callback functions. You can use this value to provide context information, for example. You may pass `NULL` in this parameter.

outDialog

On successful creation of a Save dialog, a pointer to a Navigation Services dialog reference that you can pass to the `NavDialogRun` (page 29) function.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). A result code.

Discussion

Once you have successfully created the Save dialog, you display it by calling the `NavDialogRun` (page 29) function. After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function. When you are finished with the dialog, you should dispose of it by calling the `NavDialogDispose` (page 26) function.

This function replaces the `NavPutFile` function and adds support for Unicode and new window modalities.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Related Sample Code

CarbonSketch

QTCarbonShell

Declared In

`Navigation.h`

NavCustomControl

Allows your application to control various settings in Navigation Services dialogs.

```
OSErr NavCustomControl (
   NavDialogRef dialog,
   NavCustomControlMessage selector,
   void *parms
);
```

Parameters

dialog

A Navigation Services dialog reference. You can obtain this value from the `context` field of the structure of type `NavCBRec` (page 40) specified in the `callBackParms` parameter of your event-handling function.

selector

A value of type `NavCustomControlMessage`. Pass one or more of the constants representing the possible values used to control various aspects of the active dialog. For a description of these constants, see "Custom Control Settings" (page 54).

parms

A pointer to a configuration value. Some of the control setting constants passed in the `selector` parameter require that you provide an additional configuration value. For a description of which constants require configuration values, see "Custom Control Settings" (page 54).

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

If you provide an event-handling function and an event occurs in a Navigation Services dialog, Navigation Services calls your event-handling function and specifies one of the constants described in "Event Messages" (page 64) in the `param` field of a `NavCBRec` (page 40) structure. Navigation Services specifies this structure in the `callBackParms` parameter of your event-handling function. When Navigation Services supplies the `kNavCBStart` constant in the `param` field, your application can call the `NavCustomControl` function and pass one of the constants described in "Custom Control Settings" (page 54) to control various aspects of the active Navigation Services dialog. For example, your application can tell Navigation Services to sort the browser list by date by calling the `NavCustomControl` function and passing the `kNavCtlSortBy` constant in the `selector` parameter and a pointer to the `kNavSortDateField` configuration constant in the `parms` parameter. (Some of the `NavCustomControlMessage` constants do not require a corresponding configuration constant.)

Note that your application can call the `NavCustomControl` function from within its event-handling function or its preview-drawing function.

Special Considerations

Navigation Services does not accept calls to the `NavCustomControl` function until an appropriate dialog box is fully initialized and displayed. Always check for the `kNavCBStart` constant, described in "Event Messages" (page 64), in the `param` field of the `NavCBRec` structure before calling the `NavCustomControl` function.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

NavDialogDispose

Disposes of a dialog reference.

```
void NavDialogDispose (
   NavDialogRef inDialog
);
```

Parameters

inDialog
A Navigation Services dialog reference previously obtained by your application.

Discussion
Use this function to dispose of a dialog reference when you are completely finished with its associated dialog. You may call `NavDialogDispose` from within your application-defined event-handling function.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTCarbonShell

Declared In
`Navigation.h`

NavDialogGetReply

Reports the results of a dialog session (unless cancelled or programmatically terminated).

```
OSStatus NavDialogGetReply (
   NavDialogRef inDialog,
   NavReplyRecord *outReply
);
```

Parameters

inDialog
A reference to a previously created dialog.

outReply
A pointer to a reply record you allocate to be filled out by Navigation Services.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
Call this function when you obtain a value other than `kNavUserActionCancel` or `kNavUserActionNone` from the `NavDialogGetUserAction` (page 28) function. Upon completion of the `NavDialogGetReply` (page 26) function, Navigation Services fills out the specified reply record with information about the dialog session. When you are finished with the reply record, remember to dispose of it by calling the `NavDisposeReply` function.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTCarbonShell

Declared In
`Navigation.h`

NavDialogGetSaveFileExtensionHidden

Gets the current state of extension hiding in a Save dialog.

```
Boolean NavDialogGetSaveFileExtensionHidden (
   NavDialogRef inPutFileDialog
);
```

Parameters
`inPutFileDialog`
A reference to the Save dialog. You can create a Save dialog using the `NavCreatePutFileDialog` (page 23) function.

Return Value
`True` if the extension is hidden; `false` if the extension is visible or if there is no extension.

Discussion
This function can be called at any time to determine if a Save dialog is hiding the file extension—if any—of the file to be saved.

Availability
Not available in CarbonLib.
Available in Mac OS X 10.1 and later.

Declared In
`Navigation.h`

NavDialogGetSaveFileName

Obtains the current value of the filename text field in a Save dialog.

```
CFStringRef NavDialogGetSaveFileName (
   NavDialogRef inPutFileDialog
);
```

Parameters

inPutFileDialog

A reference to a previously created dialog.

Return Value

A reference to the string containing the save filename. You should retain this string reference if you need the information after the dialog is dismissed. On Mac OS X, the full filename is returned, including any extension that may be hidden from the user. See the CFString documentation for a description of the `CFStringRef` data type.

Discussion

This function provides a Unicode-based replacement for using the `kNavGetEditFileName` selector with the `NavCustomControl` (page 25) function.

Special Considerations

Note that you cannot use `NavDialogGetSaveFileName` with a Save dialog created using the `NavPutFile` function. You should instead create your Save dialog using the `NavCreatePutFileDialog` function.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavDialogGetUserAction

Reports the user action taken to dismiss a dialog.

```
NavUserAction NavDialogGetUserAction (
   NavDialogRef inDialog
);
```

Parameters

inDialog

A reference to a previously created dialog.

Return Value

One of the constants defined by the `NavUserAction` enumeration. This value indicates the user action that dismissed the dialog. See "User Actions" (page 73) for a description of the values that may be returned here.

Discussion

If the dialog has not been dismissed or if the dialog was terminated by using the `kNavCtlTerminate` selector with the `NavCustomControl` (page 25) function, the `NavDialogGetUserAction` (page 28) function returns the `kNavUserActionNone` constant. When you obtain a value other than `kNavUserActionCancel` or `kNavUserActionNone` after returning from a file-handling dialog, Navigation Services fills out a reply record that you can obtain with the `NavDialogGetReply` (page 26) function.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch

Declared In
`Navigation.h`

NavDialogGetWindow

Obtains a window reference for a dialog.

```
WindowRef NavDialogGetWindow (
   NavDialogRef inDialog
);
```

Parameters

inDialog

A reference to a previously created dialog.

Return Value
A window reference for the specified dialog. Note that a valid dialog reference may not have a window associated with it until the `NavDialogRun` (page 29) function is called. If no window is associated with the specified dialog, the `NavDialogGetWindow` (page 29) function returns `NULL`.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

NavDialogRun

Displays a previously created dialog.

```
OSStatus NavDialogRun (
   NavDialogRef inDialog
);
```

Parameters

inDialog

A reference to a previously created Navigation Services dialog.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
You must create a dialog before displaying it. To create a dialog, call one of the `NavCreate...Dialog` functions described in "Choosing Files, Folders and Volumes" (page 7) and "Saving Files" (page 8). If you specify an application-modal or system-modal dialog, the `NavDialogRun` (page 29) function returns after the dialog is dismissed. If you specify a window-modal dialog (sheet) or a modeless dialog, the `NavDialogRun` (page 29) function returns immediately; in order to know when the dialog has been dismissed, you must supply an event-handling function and watch for the `kNavCBUserAction` event.

After the user interacts with the dialog, you can obtain information about the dialog session by calling the `NavDialogGetReply` (page 26) function.

Version Notes
On Mac OS 9 and earlier, all Navigation Services dialogs are modal, even if a window-modal or modeless dialog is requested. However, the `kNavCBUserAction` event is still sent to your event-handling function. It is possible to use a single programming model on both Mac OS 9 and on Mac OS X, provided you assume that the `NavDialogRun` function returns immediately after displaying the dialog.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTCarbonShell

Declared In
`Navigation.h`

NavDialogSetFilterTypeIdentifiers

Sets UTI filtering criteria for "get file" and "choose file" dialogs.

```
OSStatus NavDialogSetFilterTypeIdentifiers (
   NavDialogRef inGetFileDialog,
   CFArrayRef inTypeIdentifiers
);
```

Parameters

inGetFileDialog

A Navigation Services dialog reference obtained from calling `NavCreateChooseFileDialog` (page 17) or `NavCreateGetFileDialog` (page 21).

inTypeIdentifiers

A Core Foundation array of uniform type identifiers. This array specifies the file types that you want your dialog to enable. If you pass an empty array, all files are filtered (and will appear dimmed in the dialog). If you pass `NULL`, all files are enabled.

The file types you specify here also appear in the popup menu (displayed using the localized name associated with the UTI), allowing the user to filter by a specific file type. The "All readable documents" selection displays all the types specified in the UTI array.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
For simple filtering by file type, you should use this function instead of writing a custom filter callback function. However, you can also use this call in conjunction with a filter callback; your custom filter callback is called after `NavDialogSetFilterTypeIdentifiers` performs the initial filtering.

This function supersedes the list of `OSType` values you can pass in the `inTypeList` parameter in dialog creation functions.

You can call this function at any time, even while the dialog is displayed. For example, say your dialog contained a custom menu item to filter by a specific type. When the user selects your menu item, you could call `NavDialogSetFilterTypeIdentifiers` with the UTI corresponding to that type, and the dialog will automatically update with the new filtering criteria.

For more information about uniform type identifiers, see *Uniform Type Identifiers Overview*

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

Declared In
`Navigation.h`

NavDialogSetSaveFileExtensionHidden

Sets the current state of extension hiding in a Save dialog.

```
OSStatus NavDialogSetSaveFileExtensionHidden (
   NavDialogRef inPutFileDialog,
   Boolean inHidden
);
```

Parameters

inPutFileDialog

A reference to the Save dialog. You can create a Save dialog using the `NavCreatePutFileDialog` (page 23) function.

inHidden

A Boolean value indicating whether the file extension should be hidden. Pass `true` to hide the file extension; `false` to make any extension visible.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function can be called at any time to hide or show the extension of the file to be saved in a Save dialog. If the current filename has no extension, hiding the extension has no effect.

Availability
Not available in CarbonLib.
Available in Mac OS X 10.1 and later.

Declared In
`Navigation.h`

NavDialogSetSaveFileName

Specifies the current value of the filename text field in a Save dialog.

```
OSStatus NavDialogSetSaveFileName (
   NavDialogRef inPutFileDialog,
   CFStringRef inFileName
);
```

Parameters

inPutFileDialog

A reference to a previously created dialog.

inFileName

The filename to specify.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

This function may be called at any time to set the current filename for a save operation. You may use it to set an initial filename before calling `NavDialogRun`, or to change the filename dynamically while a dialog is running.

This function provides a Unicode-based replacement for using the `kNavSetEditFileName` selector with the `NavCustomControl` (page 25) function.

Special Considerations

Note that you cannot use `NavDialogSetSaveFileName` with a Save dialog created using the `NavPutFile` function. You should instead create your Save dialog using the `NavCreatePutFileDialog` function.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Navigation.h`

NavDisposeReply

Releases the memory allocated for a `NavReplyRecord` structure after your application has finished using the structure.

```
OSErr NavDisposeReply (
   NavReplyRecord *reply
);
```

Parameters

reply

A pointer to a structure of type `NavReplyRecord` (page 47) that your application has created.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

If your application calls a Navigation Services function that uses a structure of type `NavReplyRecord` (page 47), you must use the `NavDisposeReply` function afterward to release the memory allotted for the `NavReplyRecord` structure.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTCarbonShell
QTMetaData

Declared In
`Navigation.h`

NavGetDefaultDialogCreationOptions

Determines the default attributes or behavior for dialogs.

```
OSStatus NavGetDefaultDialogCreationOptions (
   NavDialogCreationOptions *outOptions
);
```

Parameters

outOptions

A pointer to a `NavDialogCreationOptions` (page 41) structure that you provide. On return, Navigation Services fills out the structure with default dialog configuration values.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function gives you a simple way to initialize a `NavDialogCreationOptions` (page 41) structure and set default options before creating a Navigation Services dialog. After you create the `NavDialogCreationOptions` structure, you can change the configuration options before you call one of the dialog creation functions.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTCarbonShell

Declared In
`Navigation.h`

NavLoad

Pre-loads the Navigation Services shared library.

Unsupported

```
OSErr NavLoad (
   void
);
```

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
Use this function to pre-load the Navigation Services library. Pre-loading increases the memory used by your application, but it provides the best performance when using Navigation Services functions. If you don't use the `NavLoad` function, the Navigation Services shared library may not be loaded until your application calls one of the Navigation Services functions. If you use the `NavLoad` function, you must call the function `NavUnload` (page 35) if you want to release reserved memory prior to quitting.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present. Not available in Mac OS X.

Declared In
`Navigation.h`

NavServicesAvailable

Reports whether the Navigation Services library is available on the user's system.

```
pascal Boolean NavServicesAvailable
```

Return Value
A Boolean value. This function returns `true` if Navigation Services is available, `false` if not.

Discussion
Use this function before attempting to use Navigation Services on Mac OS 8 and Mac OS 9. It is not necessary to call this function on Mac OS X, as Navigation Services is always available.

Special Considerations

There is a known problem with Navigation Services 1.0 that occurs if you call `NavServicesAvailable` more than once without the Appearance Manager being installed. Make sure that you check for the presence of the Appearance Manager before calling `NavServicesAvailable`.

Version Notes
Available in Navigation Services 1.0 and later.

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

Declared In
`Navigation.h`

NavServicesCanRun

Unsupported

```
Boolean NavServicesCanRun (
   void
);
```

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present. Not available in Mac OS X.

Declared In
`Navigation.h`

NavUnload

Unloads the Navigation Services shared library.

Unsupported

```
OSErr NavUnload (
   void
);
```

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function allows your application to unload the Navigation Services library and release the memory reserved for it. If you use the function `NavLoad` (page 33) to load the Navigation Services library, you must call the `NavUnload` function if you want to release reserved memory prior to quitting.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present. Not available in Mac OS X.

Declared In
`Navigation.h`

NewNavEventUPP

Creates a new universal procedure pointer to your application-defined event–handling function.

```
NavEventUPP NewNavEventUPP (
   NavEventProcPtr userRoutine
);
```

Parameters

`userRoutine`

A pointer to your event–handling function.

Return Value
On return, a universal procedure pointer (UPP) to the event–handling function. See the description of the `NavEventUPP` data type.

Availability
Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Related Sample Code
CarbonSketch
QTMetaData

Declared In
`Navigation.h`

NewNavObjectFilterUPP

Creates a new universal procedure pointer to your application-defined filter function.

```
NavObjectFilterUPP NewNavObjectFilterUPP (
   NavObjectFilterProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your filter function.

Return Value
On return, a universal procedure pointer (UPP) to the filter function. See the description of the `NavObjectFilterUPP` data type.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

NewNavPreviewUPP

Creates a new universal procedure pointer to your application-defined preview function.

```
NavPreviewUPP NewNavPreviewUPP (
   NavPreviewProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your preview function.

Return Value
On return, a universal procedure pointer (UPP) to the preview function. See the description of the `NavPreviewUPP` data type.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Navigation.h`

Callbacks

NavEventProcPtr

A pointer to an event-handling function that handles events such as window updating and resizing.

```
typedef void (*NavEventProcPtr) (
                NavEventCallbackMessage callBackSelector,
                NavCBRecPtr callBackParms,
                void * callBackUD);
```

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

```
void MyNavEventProc (
            NavEventCallbackMessage callBackSelector,
            NavCBRecPtr callBackParms,
            void * callBackUD);
```

Parameters

callBackSelector

One of the values specified by the `NavEventCallbackMessage` data type. These values indicate which type of event your function must respond to. For a description of the constants that represent these values, see "Event Messages" (page 64).

callBackParms

A pointer to a `NavCBRec` (page 40) structure. Your application uses the data supplied in this structure to process the event.

callBackUD

A pointer to a value set by your application when it calls a Navigation Services dialog creation function. When Navigation Services calls your event-handling function, the `callBackUD` value is passed back to your application in this parameter.

Discussion

Register your event-handling function by passing a Universal Procedure Pointer (UPP) in the `eventProc` parameter of a Navigation Services dialog creation function. You obtain this UPP by calling the function `NewNavEventUPP` and passing a pointer to your event-handling function. If you determine that an event is appropriate for your event-handling function, you can call other functions to handle custom control drawing.

When events involve controls, your event-handling function must respond to events only for your application-defined controls. To determine which control is affected by an event, pass the `kNavCtlGetFirstControlID` constant, described in "Custom Control Settings" (page 54), in the `selector` parameter of the function `NavCustomControl` (page 25).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Navigation.h`

NavObjectFilterProcPtr

A pointer to a filter function that determines whether file objects should be displayed in the browser list and navigation menus.

```
typedef Boolean (*NavObjectFilterProcPtr)
(
                AEDesc * theItem,
                void * info,
                void * callBackUD,
                NavFilterModes filterMode);
```

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

```
Boolean MyNavObjectFilterProc (
            AEDesc * theItem,
            void * info,
            void * callBackUD,
            NavFilterModes filterMode);
```

Parameters

theItem

A pointer to an Apple event descriptor structure (`AEDesc`). Navigation Services uses this structure to provide information about the object being passed to your filter function. Always check the Apple event descriptor type before deciding if an object needs to be filtered. Never assume that an object is a file specification, because the browser or pop-up menus may contain objects of other types. Make sure that your function only returns `true` if it recognizes the object.

info

A pointer to a `NavFileOrFolderInfo` (page 45) structure. Navigation Services uses this structure to provide file or folder information about the item being passed to your filter function. This information is only valid for objects of descriptor types `'typeFSS'` or `'typeFSRef'`.

callBackUD

A pointer to a value set by your application when it calls a Navigation Services dialog creation function. When Navigation Services calls your filter function, the `callBackUD` value is passed back to your application in this parameter.

filterMode

A value representing which list of objects is currently being filtered. For a description of the constants used to represent these values, see "Object Filtering Constants" (page 69).

Return Value

A Boolean value. If your application returns `true`, Navigation Services displays the object. If your application returns `false`, Navigation Services displays the object as dimmed.

Discussion

Register your filter function by passing a Universal Procedure Pointer (UPP) in the `filterProc` parameter of a dialog creation function. You obtain this UPP by calling the function `NewNavObjectFilterUPP` and passing a pointer to your filter function. Navigation Services calls your filter function to determine whether a file object should be displayed in the browser list or the pop-up menus.

If you use a filter function in conjunction with built-in translation, you should provide a list of file types to inform Navigation Services which document types your application can open. You can do so using the following methods:

- Call the `NavDialogSetFilterTypeIdentifiers` (page 30) function on an existing dialog to filter by uniform type identifiers. This method is preferable in Mac OS X v10.4 and later.
- Provide a list of allowable `OSType` file types in the `inTypeList` parameter of a file-opening function such as `NavCreateGetFileDialog`

If you provide a list of file types, your filter callback is called only for the files that match the specified type list. For example, if you wanted to enable only text files below a certain size, you could use `NavDialogSetFilterTypeIdentifiers` (page 30) to enable only text files, and then use a filter callback to screen for file size. You should make sure that your filter callback doesn't automatically eliminate a document type in the filter list (for example, if the list allows JPEG files and the callback eliminates everything but PICT files). This is to ensure that the user can always see some files when a particular file type is selected from the Enable popup menu.

If your filter function returns a result of `true`, Navigation Services displays the object. Note that this is the opposite of Standard File filter functions.

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

Declared In
`Navigation.h`

NavPreviewProcPtr

A pointer to a preview function that displays custom file previews.

```
typedef Boolean (*NavPreviewProcPtr)
(
                NavCBRecPtr callBackParms,
                void * callBackUD);
```

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

```
Boolean MyNavPreviewProc (
        NavCBRecPtr callBackParms,
        void * callBackUD);
```

Parameters

`callBackParms`

A pointer to a `NavCBRec` (page 40) structure. Navigation Services uses this structure to provide data needed for your function to draw the preview.

`callBackUD`

A pointer to a value set by your application when it calls a Navigation Services function such as `NavCreateGetFileDialog`. When Navigation Services calls your preview function, the `callBackUD` value is passed back to your application in this parameter.

Return Value
A Boolean value. Your application returns `true` if your preview function successfully draws the custom file preview. If your preview function returns `false`, Navigation Services displays the preview if the file contains a valid `'pnot'` resource. If your preview function returns `false` and a `'pnot'` resource is not available, Navigation Services displays a blank preview area.

Discussion
Register your preview function by passing the resulting Universal Procedure Pointer (UPP) in the `previewProc` parameter of a Navigation Services dialog creation function. You obtain this UPP by calling the function `NewNavPreviewUPP` and passing a pointer to your preview-drawing function. When the user selects a file, Navigation Services calls your preview-drawing function. Your preview function, in turn, calls the function `NavCustomControl` (page 25) to determine if the preview area is visible and, if so, what its dimensions are.

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

Declared In
`Navigation.h`

Data Types

NavDialogRef

An opaque reference to an instance of a Navigation Services dialog.

```
typedef struct __NavDialog * NavDialogRef;
```

Discussion
Your application obtains a `NavDialogRef` by calling one of the dialog creation functions described in "Choosing Files, Folders and Volumes" (page 7) and "Saving Files" (page 8). Once you obtain a valid reference, you pass it to other functions in order to display and process dialogs. When you are completely finished using the reference, dispose of it by calling the function `NavDialogDispose` (page 26). This data type is available in CarbonLib 1.1 and later and in Mac OS X. It replaces the `NavContext` data type previously used by Navigation Services.

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

Declared In
`Navigation.h`

NavCBRec

Provides information you can use for event-handling and customization.

```
struct NavCBRec {
    UInt16 version;
    NavDialogRef context;
    WindowRef window;
    Rect customRect;
    Rect previewRect;
    NavEventData eventData;
    NavUserAction userAction;
    char reserved[218];
};
typedef struct NavCBRec NavCBRec;
typedef NavCBRec * NavCBRecPtr;
```

Fields

`version`
: Identifies the version of this structure. This value is defined by the `kNavCBRecVersion` constant.

`context`
: An opaque object identifying the dialog instance.

`window`
: An opaque object identifying the dialog's window.

`customRect`
: A local coordinate rectangle describing the customization area available to your application. This determines how much room your application has to install custom controls.

`previewRect`
: A local coordinate rectangle describing the preview area available to your application's preview function. The minimum size is 145 pixels wide by 118 pixels high.

`eventData`
: A structure of type `NavEventData` (page 43). This structure provides event-specific data to your `NavEventProcPtr` (page 37) function.

`userAction`
: A constant specifying the action taken by the user, generating a `kNavCBUserAction` event. See "User Actions" (page 73) for a description of the possible values for this field.

  This field is available in CarbonLib 1.1 and later or in Mac OS X version 10.0 and later.

`reserved`
: Reserved.

Discussion

The `NavCBRec` structure is passed to your application-defined event-handling and preview functions. For more information on event-handling and preview functions, see `NavEventProcPtr` (page 37) and `NavPreviewProcPtr` (page 39), respectively.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Navigation.h`

NavDialogCreationOptions

Contains dialog configuration settings you can pass to Navigation Services dialog creation functions.

```
struct NavDialogCreationOptions {
    UInt16 version;
    NavDialogOptionFlags optionFlags;
    Point location;
    CFStringRef clientName;
    CFStringRef windowTitle;
    CFStringRef actionButtonLabel;
    CFStringRef cancelButtonLabel;
    CFStringRef saveFileName;
    CFStringRef message;
    UInt32 preferenceKey;
    CFArrayRef popupExtension;
    WindowModality modality;
    WindowRef parentWindow;
    char reserved[16];
};
typedef struct NavDialogCreationOptions NavDialogCreationOptions;
```

Fields

`version`

Identifies the version of this structure. The structure version is represented by the `kNavDialogCreationOptionsVersion` constant.

`optionFlags`

One of several constants defined by the `NavDialogOptionFlags` data type as described in "Dialog Configuration Options" (page 60).

`location`

A point describing the location of the upper-left corner of the dialog window, in global coordinates. If you set this field to (-1,-1), then the dialog window appears in the same location as when it was last closed. The size and location of the dialog window is persistent, but defaults to opening in the middle of the main screen if any portion is not visible when opened at the persistent location and size.

This field is ignored for sheet dialogs.

`clientName`

A string that identifies your application in the window title of file dialogs and in the message displayed for the Save Changes, Review Changes, and Ask Discard changes alerts.

On Mac OS 8 and 9, Navigation Services cannot maintain persistence information for your application if you do not provide this string.

`windowTitle`

A string that you can provide to override the default window title. If you pass `NULL`, the default window title is used.

`actionButtonLabel`

An alternative button title for the dialog's default button. If you pass `NULL`, the button uses the default label (Open or Save, for example.

`cancelButtonLabel`

An alternative button title for the dialog's Cancel button. If you pass `NULL`, the default button title is used.

`saveFileName`

The default filename for a file to be saved (Save dialog only). If you pass `NULL`, the filename field is blank.

`message`

For the file dialogs, a string for the banner, or prompt, below the browser list. This message can provide more descriptive instructions for the user. If you pass `NULL`, no banner appears and the browser list expands to fill that area.

For the Save Changes, Review Changes and Ask Discard Changes alerts, a string specifying a custom message that replaces the default message.

`preferenceKey`

An application-defined value that identifies which set of dialog preferences Navigation Services should use. If your application maintains multiple sets of preferences for a particular type of dialog, you can determine which set is active by specifying the appropriate value in the `preferenceKey` field. For example, an application may provide one set of preferences when it calls the function to open text files and a different set of preferences when opening movie files. If you do not wish to provide a preference key, specify 0 for the `preferenceKey` value.

`popupExtension`

A reference to an array of menu item strings. These strings are used to add extra menu items to the Show pop-up menu in an Open dialog or to the Format pop-up menu in a Save dialog. Your application can use this array to add additional document types to be opened or saved, or different ways of saving a file (with or without line breaks, for example).

`modality`

This value allows you to specify the modality of the dialog. The default modality for all dialogs is `kWindowModalityAppModal`. If you specify the `kWindowModalityWindowModal` constant to make a dialog appear as a sheet, you must provide a valid window reference in the `parentWindow` field. If you specify the `kWindowModalityWindowModal` constant on Mac OS 8 or 9, the modality is set to `kWindowModalityAppModal`.

This field is available in CarbonLib 1.1 and later or in Mac OS X version 10.0 and later.

`parentWindow`

A reference to the parent window for a sheet.

This field is available in CarbonLib 1.1 and later or in Mac OS X version 10.0 and later.

`reserved`

Reserved.

Discussion

When you create a Navigation Services dialog, using one of the `NavCreate...Dialog` creation functions, you must supply a `NavDialogCreationOptions` structure to specify the appearance and behavior of the dialog. You can initialize a `NavDialogCreationOptions` structure using the `NavGetDefaultDialogCreationOptions` (page 33) function; this fills out the structure with the default dialog creation settings.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Navigation.h`

NavEventData

Contains event data for Navigation Services dialogs.

```
struct NavEventData {
    NavEventDataInfo eventDataParms;
    SInt16 itemHit;
};
typedef struct NavEventData NavEventData;
```

Fields

`eventDataParms`

A structure of type `NavEventDataInfo` (page 44).

`itemHit`

A signed integer value. On return, this value represents the item number of the dialog item last clicked by the user. If the user clicks something other than a valid Navigation Services-generated control item, this value is -1.

Discussion

The `NavEventData` structure is passed to your application-defined event-handling or preview function in the `eventData` field of the `NavCBRec` structure.

The `NavEventData` structure contains a structure of type `NavEventDataInfo` (page 44). In Navigation Services 1.1 or later, the `NavEventData` structure also contains a field describing the dialog item last clicked by the user.

Version Notes

`itemHit` field added in Navigation Services 1.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Navigation.h`

NavEventDataInfo

Provides event–handling data to your application.

```
union NavEventDataInfo {
    EventRecord * event;
    void * param;
};
typedef union NavEventDataInfo NavEventDataInfo;
```

Fields

`event`

A pointer to the `EventRecord` structure describing an event to be handled by your event-handling function.

`param`

A pointer to additional event data. In most cases, this data consists of an Apple event descriptor list (`AEDescList`) for the file or files affected by the event described in the `event` field. For example, if the event consists of the user making a selection in the browser list, the `AEDescList` specifies the file or files selected.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`Navigation.h`

NavFileOrFolderInfo

Contains file or folder information for use by your application-defined filter function.

```
struct NavFileOrFolderInfo {
    UInt16 version
    Boolean isFolder
    Boolean visible
    UInt32 creationDate
    UInt32 modificationDate
    union {
        struct {
            Boolean locked;
            Boolean resourceOpen;
            Boolean dataOpen;
            Boolean reserved1;
            UInt32 dataSize;
            UInt32 resourceSize;
            FInfo finderInfo;
            FXInfo finderXInfo;
        } fileInfo;
        struct {
            Boolean shareable;
            Boolean sharePoint;
            Boolean mounted;
            Boolean readable;
            Boolean writeable;
            Boolean reserved2;
            UInt32 numberOfFiles;
            DInfo finderDInfo;
            DXInfo finderDXInfo;
            OSType folderType;
            OSType folderCreator;
            char reserved3[206];
        } folderInfo;
    } fileAndFolder;
};
typedef struct NavFileOrFolderInfo NavFileOrFolderInfo;
```

Fields

`version`
: Identifies the version of this structure.

`isFolder`
: A Boolean value. If this value is set to `true`, the object being described is a folder or volume; otherwise, the value is set to `false`. An alias to a folder or volume returns `true`. Check for the `kIsAlias` constant in the `fileAndFolder.folderInfo.finderDInfo` field to determine whether an object is an alias.

`visible`
: A Boolean value. If this value is set to `true`, the object being described is visible in the browser list; otherwise, the value is set to `false`.

`creationDate`
: The creation date of the object being described.

`modificationDate`
The modification date of the object being described.

`fileAndFolder.fileInfo.locked`
If `isFolder` is `false`, a Boolean value indicating whether the file is locked.

`fileAndFolder.fileInfo.resourceOpen`
If `isFolder` is `false`, a Boolean value indicating whether the resource fork of the file is open.

`fileAndFolder.fileInfo.dataOpen`
If `isFolder` is `false`, a Boolean value indicating whether the data fork of the file is open.

`fileAndFolder.fileInfo.reserved1`
Reserved.

`fileAndFolder.fileInfo.dataSize`
If `isFolder` is `false`, the size of the file's data fork.

`fileAndFolder.fileInfo.resourceSize`
If `isFolder` is `false`, the size of the file's resource fork.

`fileAndFolder.fileInfo.finderInfo`
If `isFolder` is `false`, a structure specifying further information about the file. See the Finder Interface documentation for more information on the `FInfo` structure.

`fileAndFolder.fileInfo.finderXInfo`
If `isFolder` is `false`, a structure specifying extended Finder information for the file. See the Finder Interface documentation for more information on the `FXInfo` structure.

`fileAndFolder.folderInfo.shareable`
If `isFolder` is `true`, a Boolean value indicating whether the folder is shareable.

`fileAndFolder.folderInfo.sharePoint`
If `isFolder` is `true`, a Boolean value indicating whether the folder is a share point.

`fileAndFolder.folderInfo.mounted`
If `isFolder` is `true`, a Boolean value indicating whether the folder is mounted.

`fileAndFolder.folderInfo.readable`
If `isFolder` is `true`, a Boolean value indicating whether the folder is readable.

`fileAndFolder.folderInfo.writeable`
If `isFolder` is `true`, a Boolean value indicating whether the folder is writeable.

`fileAndFolder.folderInfo.reserved2`
Reserved.

`fileAndFolder.folderInfo.numberOfFiles`
If `isFolder` is `true`, the number of files in the folder.

`fileAndFolder.folderInfo.finderDInfo`
If `isFolder` is `true`, a directory information structure describing the folder. See the Finder Interface documentation for further information on the `DInfo` structure.

`fileAndFolder.folderInfo.finderDXInfo`
If `isFolder` is `true`, an extended directory information structure describing the folder. See the Finder Interface documentation for further information on the `DXInfo` structure.

`fileAndFolder.folderInfo.folderType`
If `isFolder` is `true`, the package type (for structure version 1 or greater).

`fileAndFolder.folderInfo.folderCreator`
If `isFolder` is `true`, the creator code for the package (for structure version 1 or greater).

Discussion
The `NavFileOrFolderInfo` structure contains file or folder information for use by your application-defined filter function. Your filter function can determine whether the currently selected object is a file by checking the `isFolder` field of the `NavFileOrFolderInfo` structure for the value `false`. After making this determination, you can obtain more information about the object from the structure specified in the `fileAndFolder` field.

Special Considerations

The information in this structure is valid only for HFS file objects.

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

Declared In
`Navigation.h`

NavReplyRecord

Contains information about user interaction with a dialog.

```
struct NavReplyRecord {
    UInt16 version;
    Boolean validRecord;
    Boolean replacing;
    Boolean isStationery;
    Boolean translationNeeded;
    AEDescList selection;
    ScriptCode keyScript;
    FileTranslationSpecArrayHandle fileTranslation;
    UInt32 reserved1;
    CFStringRef saveFileName;
    Boolean saveFileExtensionHidden;
    UInt8 reserved2;
    char reserved[225];
};
typedef struct NavReplyRecord NavReplyRecord;
```

Fields

`version`
: Identifies the version of this structure. The structure version is represented by the constant `kNavReplyRecordVersion`.

`validRecord`
: A Boolean value of `true` if the user closes a dialog by pressing Return or Enter, or by clicking the default button in an Open or Save dialog. If this field is `false`, all other fields are unused and do not contain valid data.

`replacing`
: A Boolean value of `true` if the user chooses to save a file by replacing an existing file (thereby necessitating the removal or renaming of the existing file).

`isStationery`
: A Boolean value informing your application whether the file about to be saved should be saved as a stationery document.

`translationNeeded`
: A Boolean value indicating whether translation was or will be needed for files selected in Open and Save dialogs.

`selection`
: For a file-opening or file-choosing dialog, this is an Apple event descriptor list (`AEDescList`) containing references to items selected by the user. Navigation Services creates this list, which is automatically disposed of when your application calls the `NavDisposeReply` function. Some dialogs may return one or more items; you can determine the number of items in the list by calling the Apple Event Manager function `AECountItems`. Each selected HFS file object is described in an `AEDesc` structure of type `'typeFSS'` or `'typeFSRef'`. You can coerce this descriptor into a file reference to perform operations such as opening the file. If you use one of the Carbon-compliant dialog creation functions described in"Choosing Files, Folders and Volumes" (page 7) and "Saving Files" (page 8), the descriptor is of type `'typeFSS'` on Mac OS 8 or 9; on Mac OS X systems, this descriptor is of type `'typeFSRef'`.File-saving dialogs always return a single descriptor in the list. If you use the `NavCreatePutFileDialog` (page 23) function, this descriptor specifies the directory where the file is to be saved. You can obtain the name for the save file from the `saveFileName` field.

`keyScript`
: The keyboard script system used for the filename.

`fileTranslation`
: A handle to a Translation Manager structure of type `FileTranslationSpec`. This structure contains a corresponding translation array for each file reference returned in the `selection` field. When opening files, Navigation Services performs the translation automatically unless you set the `kNavDontAutoTranslate` flag in the `dialogOptionFlags` field of the `NavDialogCreationOptions` (page 41) structure. When Navigation Services performs an automatic translation, the `FileTranslationSpec` structure is strictly for the Translation Manager's use. If you turn off automatic translation, your application may use the `FileTranslationSpec` structure for your own translation scheme. If the user chooses a translation for a saved file, the `FileTranslationSpec` structure contains a single translation reference for the saved file and the `translationNeeded` field of the `NavReplyRecord` structure is set to `true`. The handle to the `FileTranslationSpec` structure is locked, so you can safely use dereferenced pointers.

`reserved1`
: Reserved.

`saveFileName`
: If the reply record is filled out by a dialog created with the `NavCreatePutFileDialog` (page 23) function, this field contains a string specifying the name of a file to be saved. This field contains the entire name of the file, regardless of whether or not any file extension is visible to the user. You can identify the directory in which the file is to be saved by checking the `selection` field.

  This field was added in structure version 1.

`saveFileExtensionHidden`
: A Boolean value indicating whether the extension on the name of the saved file should be hidden. Once the file has been saved, the client call the `NavCompleteSave` function. `NavCompleteSave` hides the extension on the file. However, the client needs to know that the extension is hidden so that it can display the document name correctly in the user interface, such as in window titles and menus. This field is only used if the client has requested extension preservation using the `kNavPreserveSaveFileExtension` dialog option flag. This field was added in structure version 2.

`reserved2`
: Reserved.

`reserved`
: Reserved.

Discussion
Navigation Services uses the `NavReplyRecord` structure to provide your application with information about the user's interactions with a Navigation Services dialog. If the dialog is created with the Carbon-compliant `NavCreate...Dialog` functions, you obtain the `NavReplyRecord` by calling the `NavDialogGetReply` (page 26) function after your event-handling function receives the `kNavCBUserAction` event. If you create the dialog using one of the older functions, you pass the address of a `NavReplyRecord` directly to the function that invokes the dialog. When your application is through using the structure, remember to dispose of it by calling the function `NavDisposeReply` (page 32).

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

Declared In
`Navigation.h`

NavTypeList

Defines a list of file types that your application is capable of opening.

```
struct NavTypeList {
    OSType componentSignature;
    short reserved;
    short osTypeCount;
    OSType osType[1];
};
typedef struct NavTypeList NavTypeList;
typedef NavTypeList * NavTypeListPtr;
```

Fields

`componentSignature`
: A four character code specifying your application signature. If you want your application to be able to open all files of the types you specify in the `osType` field (regardless of which application created them), specify the `kNavGenericSignature` constant in this field.

`reserved`
: Reserved.

`osTypeCount`
: A number indicating how many file types are defined in the `osType` field.

`osType`
: A list of file types your application can open.

Discussion
Your application uses the `NavTypeList` structure to define a list of file types that your application is capable of opening. Your application passes a pointer to this list to Navigation Services functions that display Open or Save dialogs. You may create this list dynamically or reference a Translation Manager `'open'` resource.

For more information on the `'open'` resource and the Translation Manager, see the "Translation Manager" chapter in *Inside Macintosh: More Macintosh Toolbox*.

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

Declared In
`Navigation.h`

NavEventUPP

Defines a universal procedure pointer (UPP) to an application-defined event–handling function.

```
typedef NavEventProcPtr NavEventUPP;
```

Discussion
For more information, see the description of the `NavEventProcPtr` (page 37) callback function.

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

Declared In
`Navigation.h`

NavObjectFilterUPP

Defines a universal procedure pointer (UPP) to an application-defined filter function.

```
typedef NavObjectFilterProcPtr NavObjectFilterUPP;
```

Discussion
For more information, see the description of the `NavObjectFilterProcPtr` (page 38) callback function.

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

Declared In
`Navigation.h`

NavPreviewUPP

Defines a universal procedure pointer (UPP) to an application-defined preview function.

```
typedef NavPreviewProcPtr NavPreviewUPP;
```

Discussion
For more information, see the description of the `NavPreviewProcPtr` (page 39) callback function.

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

Declared In
`Navigation.h`

NavMenuItemSpec

Defines additional items in an Open dialog's Show pop-up menu or a Save dialog's Format pop-up menu.

```
struct NavMenuItemSpec {
    UInt16 version;
    OSType menuCreator;
    OSType menuType;
    Str255 menuItemName;
    char reserved[245];
};
typedef struct NavMenuItemSpec          NavMenuItemSpec;
typedef NavMenuItemSpec *               NavMenuItemSpecArrayPtr;
typedef NavMenuItemSpecArrayPtr *       NavMenuItemSpecArrayHandle;
typedef NavMenuItemSpecArrayPtr         NavMenuItemSpecPtr;
typedef NavMenuItemSpecArrayHandle      NavMenuItemSpecHandle;
```

Fields

`version`

Identifies the version of this structure. Be sure to specify the `kNavMenuItemSpecVersion` constant in this field.

`menuCreator`

A unique value set by your application. Navigation Services passes this value back to your application to identify the application type of the selected menu item.

`menuType`

A unique value set by your application. Navigation Services passes this value back to your application to identify the type of the selected menu item. Values from -1 to 10 are reserved for Navigation Services.

`menuItemName`

The item name that appears in the pop-up menu.

`reserved`

Reserved for future use.

Discussion

For information about file creators and file types, see *Inside Macintosh: Macintosh Toolbox Essentials*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Navigation.h`

NavContext

An old name for `NavDialogRef`.

Not recommended

```
typedef NavDialogRef NavContext;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Navigation.h`

NavDialogOptions

Contains dialog box configuration settings.

Not recommended

```
struct NavDialogOptions {
    UInt16 version;
    NavDialogOptionFlags dialogOptionFlags;
    Point location;
    Str255 clientName;
    Str255 windowTitle;
    Str255 actionButtonLabel;
    Str255 cancelButtonLabel;
    Str255 savedFileName;
    Str255 message;
    UInt32 preferenceKey;
    NavMenuItemSpecArrayHandle popupExtension;
    char reserved[494];
};
typedef struct NavDialogOptions NavDialogOptions;
```

Fields

`version`

Identifies the version of this structure. Be sure to specify the `kNavDialogOptionsVersion` constant in this field.

`dialogOptionFlags`

One of several constants defined by the `NavDialogOptionFlags` data type as described in "Dialog Configuration Options" (page 60).

`location`

The upper-left location of the dialog box (in global coordinates). If you set the `dialogOptionFlags` field to `NULL` or set this field to (-1,-1), then the dialog box appears in the same location as when last closed. The size and location of the dialog box is persistent, but defaults to opening in the middle of the main screen if any portion is not visible when opened at the persistent location and size.

`clientName`

A string that identifies your application in the dialog box window title.

`windowTitle`

A string that you can provide to override the default window title.

`actionButtonLabel`

An alternative button title for the dialog box's action button. If you do not specify a title, the button will use the default label (Open or Save, for example.

`cancelButtonLabel`

An alternative button title for the Cancel button in dialog boxes.

`savedFileName`

The default filename for a saved file.

`message`

The string for the banner, or prompt, below the browser list. This message can provide more descriptive instructions for the user. If you don't provide a message string, the browser list expands to fill that area.

`preferenceKey`

An application-defined value that identifies which set of dialog box preferences Navigation Services should use. If your application maintains multiple sets of preferences for a particular type of dialog box, you can determine which set is active by specifying the appropriate value in the `preferenceKey` field. For example, an application may allow one set of preferences when it calls the function `NavGetFile` (page 87) to open text files and a different set of preferences when opening movie files. If you do not wish to provide a preference key, specify `NULL` for the `preferenceKey` value.

`popupExtension`

A handle to one or more structures of type `NavMenuItemSpec` (page 50) used to add extra menu items to the Show pop-up menu in an Open dialog box or the Format pop-up menu in Save dialog boxes. Using `NavMenuItemSpec` structures allows your application to add additional document types to be opened or saved, or different ways of saving a file (with or without line breaks, for example).

`reserved`

Reserved for future use.

Carbon Porting Notes

The `NavDialogCreationOptions` (page 41) structure is the recommended replacement for the `NavDialogOptions` structure. *`NavDialogCreationOptions`* uses CFString objects instead of Pascal strings, thereby adding support for Unicode. In addition, `NavDialogCreationOptions` adds fields for controlling window modality and setting the parent window (necessary for sheets on OS X).

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

Declared In
`Navigation.h`

Constants

Action State Constants

Let you block certain actions in dialogs.

```
typedef UInt32 NavActionState;
enum {
    kNavNormalState = 0x00000000,
    kNavDontOpenState = 0x00000001,
    kNavDontSaveState = 0x00000002,
    kNavDontChooseState = 0x00000004,
    kNavDontNewFolderState = 0x00000010
};
```

Constants

`kNavNormalState`

Allows all user actions. This is the default state.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontOpenState`
Prevents Navigation Services from opening files.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontSaveState`
Prevents Navigation Services from saving files.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontChooseState`
Prevents Navigation Services from choosing files.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontNewFolderState`
Prevents Navigation Services from creating new folders.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
The `NavActionState` enumeration defines constants you can specify in the `parms` parameter of the function `NavCustomControl` (page 25) in order to block certain actions in Navigation Services dialogs. When you specify these constants, you must also specify the `kNavSetActionState` constant in the `selector` parameter of the `NavCustomControl` function.

Version Notes
These constants are only available in Navigation Services 2.0 or later.

Custom Control Settings

Provide constants that allow you to control various aspects of the active dialog.

```
typedef SInt32 NavCustomControlMessage;
enum {
    kNavCtlShowDesktop = 0,
    kNavCtlSortBy = 1,
    kNavCtlSortOrder = 2,
    kNavCtlScrollHome = 3,
    kNavCtlScrollEnd = 4,
    kNavCtlPageUp = 5,
    kNavCtlPageDown = 6,
    kNavCtlGetLocation = 7,
    kNavCtlSetLocation = 8,
    kNavCtlGetSelection = 9,
    kNavCtlSetSelection = 10,
    kNavCtlShowSelection = 11,
    kNavCtlOpenSelection = 12,
    kNavCtlEjectVolume = 13,
    kNavCtlNewFolder = 14,
    kNavCtlCancel = 15,
    kNavCtlAccept = 16,
    kNavCtlIsPreviewShowing = 17,
    kNavCtlAddControl = 18,
    kNavCtlAddControlList = 19,
    kNavCtlGetFirstControlID = 20,
    kNavCtlSelectCustomType = 21,
    kNavCtlSelectAllType = 22,
    kNavCtlGetEditFileName = 23,
    kNavCtlSetEditFileName = 24,
    kNavCtlSelectEditFileName = 25,
    kNavCtlBrowserSelectAll = 26,
    kNavCtlGotoParent = 27,
    kNavCtlSetActionState = 28,
    kNavCtlBrowserRedraw = 29,
    kNavCtlTerminate = 30
};
```

Constants

`kNavCtlShowDesktop`

Tells Navigation Services to change the browser list location to the desktop.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSortBy`

Alerts Navigation Services that your application is setting a sort key in the browser list. In addition to the `kNavCtlSortBy` constant, your application passes one of the `NavSortKeyField` constants in the `parms` parameter of the function `NavCustomControl` (page 25). For a description of the `NavSortKeyField` constants, see "File Sorting Constants" (page 67).

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSortOrder`

Alerts Navigation Services that your application is setting sort order, either ascending or descending, in the browser list. In addition to passing the `kNavCtlSortOrder` constant, your application must pass one of the `NavSortOrder` constants in the `parms` parameter of the `NavCustomControl` function. For a description of the `NavSortOrder` constants, see "Sort Order Constants" (page 71).

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlScrollHome`
Tells Navigation Services to scroll the browser to the top of the file list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlScrollEnd`
Tells Navigation Services to scroll the browser to the bottom of the file list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlPageUp`
Tells Navigation Services to scroll the browser up one page length as a result of the user clicking the scroll bar above the scroll box.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlPageDown`
Tells Navigation Services to scroll the browser down one page length as a result of the user clicking the scroll bar below the scroll box.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlGetLocation`
Tells Navigation Services to return the current location. Navigation Services reports the current location by setting a pointer to an `AEDesc` structure in the `param` field of the structure of type `NavCBRec` (page 40) that you specified in your event-handling function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSetLocation`
Tells Navigation Services that your application wishes to set the location being viewed in the browser list. In addition to specifying the `kNavCtlSetLocation` constant, your application passes a pointer to an `AEDesc` structure describing the new location in the `parms` parameter of the `NavCustomControl` function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlGetSelection`
Tells Navigation Services to return the selected item or items in the browser. When you specify this constant, Navigation Services returns a pointer to an `AEDesc` structure describing the selected item(s) in the `param` field of the structure of type `NavCBRec` (page 40) that you specified in your event-handling function. If the user deselects the current selection, the `AEDescList` returned by Navigation Services contains an empty reference. You can account for this case by using the function `AECountItems` and checking for a zero count.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSetSelection`

Tells Navigation Services to change the browser list selection. In addition to specifying the `kNavCtlSetSelection` constant, your application must pass a pointer to an `AEDescList` structure describing the selection in the `parms` parameter of the `NavCustomControl` function. If you want to deselect the current selection without making a new selection, pass `NULL` for the pointer. Note: If you specify this constant, Navigation Services notifies your event-handling function by setting the `kNavCBSelectEntry` constant twice; once when the previous selection is deselected, and once when the new selection is made.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlShowSelection`

Tells Navigation Services to make the current selection visible in the browser list if the selection has been scrolled out of sight by the user.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlOpenSelection`

Tells Navigation Services to open the current selection.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlEjectVolume`

Tells Navigation Services to eject a volume. In addition to specifying this constant, you pass a pointer to the volume reference number (`vRefNum`) of the volume to be ejected in the `parms` parameter of the `NavCustomControl` function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlNewFolder`

Tells Navigation Services to create a new folder in the current browser location. In addition to specifying the `kNavCtlNewFolder` constant, your application passes a string representing the name of the new folder in the `parms` parameter of the `NavCustomControl` function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlCancel`

Tells Navigation Services to dismiss the Open or Save dialog as if the user had pressed the Cancel button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlAccept`

Tells Navigation Services to close the Open or Save dialog as if the user had pressed the Open or Save button. Navigation Services does not act on this constant if there is no current selection.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlIsPreviewShowing`

Asks Navigation Services if the preview area is currently available. If you specify this constant, Navigation Services sets a pointer to a Boolean value in the `param` field of the `NavCBRec` (page 40) structure that you specified in your event-handling function. This value is `true` if the preview area is available, `false` otherwise.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlAddControl`

Tells Navigation Services to add one application-defined control to Open or Save dialogs. In addition to sending this message, your application passes a control handle in the `parms` parameter of the `NavCustomControl` function. Design the control in local coordinates.

Note: To avoid any unnecessary flickering or redrawing, ensure the control is initially invisible before specifying this constant. You may set the control to visible after Navigation Services supplies the `kNavCBStart` constant, described in "Event Messages" (page 64), in the `param` field of the `NavCBRec` (page 40) structure. If the user resizes the dialog, your application must move the control because it is not maintained by Navigation Services. If you use the `kNavCtlAddControlList` constant (described next) and you supply a `'DITL'` resource, you avoid the need to move the control yourself.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlAddControlList`

Tells Navigation Services to add a list of application-defined dialog items to Open or Save dialogs. In addition to specifying this constant, your application passes a handle to a dialog item list or `'DITL'` resource in the `parms` parameter of the `NavCustomControl` function. Design the `'DITL'` resource in local coordinates. Navigation Services adds the custom items relative to the upper left corner of the customization area. If the user resizes the dialog, your custom items are moved automatically.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlGetFirstControlID`

Asks Navigation Services to help you identify the first custom control in the dialog, in order to determine which custom control item was selected by the user. Navigation Services returns a pointer to a 16-bit integer that indicates the item number of the first custom control in the param field of the structure of type `NavCBRec` (page 40) that you specified in your event-handling function. In your event-handling function, use the Dialog Manager function `FindDialogItem` to find out which item was selected. The `FindDialogItem` function returns 0 for the first item, 1 for the second and so on. To get the proper item number, add 1 to the `FindDialogItem` function result. The Open or Save dialog's standard controls precede yours, so use the formula `(itemHit - yourFirstItem + 1)` to determine which of your items was selected. Your application should not depend on any hardcoded value for the number of items, since this value is likely to change in the future.

Be sure to test the result from `FindDialogItem` to ensure that it describes a control that you defined. Your application must not respond to any controls that do not belong to it.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSelectCustomType`

Tells Navigation Services to set one of your custom menu items in the Show pop-up menu or the Format pop-up menu as the default selection. This is useful if you want to override the default pop-up menu selection. In addition to specifying this constant, pass a pointer to a `NavMenuItem` structure in the `parms` parameter of the `NavCustomControl` function. This structure describes the item you wish to have selected.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSelectAllType`

Tells Navigation Services to override the default menu item in the Type pop-up menu. By specifying one of the `NavPopupMenuItem` constants, described in"Menu Item Selection Constants" (page 68), in the `parms` parameter of the `NavCustomControl` function, you can set the default item to All [AppName] Documents, All Readable Documents or All Documents.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlGetEditFileName`

Tells Navigation Services to return the name of the file to be saved by a file-saving function. This would be useful if you wanted to automatically add an extension to the filename, for example. When you send this message, the `parms` parameter of the `NavCustomControl` function returns a `StringPtr` to a Pascal string containing the filename. Note that in Carbon, you can use the `NavDialogGetSaveFileName` (page 27) function to obtain a Unicode string containing the filename.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSetEditFileName`

Tells Navigation Services that your application wishes to set the name of the file to be saved by a file-saving function. Your application normally specifies the `KNavCtlSetEditFileName` constant after modifying the filename obtained by specifying the `kNavCtlGetEditFileName` constant. In addition to specifying the `kNavCtlSetEditFileName` constant, your application passes a `StringPtr` to a Pascal string containing the filename in the `parms` parameter of the `NavCustomControl` function. Note that you can set the filename with a Unicode string by calling the `NavDialogSetSaveFileName` (page 31) function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSelectEditFileName`

(Navigation Services 1.1 or later) Tells Navigation Services to display the name of the file to be saved by the function with some or all of the filename string highlighted for selection. In addition to specifying the `kNavCtlSelectEditFileName` constant, your application passes a Control Manager structure of type `ControlEditTextSelectionRec` in the `parms` parameter of the `NavCustomControl` function in order to specify which part of the filename string to highlight. For more information on the `ControlEditTextSelectionRec` structure, see *Inside Mac OS X: Control Manager Reference*.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlBrowserSelectAll`

(Navigation Services 2.0 or later.) Tells Navigation Services to select all files in the browser list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlGotoParent`
(Navigation Services 2.0 or later.) Tells Navigation Services to navigate to the parent folder or volume of the current selection.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlSetActionState`
(Navigation Services 2.0 or later.) Prevents Navigation Services from handling certain user actions, such as opening or saving files. This is useful if you want to prevent the dismissal of a dialog until certain conditions are met, for example. Specify which actions to prevent by passing one or more of the constants defined by the `NavActionState` enumeration, described in "Action State Constants" (page 53).

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlBrowserRedraw`
(Navigation Services 2.0 or later.) Tells Navigation Services to refresh the browser list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCtlTerminate`
(Navigation Services 2.0 or later.) Tells Navigation Services to dismiss the current dialog. This constant is similar to `kNavCtlCancel`, except that using `kNavCtlTerminate` does not return an error code.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
The `NavCustomControlMessage` data type defines constants that your application can pass in the `selector` parameter of the function `NavCustomControl` (page 25) to control various aspects of the active dialog.

Dialog Configuration Options

Specify dialog configuration options.

```
typedef UInt32 NavDialogOptionFlags;
enum {
    kNavDefaultNavDlogOptions = 0x000000E4,
    kNavNoTypePopup = 0x00000001,
    kNavDontAutoTranslate = 0x00000002,
    kNavDontAddTranslateItems = 0x00000004,
    kNavAllFilesInPopup = 0x00000010,
    kNavAllowStationery = 0x00000020,
    kNavAllowPreviews = 0x00000040,
    kNavAllowMultipleFiles = 0x00000080,
    kNavAllowInvisibleFiles = 0x00000100,
    kNavDontResolveAliases = 0x00000200,
    kNavSelectDefaultLocation = 0x00000400,
    kNavSelectAllReadableItem = 0x00000800,
    kNavSupportPackages = 0x00001000,
    kNavAllowOpenPackages = 0x00002000,
    kNavDontAddRecents = 0x00004000,
    kNavDontUseCustomFrame = 0x00008000,
    kNavDontConfirmReplacement = 0x00010000,
    kNavPreserveSaveFileExtension = 0x00020000
};
```

Constants

`kNavDefaultNavDlogOptions`

Tells Navigation Services to use default configuration options. These default options include:

- no custom control titles
- no banner or prompt message
- automatic resolution of aliases
- support for file previews
- no display of invisible file objects
- support for multiple file selection
- support for stationery
- no package support
- all chosen items added to Recent list
- customization area is framed

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavNoTypePopup`

Tells Navigation Services not to display the Show pop-up menu in the Open dialog or the Format pop-up menu in the Save dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontAutoTranslate`

Tells Navigation Services not to do automatic file translation. Normally a file chosen in an Open dialog that requires translation is automatically translated. Navigation Services informs your application that a file needs translating by setting the `translationNeeded` field of the `NavReplyRecord` (page 47) structure to `true`. A translation specification array specified in the `fileTranslation` field of the `NavReplyRecord` structure contains the associated translation specification records. When you set the `kNavDontAutoTranslate` flag, your application is responsible for translation, either by calling the function `NavTranslate` or by performing the translation itself.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontAddTranslateItems`

Tells Navigation Services not to display file translation options in the Show pop-up menu.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllFilesInPopup`

Tells Navigation Services to add a pop-up menu item called All Documents, so the user can see a display of all files in the current directory.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllowStationery`

Tells Navigation Services to display a Stationery Option command in the Format pop-up menu of Save dialogs, so users can choose to save a file as a document or as stationery. This is a default option.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllowPreviews`

Tells Navigation Services to provide previews, when available, of selected files. This is a default option.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllowMultipleFiles`

Tells Navigation Services to allow users to select and open multiple files in the browser list by shift-clicking or using the Select All command. If you don't specify this constant, users can select multiple files for drag-and-drop operations, but the default button (normally titled Open) is disabled when multiple items are selected. Note that the user cannot add folders or volumes to a multiple selection.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllowInvisibleFiles`

Tells Navigation Services to show invisible file objects in the browser list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontResolveAliases`

Tells Navigation Services not to resolve any alias selected by the user. If the user selects an alias with this option set, the file system specification returned by Navigation Services designates the alias file instead of its referenced original.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSelectDefaultLocation`

Tells Navigation Services to select the default location in the browser list. By default, Navigation Services will open the browser list with the default location displayed, not selected. For example, if you define the System Folder as the default location and specify the `kNavSelectDefaultLocation` constant, the System Folder appears as the current selection in the browser list. Without this constant, the browser list displays the contents of the System Folder.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSelectAllReadableItem`

Tells Navigation Services to show All Readable Documents as the default selection in the Show pop-up menu when the Open dialog is first displayed. If you do not specify this constant, Navigation Services shows the All [AppName] Documents menu item as the default selection in the Show pop-up menu when the Open dialog is first displayed.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSupportPackages`

(Available in Navigation Services 2.0 and later.) Tells Navigation Services to allow packages to be displayed in the browser list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllowOpenPackages`

(Available in Navigation Services 2.0 and later.) Tells Navigation Services to allow packages to be opened and navigated in the browser list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontAddRecents`

(Available in Navigation Services 2.0 and later.) Tells Navigation Services not to add file objects to the Recent list after a dialog is closed. This is useful if you want to allow users to choose long lists of items without cluttering up the Recent list.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontUseCustomFrame`

(Available in Navigation Services 2.0 and later.) Tells Navigation Services not to draw a bevelled border around the customization area. The border is drawn by default, so you must specify this constant if you want to turn it off.

Note: Keep in mind that turning off the border may affect the placement of any controls you create in the customization area. This means your controls may appear differently in different versions of Navigation Services.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavDontConfirmReplacement`
(Mac OS X only.) Tells Navigation Services not to display an alert when the user attempts to save a file over another file with the same name.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavPreserveSaveFileExtension`
(Available in Navigation Services 3.1 and later.) Tells Navigation Services that the extension in the default filename should be preserved between dialog invocations and is initially hidden.

Available in Mac OS X v10.1 and later.

Declared in `Navigation.h`.

Discard Changes Actions

Describe the user response to a Discard Changes dialog.

```
typedef UInt32 NavAskDiscardChangesResult;
enum {
    kNavAskDiscardChanges = 1,
    kNavAskDiscardChangesCancel = 2
};
```

Constants

`kNavAskDiscardChanges`
User clicked the Okay button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAskDiscardChangesCancel`
User clicked the Cancel button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion

After you display a dialog created using the `NavCreateAskDiscardChangesDialog` (page 14) function, your application determines the result of the function call by checking the `eventData` field of a structure of type `NavCBRec` (page 40) for one of the constants defined by the `NavAskDiscardChangesResult` data type.

Event Messages

Define messages sent to your event-handling function.

```
typedef SInt32 NavEventCallbackMessage;
enum {
    kNavCBEvent = 0,
    kNavCBCustomize = 1,
    kNavCBStart = 2,
    kNavCBTerminate = 3,
    kNavCBAdjustRect = 4,
    kNavCBNewLocation = 5,
    kNavCBShowDesktop = 6,
    kNavCBSelectEntry = 7,
    kNavCBPopupMenuSelect = 8,
    kNavCBAccept = 9,
    kNavCBCancel = 10,
    kNavCBAdjustPreview = 11,
    kNavCBUserAction = 12,
    kNavCBOpenSelection = (long)0x80000000
};
```

Constants

`kNavCBEvent`

Tells your application that an event has occurred (including an idle event), which provides an opportunity for your application to track controls, update other windows, and so forth. Your application can obtain the event record describing this event from the `event` field of the `NavCBRec` (page 40) structure. The `kNavCBEvent` constant is the only message that needs to be processed by most applications that do not customize Open and Save dialogs.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBCustomize`

Tells your application to supply a dialog customization request. The `customRect` field of the `NavCBRec` (page 40) structure defines a rectangle in the local coordinates of the dialog; the top-left coordinates define the anchor point for a customization rectangle. If you want to customize the dialog, your application responds to the `kNavCBCustomize` message by setting a value in the `customRect` field that completes the dimensions of the customization rectangle. After your application responds, Navigation Services inspects the `customRect` field to determine if the requested dimensions result in a dialog that can fit on the screen. If the dimensions are too large, then Navigation Services responds by setting the rectangle to the largest size that the screen can accommodate. Your application can continue to "negotiate" by examining the `customRect` field and requesting a different size until Navigation Services provides an acceptable rectangle value, at which time you should create your custom control or item list. The minimum size for the customization area is 400 pixels wide by 40 pixels high.

Note: Don't add new dialog items until your application receives the `kNavCBStart` event message constant.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBStart`

Tells your application that a dialog is ready to be displayed. After receiving the `kNavCBCustomize` event message constant, your event-handling function should wait for the `kNavCBStart` event message constant to ensure that your application can safely add dialog items. No additional data is provided to your application with this constant.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBTerminate`

Tells your application that the dialog is about to be closed, which means you must remove any user-interface items that were created in response to the `kNavCBStart` message. You can determine which user action closed the dialog by checking the `userAction` field of the `NavCBRec` (page 40) structure.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBAdjustRect`

Tells your application that the dialog has been resized and the customization rectangle has been accordingly resized. Use the `customRect` field from the `NavCBRec` (page 40) structure to determine the new customization rectangle size. Your application does not need to offset the controls; Navigation Services moves them automatically. Your application is responsible for any redrawing of the controls or handling events beyond moving the controls, however.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBNewLocation`

Tells your application that a new location is being viewed in the dialog. The `param` field of the `NavCBRec` (page 40) structure contains a pointer to an `AEDesc` structure of type `'typeFSS'` describing the new location. This pointer is valid only during the execution of your event-handling function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBShowDesktop`

Tells your application that the Open or Save dialog is showing the desktop view, consisting of the composite of all desktop folders from all mounted volumes. The `param` field of the `NavCBRec` (page 40) structure contains a pointer to an `AEDescList` structure identifying the desktop location. This pointer is valid only during the execution of your event-handling function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBSelectEntry`

Tells your application that an entry in the browser list has been selected or deselected by the user. The `param` field of the `NavEventDataInfo` structure contains a pointer to an `AEDescList` record of type `'typeFSS'` identifying the current selection. If the user deselects the current selection, the `AEDescList` record contains an empty reference. This pointer is valid only during the execution of your event-handling function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBPopupMenuSelect`

Tells your application that a selection was made from the Open dialog's Show pop-up menu or Save dialog's Format pop-up menu. The`NavCBRec.eventData.eventDataParms.param` field contains a pointer to a `NavMenuItemSpec` structure describing the pop-up menu item selected. If the dialog was created using the Carbon-only `NavCreateXXXDialog` APIs, then the `menuType` field of the `NavMenuItemSpec` structure is set to the index into the client's CFArray of `popupExtension` strings in the `NavDialogCreationOptions` (page 41) structure. This data is valid only during the execution of your event-handling function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBAccept`
: Tells your application that the user has pressed the Accept button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBCancel`
: Tells your application that the user has pressed the Cancel button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBAdjustPreview`
: Tells your application that the user has toggled the preview area on or off. The `param` field of the `NavCBRec` (page 40) structure contains a pointer to a Boolean value of `true` if the preview area is toggled on and `false` if toggled off. This information is useful if your application creates custom controls in the preview area.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBUserAction`
: (Available only in CarbonLib 1.1 and later and in Mac OS X version 10.0 and later.) Tells your application that the user has taken an action. To determine which action the user took, call the `NavDialogGetUserAction` (page 28) function.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavCBOpenSelection`
: (Navigation Services 2.0 or later.) Tells your application that the user has opened a file or chosen a file object. After detecting this constant, you can call the `NavCustomControl` (page 25) function and specify one of the `NavActionState` constants, described in "Action State Constants" (page 53), in order to block the opening or choosing action.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

File Sorting Constants

Let you specify a sort key for how files display in browser.

```
typedef UInt16 NavSortKeyField;
enum {
    kNavSortNameField = 0,
    kNavSortDateField = 1
};
```

Constants

`kNavSortNameField`
: Sort by filename.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSortDateField`
Sort by modification date.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
Your application can determine the sort key for displayed files by passing the `kNavCtlSortBy` constant, described in "Custom Control Settings" (page 54), in the `selector` parameter of the function `NavCustomControl` (page 25), and passing one of the constants defined in the `NavSortKeyField` data type in the `parms` parameter of the `NavCustomControl` function.

Generic File Signature Constant

Defines a generic creator code.

```
enum {
    kNavGenericSignature = '****'
};
```

Constants
`kNavGenericSignature`
Tells Navigation Services to display all files of a specified type, regardless of the file's creator code.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
To specify that your application can open all files of a given type (such as `'TEXT'`, for example), specify the `kNavGenericSignature` constant in the `componentSignature` field of the structure of type `NavTypeList` (page 49) that you pass to a function that creates a file-opening dialog, such as `NavCreateGetFileDialog` (page 21). You can also pass this constant in the `inFileCreator` parameter of the function `NavCreatePutFileDialog` (page 23) in order to override the types of files appearing in the Format pop-up menu.

Version Notes
Added in Navigation Services 2.0

Menu Item Selection Constants

Let you specify the default selection in the Show pop-up menu.

```
typedef UInt16 NavPopupMenuItem;
enum {
    kNavAllKnownFiles = 0,
    kNavAllReadableFiles = 1,
    kNavAllFiles = 2
};
```

Constants
`kNavAllKnownFiles`
Tells Navigation Services to display all files identified as readable by your application.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllReadableFiles`
Tells Navigation Services to display all files identified as readable or translatable by your application.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAllFiles`
Tells Navigation Services to display all files.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
To set the default selection for the Show pop-up menu of an Open dialog box, your application passes the `kNavCtlSelectAllType` constant, described in "Custom Control Settings" (page 54), in the `selector` parameter of the function `NavCustomControl` (page 25) and passes one of the constants defined in the `NavPopupMenuItem` data type in the `parms` parameter of the `NavCustomControl` function.

Object Filtering Constants

Inform you which part of a dialog contains object being filtered.

```
typedef SInt16 NavFilterModes;
enum {
    kNavFilteringBrowserList = 0,
    kNavFilteringFavorites = 1,
    kNavFilteringRecents = 2,
    kNavFilteringShortCutVolumes = 3,
    kNavFilteringLocationPopup = 4
};
```

Constants
`kNavFilteringBrowserList`
The browser list contains the object being filtered.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavFilteringFavorites`
The Favorites pop-up menu contains the object being filtered.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavFilteringRecents`
The Recent pop-up menu contains the object being filtered.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavFilteringShortCutVolumes`
The Shortcuts pop-up menu contains the object being filtered.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavFilteringLocationPopup`
The object being filtered is the path described by the Location menu.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
Navigation Services passes one of the constants defined by the `NavFilterModes` data type to the `filterMode` parameter of your application-defined filter function to tell your application whether the browser list or one of the navigation option pop-up menus contains the object currently being filtered.

Save Changes Actions

Describe the user response to a Save Changes dialog.

```
typedef UInt32 NavAskSaveChangesResult;
enum {
    kNavAskSaveChangesSave = 1,
    kNavAskSaveChangesCancel = 2,
    kNavAskSaveChangesDontSave = 3
};
```

Constants
`kNavAskSaveChangesSave`
User clicked the Save button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAskSaveChangesCancel`
User clicked the Cancel button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavAskSaveChangesDontSave`
User clicked the Don't Save button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
When you create a Save dialog using the function `NavCreatePutFileDialog` (page 23), you obtain the user's response by calling `NavDialogGetReply` (page 26), specifying a `NavReplyRecord` (page 47) structure in the `outReply` parameter. On completion, this structure contains a value represented by one of the constants defined by the `NavAskSaveChangesResult` data type.

Save Changes Requests

Describe the condition that prompts a Save Changes dialog.

```
typedef UInt32 NavAskSaveChangesAction;
enum {
    kNavSaveChangesClosingDocument = 1,
    kNavSaveChangesQuittingApplication = 2,
    kNavSaveChangesOther = 0
};
```

Constants

`kNavSaveChangesClosingDocument`

Requests a Save Changes alert that asks the user whether to save changes when closing a document.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSaveChangesQuittingApplication`

Requests a Save Changes alert that asks the user whether to save changes when quitting your application.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSaveChangesOther`

Requests a Save Changes alert that asks the user whether to save changes at some time other than closing or quitting. This is useful when your application prompts the user to save documents at timed intervals, for example.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion

Your application requests a Save Changes alert by specifying one of the following constants, defined by the `NavAskSaveChangesAction` data type, in the `inAction` parameter of the function `NavCreatePutFileDialog` (page 23).

Sort Order Constants

Let you specify ascending or descending sort order in the browser.

```
typedef UInt16 NavSortOrder;
enum {
    kNavSortAscending = 0,
    kNavSortDescending = 1
};
```

Constants

`kNavSortAscending`

Sort in ascending order.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavSortDescending`

Sort in descending order.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
Your application can specify the sort order for displayed files by passing the `kNavCtlSortOrder` constant in the `selector` parameter of the function `NavCustomControl` (page 25) and passing one of the constants defined in the `NavSortOrder` data type in the `parms` parameter of the `NavCustomControl` function.

Translation Options

Let you specify how files are translated.

```
typedef UInt32 NavTranslationOptions;
enum {
    kNavTranslateInPlace = 0,
    kNavTranslateCopy = 1
};
```

Constants

`kNavTranslateInPlace`
Tells Navigation Services to replace the source file with the translation. This setting is the default for Save dialogs.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavTranslateCopy`
Tells Navigation Services to create a translated copy of the source file. This setting is the default for Open dialogs. The `NavCompleteSave` function always uses this setting under automatic translation.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

Discussion
Your application passes one of the `NavTranslationOptions` constants to the `howToTranslate` parameter to specify how files are to be translated by the function `NavTranslateFile` (page 92).

User Actions

```
typedef UInt32 NavUserAction;
enum {
    kNavUserActionNone = 0,
    kNavUserActionCancel = 1,
    kNavUserActionOpen = 2,
    kNavUserActionSaveAs = 3,
    kNavUserActionChoose = 4,
    kNavUserActionNewFolder = 5,
    kNavUserActionSaveChanges = 6,
    kNavUserActionDontSaveChanges = 7,
    kNavUserActionDiscardChanges = 8,
    kNavUserActionReviewDocuments = 9,
    kNavUserActionDiscardDocuments = 10
};
```

Constants

`kNavUserActionNone`
The dialog is still running or was terminated programmatically.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionCancel`
The user pressed the Cancel button.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionOpen`
The user pressed the Open button in a file-opening dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionSaveAs`
The user pressed the Save button in a file-saving dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionChoose`
The user pressed the Choose button in a Choose dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionNewFolder`
The user pressed the New Folder button in a New Folder dialog and Navigation Services created a folder.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionSaveChanges`
The user pressed the Save button in a Save Changes dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionDontSaveChanges`
The user pressed the Don't Save button in a Save Changes dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionDiscardChanges`
The user pressed the Discard button in a Discard Changes dialog.

Available in Mac OS X v10.0 and later.

Declared in `Navigation.h`.

`kNavUserActionReviewDocuments`
The user clicked the Review Unsaved button in the Review Documents dialog (used only on Mac OS X).

Available in Mac OS X v10.1 and later.

Declared in `Navigation.h`.

`kNavUserActionDiscardDocuments`
The user clicked the Discard Changes button in the Review Documents dialog (used only on Mac OS X).

Available in Mac OS X v10.1 and later.

Declared in `Navigation.h`.

Discussion
Navigation Services passes one of the constants defined by the `NavUserAction` enumeration to your application in the `userAction` field of a `NavCBRec` (page 40) structure in order to indicate which action was taken by the user during a Navigation Services dialog.

NavDialogCreationOptions Version Constant

Represents the current version of the `NavDialogCreationOptions` structure.

```
enum {
        kNavDialogCreationOptionsVersion = 0
};
```

NavCBRec Version Constant

Represents the current version of the `NavCBRec` structure.

```
enum {
    kNavCBRecVersion = 1
};
```

NavFileOrFolder Version Constant

Represents the current version of the `NavFileOrFolder` structure.

```
enum {
    kNavFileOrFolderVersion = 1
};
```

NavMenuItemSpec Version Constant

Represents the current version of the `NavMenuItemSpec` structure.

```
enum {
    kNavMenuItemSpecVersion = 0
};
```

NavReplyRecord Version Constant

Represents the current version of the `NavReplyRecord` structure.

```
enum {
    kNavReplyRecordVersion = 2
};
```

Result Codes

The table below shows the most common result codes returned by Navigation Services.

Result Code	Value	Description
`kNavWrongDialogStateErr`	-5694	Dialog is not in correct state for requested operation (it must be running but is not, or vice versa. Available in Mac OS X v10.0 and later.
`kNavWrongDialogClassErr`	-5695	Requested operation is not valid for this type of dialog. Available in Mac OS X v10.0 and later.
`kNavInvalidSystemConfigErr`	-5696	One or more Navigation Services–required system components is missing or out of date. Available in Mac OS X v10.0 and later.
`kNavCustomControlMessageFailedErr`	-5697	Navigation Services did not accept a control message sent by your application. Available in Mac OS X v10.0 and later.
`kNavInvalidCustomControlMessageErr`	-5698	Your application sent an invalid custom control message. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kNavMissingKindStringErr	-5699	No kind strings were provided to describe your application's native file types. Available in Mac OS X v10.0 and later.

Deprecated Navigation Services Functions

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

Deprecated in Mac OS X v10.5

NavAskDiscardChanges

Displays an alert box that asks the user whether to discard changes to a particular document. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavAskDiscardChanges (
   NavDialogOptions *dialogOptions,
   NavAskDiscardChangesResult *reply,
   NavEventUPP eventProc,
   void *callBackUD
);
```

Parameters

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavAskDiscardChanges`, set up this structure to specify dialog box settings. In this case, the `savedFileName` field is the only one you must supply with a value.

reply

A pointer to a structure of type `NavAskDiscardChanges`. On return, the value describes the user's response to the Discard Changes alert box. For a description of the constants used to represent possible responses, see "Discard Changes Actions" (page 64).

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Discard Changes alert box is not movable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

callBackUD

A pointer to a value set by your application. When the `NavAskDiscardChanges` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

If your application provides a Revert to Saved command, you can use the `NavAskDiscardChanges` function to display a confirmation alert box when a user selects Revert to Saved for a document with unsaved changes. Navigation Services uses the string you supply in the `savedFileName` field of the `NavDialogOptions` structure you passed in the `dialogOptions` parameter to display the alert message, "Discard changes to [savedFilename]?".

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateAskDiscardChangesDialog` (page 14) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In

`Navigation.h`

NavAskSaveChanges

Displays a Save Changes alert box. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavAskSaveChanges (
   NavDialogOptions *dialogOptions,
   NavAskSaveChangesAction action,
   NavAskSaveChangesResult *reply,
   NavEventUPP eventProc,
   void *callBackUD
);
```

Parameters

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavAskSaveChanges`, set up this structure to specify dialog box settings. When calling `NavAskSaveChanges`, the `clientName` and `savedFileName` fields are the only two fields you must supply with values.

action

A value of type `NavAskSaveChangesAction`. Pass a constant describing the user action that prompted the Save Changes alert box. For a description of the constants, see "Save Changes Requests" (page 70).

reply

A pointer to a value of type `NavAskSaveChangesResult`. On return, the value describes the user's response to the Save Changes alert box. For a description of the constants used to represent possible responses, see "Save Changes Actions" (page 70).

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Save Changes alert box is not movable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

callBackUD

A pointer to a value set by your application. When the `NavAskSaveChanges` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function is useful when your application needs to display an alert when the user attempts to close a document or an application with unsaved changes.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateAskSaveChangesDialog` (page 16) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In
`Navigation.h`

NavChooseFile

Creates a simple dialog box that prompts the user to select a file. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavChooseFile (
   AEDesc *defaultLocation,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   NavPreviewUPP previewProc,
   NavObjectFilterUPP filterProc,
   NavTypeListHandle typeList,
   void *callBackUD
);
```

Parameters

defaultLocation

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavChooseFile`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services displays the last location visited during a call to the `NavChooseFile` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavChooseFile` call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavChooseFile`, you can set up this structure to specify dialog box settings. If you pass `NULL` in this parameter, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Choose a File dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

previewProc

A Universal Procedure Pointer (UPP) to your application-defined preview function. Obtain this UPP by calling the function `NewNavPreviewUPP`. A preview function allows your application to draw previews or to override Navigation Services previews. For more information on preview functions, see `NavPreviewProcPtr` (page 39).

filterProc

A Universal Procedure Pointer (UPP) to your application-defined filter function. Obtain this UPP by calling the function `NewNavObjectFilterUPP`. An application-defined filter function determines if a volume, directory, or file should be displayed in the browser list or pop-up menus. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

typeList

A handle to a structure of type `NavTypeList` (page 49). Before calling `NavChooseFile`, you can set up this structure to declare file types that your application can open.

callBackUD

A pointer to a value set by your application. When the `NavChooseFile` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

This function allows the user to choose a single file, such as a preferences file, for an action other than opening. The `NavChooseFile` function is similar to `NavGetFile` (page 87), but is limited to selecting a single file.

The dialog box displayed by the `NavChooseFile` function does not display a Show menu. If you wish to control the files displayed by the browser list or the pop-up menus, you must specify a list of file types in the `typeList` parameter or specify a filter function in the `filterProc` parameter. If you specify a list of file types in the `typeList` parameter, the `NavChooseFile` function ignores the `signature` field of the `NavTypeList` structure. This means that all files of the types specified in the list of file types will be displayed, regardless of their application signature.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateChooseFileDialog` (page 17) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In
`Navigation.h`

NavChooseFolder

Displays a dialog box that prompts the user to choose a folder or volume. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavChooseFolder (
   AEDesc *defaultLocation,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   NavObjectFilterUPP filterProc,
   void *callBackUD
);
```

Parameters

defaultLocation

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavChooseFolder`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services displays the last location visited during a call to the `NavChooseFolder` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavChooseFolder` call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavChooseFolder`, set up this structure to specify dialog box settings. If you pass `NULL` in this parameter, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

filterProc

A Universal Procedure Pointer (UPP) to your application-defined filter function. Obtain this UPP by calling the function `NewNavObjectFilterUPP`. An application-defined filter function determines if a volume, directory, or file should be displayed in the browser list or pop-up menus. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

callBackUD

A pointer to a value set by your application. When the `NavChooseFolder` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function provides a way for your application to prompt the user to select a folder or volume. This might be useful if you need to install application files, for example.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateChooseFolderDialog` (page 18) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In
`Navigation.h`

NavChooseObject

Displays a dialog box that prompts the user to choose a file, folder, or volume. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavChooseObject (
   AEDesc *defaultLocation,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   NavObjectFilterUPP filterProc,
   void *callBackUD
);
```

Parameters

defaultLocation

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavChooseObject`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services displays the last location visited during a call to the `NavChooseObject` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavChooseObject` call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavChooseObject`, set up this structure to specify dialog box settings. If you do not provide this structure, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

filterProc

A Universal Procedure Pointer (UPP) to your application-defined filter function. Obtain this UPP by calling the function `NewNavObjectFilterUPP`. An application-defined filter function determines if a volume, directory, or file should be displayed in the browser list or pop-up menus. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

callBackUD

A pointer to a value set by your application. When the `NavChooseObject` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function is useful when you need to display a dialog box that prompts the user to choose a file object that might be a file, folder, or volume. If you want the user to choose a specific type of file object, you should use the function designed for that type of object; to select a file, for example, use the function `NavChooseFile` (page 79).

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateChooseObjectDialog` (page 19) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In
`Navigation.h`

NavChooseVolume

Displays a dialog box that prompts the user to choose a volume. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavChooseVolume (
   AEDesc *defaultSelection,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   NavObjectFilterUPP filterProc,
   void *callBackUD
);
```

Parameters

defaultSelection

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavChooseVolume`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services displays the last location visited during a call to the `NavChooseVolume` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavChooseVolume` call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling, set up this structure to specify dialog box settings. If you pass `NULL` in this parameter, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Choose a Volume dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

filterProc

A Universal Procedure Pointer (UPP) to your application-defined filter function. Obtain this UPP by calling the function `NewNavObjectFilterUPP`. An application-defined filter function determines if a volume, directory, or file should be displayed in the browser list or pop-up menus. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

callBackUD

A pointer to a value set by your application. When the `NavChooseVolume` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

This function provides a way for your application to prompt the user to select a volume. This might be useful for a disk repair utility, for example.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateChooseVolumeDialog` (page 21) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In

`Navigation.h`

NavCreatePreview

Creates a document preview in a specified file. (Deprecated in Mac OS X v10.5.)

```
OSErr NavCreatePreview (
   AEDesc *theObject,
   OSType previewDataType,
   const void *previewData,
   Size previewDataSize
);
```

Parameters

theObject

A pointer to an Apple Event Descriptor (`AEDesc`) structure specifying the file in which to create the preview.

previewDataType

A four character code specifying the type of preview data to create. If you pass `NULL` in this parameter, Navigation Services creates a preview of type '`PICT`'.

previewData

A pointer to a buffer holding preview data. If you pass `NULL` in this parameter, Navigation Services provides its own data.

previewDataSize

A value specifying the size, in bytes, of the preview data you are providing. If you pass `NULL` in this parameter, Navigation Services provides its own data.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

This function creates a preview of the specified file and stores the data in an appropriate resource. If you call this function without passing in preview data, as in the following snippet, Navigation Services obtains and creates the preview automatically:

```
NavCreatePreview(theObject,0,NULL,0)
```

If the specified file is image-based ('`PICT`', '`JPEG`', etc.), Navigation Services creates a thumbnail custom icon for the file. Navigation Services does not create a custom icon if you pass in your own preview data.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 2.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Navigation.h`

NavCustomAskSaveChanges

Displays a Save Changes alert box with a custom alert message. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavCustomAskSaveChanges (
   NavDialogOptions *dialogOptions,
   NavAskSaveChangesResult *reply,
   NavEventUPP eventProc,
   void *callBackUD
);
```

Parameters

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavCustomAskSaveChanges`, set up this structure to specify dialog box settings. When calling `NavCustomAskSaveChanges`, the `message` field is the only field you must supply with a value.

reply

A pointer to a value of type `NavAskSaveChangesResult`. On return, the value describes the user's response to the Save Changes alert box. For a description of the constants used to represent possible responses, see "Save Changes Actions" (page 70).

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Save Changes alert box is not movable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

callBackUD

A pointer to a value set by your application. When the `NavCustomAskSaveChanges` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

This function is similar to the function `NavAskSaveChanges` (page 78), except that you provide a custom alert message. This function is useful when you need to post a Save Changes alert box at times other than quitting or closing a file. Your application can display this alert box if a specified time interval has passed since the user last saved changes, for example.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateAskSaveChangesDialog` in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality. In order to provide a customized alert message, pass a non-null message string in the `NavDialogCreationOptions` structure passed to `NavCreateAskSaveChangesDialog`.

Declared In

`Navigation.h`

NavGetDefaultDialogOptions

Determines the default attributes or behavior for dialog boxes. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavGetDefaultDialogOptions (
   NavDialogOptions *dialogOptions
);
```

Parameters

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). On return, Navigation Services fills out the structure with default option values that your application can change as needed.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

This function gives you a simple way to initialize a structure of type `NavDialogOptions` (page 52) and set the default dialog box options before calling one of the dialog box display functions. After you create the `NavDialogOptions` structure, you can supply it with the `NavDialogOptions` constants, described in "Dialog Configuration Options" (page 60), to change the configuration options.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends that you adopt the dialog creation functions (`NavCreate...Dialog`); you pass these functions a `NavDialogCreationOptions` (page 41) structure rather than a `NavDialogOptions` (page 52) structure.

Related Sample Code

QTMetaData

Declared In

`Navigation.h`

NavGetFile

Displays a dialog box that prompts the user to select a file or files to be opened. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavGetFile (
   AEDesc *defaultLocation,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   NavPreviewUPP previewProc,
   NavObjectFilterUPP filterProc,
   NavTypeListHandle typeList,
   void *callBackUD
);
```

Parameters

defaultLocation

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavGetFile`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services defaults to the last location visited during a call to the `NavGetFile` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavGetFile` call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavGetFile`, set up this structure to specify dialog box settings. If you pass `NULL` in this parameter, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Open dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

previewProc

A Universal Procedure Pointer (UPP) to your application-defined preview function. Obtain this UPP by calling the function `NewNavPreviewUPP`. A preview function allows your application to draw previews or to override Navigation Services previews. For more information on preview functions, see `NavPreviewProcPtr` (page 39).

filterProc

A Universal Procedure Pointer (UPP) to your application-defined filter function. Obtain this UPP by calling the function `NewNavObjectFilterUPP`. An application-defined filter function determines if a volume, directory, or file should be displayed in the browser list or pop-up menus. For more information on filter functions, see `NavObjectFilterProcPtr` (page 38).

typeList

A handle to a structure of type `NavTypeList` (page 49). Before calling, set up this structure to declare file types that your application can open.

callBackUD

A pointer to a value set by your application. When the `NavGetFile` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion
After your application calls the `NavGetFile` function to display an Open dialog box and the user selects one or more files and clicks the Open button, `NavGetFile` closes the dialog box and returns references to the files to be opened in the `NavReplyRecord` structure. Your application should check the `validRecord` field of the `NavReplyRecord` structure; if this field is set to `true`, your application should open the files specified in the `selection` field of the `NavReplyRecord` structure.

Always dispose of the `NavReplyRecord` structure after completing the file opening operation by calling the function `NavDisposeReply` (page 32). If you fail to use the `NavDisposeReply` function, memory used for the `NavReplyRecord` structure remains allocated and unavailable.

If you use the Show pop-up menu in an Open dialog box, your application must provide adequate kind strings to describe its native file types. For more information on `kind` strings, see Inside Macintosh: More Macintosh Toolbox.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateGetFileDialog` (page 21) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Related Sample Code
QTMetaData

Declared In
`Navigation.h`

NavLibraryVersion

Reports the currently installed version of the Navigation Services shared library. (Deprecated in Mac OS X v10.5.)

```
UInt32 NavLibraryVersion (
   void
);
```

Return Value
An unsigned 32-bit integer. This value represents the version number (in binary-coded decimal) of Navigation Services installed on the user's system.

Discussion
If you want to use features that are present only in a specific version of Navigation Services, use the `NavLibraryVersion` function to determine which version of Navigation Services is installed.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.

Declared In
`Navigation.h`

NavNewFolder

Displays a dialog box that prompts the user to create a new folder. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavNewFolder (
   AEDesc *defaultLocation,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   void *callBackUD
);
```

Parameters

defaultLocation

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavNewFolder`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services displays the last location visited during a call to the `NavNewFolder` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of the `NavNewFolder` function call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavNewFolder`, set up this structure to specify dialog box settings. If you pass `NULL` in this parameter, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

callBackUD

A pointer to a value set by your application. When the `NavNewFolder` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value
A result code. See "Navigation Services Result Codes" (page 75).

Discussion
This function provides a way for your application to prompt the user to create a new folder. This might be useful for creating a project folder, for example.

Availability
Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreateNewFolderDialog` (page 23) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Declared In

`Navigation.h`

NavPutFile

Displays a Save dialog box. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr NavPutFile (
   AEDesc *defaultLocation,
   NavReplyRecord *reply,
   NavDialogOptions *dialogOptions,
   NavEventUPP eventProc,
   OSType fileType,
   OSType fileCreator,
   void *callBackUD
);
```

Parameters

defaultLocation

A pointer to an Apple event descriptor structure (`AEDesc`). Before calling `NavPutFile`, you can set up a structure of `AEDesc` type `'typeFSS'` to specify a default location to be viewed. If you pass `NULL` in this parameter, Navigation Services displays the last location visited during a call to the `NavPutFile` function. If the file system specification in the `AEDesc` structure does not describe a directory or volume, Navigation Services uses the desktop as the default location.

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide data to your application about the results of your `NavPutFile` call.

dialogOptions

A pointer to a structure of type `NavDialogOptions` (page 52). Before calling `NavPutFile`, you can set up this structure to specify dialog box settings. If you pass `NULL` in this parameter, Navigation Services uses the defaults for all options. See "Dialog Configuration Options" (page 60) for a description of the default settings.

eventProc

A Universal Procedure Pointer (UPP) to your application-defined event-handling function. You obtain this UPP by calling the function `NewNavEventUPP`. Implementing an event-handling function allows your application to update windows after the user moves or resizes the dialog box. If you pass `NULL` in this parameter, the Save dialog box is not movable or resizable. For more information on event-handling functions, see `NavEventProcPtr` (page 37).

fileType

A four-character code. Pass the file type code for the document to be saved.

fileCreator

A four-character code. Pass the file creator code for the document to be saved. Under Navigation Services 2.0 or later, you may pass the "Generic File Signature Constant" (page 68) constant if you want to override the types of files appearing in the Format popup.

callBackUD

A pointer to a value set by your application. When the `NavPutFile` function calls your event-handling function, the `callBackUD` value is passed back to your application.

Return Value

A result code. See "Navigation Services Result Codes" (page 75). Note: If you specify the `kNavDontResolveAliases` constant as a dialog box option, as described in "Dialog Configuration Options" (page 60), before calling the `NavPutFile` function, Navigation Services returns a `paramErr (-50)`.

Discussion

After your application calls the `NavPutFile` function to display a Save dialog box and the user selects a location, enters a filename, and clicks OK, `NavPutFile` closes the dialog box and returns references to the file to be saved in the `NavReplyRecord` structure. Your application should check the `validRecord` field of the `NavReplyRecord` structure; if this field is set to `true`, your application should save the file and call the function `NavCompleteSave` (page 13).

If you specify the Format pop-up menu in a dialog box displayed by the `NavPutFile` function, your application must provide adequate kind strings to describe the file types available. If the user uses the Format menu to save a file to a format other than the file's native format, Navigation Services translates the file automatically. If you wish to turn off automatic translation, set to `false` the value of the `translationNeeded` field of the `NavReplyRecord` structure you pass in the `reply` parameter. If you turn off automatic translation, your application is responsible for any required translation.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Carbon Porting Notes

Apple recommends you use the function `NavCreatePutFileDialog` (page 23) in order to take advantage of Mac OS X features like Unicode, long filenames and enhanced modality.

Related Sample Code

QTMetaData

Declared In

`Navigation.h`

NavTranslateFile

Provides a means for files opened through Navigation Services to be read from different file formats. (Deprecated in Mac OS X v10.5.)

```
OSErr NavTranslateFile (
   const NavReplyRecord *reply,
   NavTranslationOptions howToTranslate
);
```

Parameters

reply

A pointer to a structure of type `NavReplyRecord` (page 47). Upon return, Navigation Services uses this structure to provide translation information about the selected files.

howToTranslate

A value of type `NavTranslationOptions`. Pass one of these constants to tell Navigation Services how to perform the translation: either in-place or by making a copy of the file. For a description of the constants, see "Translation Options" (page 72).

Return Value

A result code. See "Navigation Services Result Codes" (page 75).

Discussion

Under automatic file translation, Navigation Services calls the `NavTranslateFile` function as necessary before returning from a file-opening function.

Your application can perform its own translation using the `NavReplyRecord` structure you specified in the `translateInfo` parameter. The `NavReplyRecord` structure contains a list of descriptors for the file or files to be opened and a corresponding list of translation specification records that can be passed to the Translation Manager. To determine if your application has to translate a file, your application can examine the `NavReplyRecord` structure to see if Navigation Services set the `translationNeeded` field to `true`. (The `translationNeeded` field of the `NavReplyRecord` structure is also set to `true` after returning from a `NavGetFile` call during which automatic translation was performed.) If you want to turn off automatic file translation, set the constant `kNavDontAutoTranslate` in the `dialogOptionFlags` field of the structure of type `NavDialogOptions` (page 52) that you pass in the `dialogOptions` parameter of the file-opening function.

If your application uses the `NavTranslateFile` function after opening a file without automatic translation, Navigation Services checks to see if the source location can accept a new file. If the source location is not available (as occurs when the volume is locked or there is insufficient space), Navigation Services prompts the user to select a location in which to save the translated file. The same prompt may occur when automatic translation is enabled in an Open dialog box.

Availability

Available in CarbonLib 1.0 and later when Navigation Services 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Declared In

`Navigation.h`

Document Revision History

This table describes the changes to *Navigation Services Reference*.

Date	Notes
2006-08-16	Made minor technical corrections.
2005-11-09	Added new function, NavDialogSetFilterTypeIdentifiers, fixed incorrect kNavCBPopupMenuSelect description, and clarified usage of filter callbacks with types lists in NavObjectFilterProcPtr discussion.
2005-07-07	Added information to the NavDialogCreationOptions parameter of the function NavCreateAskDiscardChangesDialog.
2004-02-17	Added information to the function `NavCreatePutFile` on how to remove the first item in a Format popup menu.
2003-02-06	Added documentation for the `NavDialogSetSaveFileExtensionHidden`, `NavDialogGetSaveFileExtensionHidden`, and `NavCreateAskReviewDocumentsDialog` functions.
	Documented new constants.
	Fixed typographical errors.

Document Revision History

Index

M

N

O

S

T

U