NSDataLinkManager

Inherits From:	NSObject
Conforms To:	NSCoding NSObject (NSObject)
Declared In:	AppKit/NSDataLinkManager.h

Class Description

An NSDataLinkManager object (also known as a *data link manager* or simply *link manager*) manages data linked from and into a document through NSDataLink objects. NSDataLink objects (or *data links*) provide a link between a selection (NSSelection object) in a source document and a dependent, dynamically updated selection in a destination document. When a user does a Paste and Link command in the destination document, the link manager creates the link in response to a **addLink:at:** message. When this link is added to the destination document, it makes a connection with the source document's link manager, which creates a source link in the source application.

If an application supports data linking, a link manager should be instantiated for every document the application creates. A link manager must be assigned a delegate that assists it in keeping the document up to date; this delegate must implement some or all of the methods listed in the "Methods Implemented by the Delegate" section of this class specification. In addition, the delegate must keep the link manager informed of the state of the document, sending messages such as **noteDocumentEdited** and **noteDocumentSaved** whenever the document is edited, saved, or otherwise altered.

Only applications that support continuously updating links need to be aware of when source links are created; these applications can have the delegate of the destination document's link manager return YES in response to a **dataLinkManagerTracksLinksIndividually:** message, and then respond to **dataLinkManager:startTrackingLink:** messages to receive notifications that source links are created.

For more information about NSDataLink objects and NSSelection objects, see their class descriptions.

Adopted Protocols

NSCoding	– encodeWithCoder: – initWithCoder:

Method Types

Creating an NSDataLinkManager
- initWithDelegate:
- initWithDelegate:fromFile:

Handling links
- addLink:at:
- addLinkAsMarker:at:
- addLinkPreviouslyAt:fromPasteboard:at:
- destinationLinkEnumerator
- destinationLinkWithSelection:
- sourceLinkEnumerator
- breakAllLinks
- writeLinksToPasteboard:
- setLinkOutlinesVisible
- areLinkOutlinesVisible

Informing the link manager of document status
- noteDocumentClosed
- noteDocumentEdited
- noteDocumentReverted
- noteDocumentSaved
- noteDocumentSavedAs:
- noteDocumentSavedTo:

Getting and setting information about the link manager
- delegate
- setDelegateVerifiesLinks:
- delegateVerifiesLinks
- setInteractsWithUser:
- interactsWithUser
- filename
- isEdited

Methods implemented by the delegate

– copyToPasteboard:at:cheapCopyAllowed:
– dataLinkManager:didBreakLink:
– dataLinkManager:isUpdateNeededForLink:
– dataLinkManager:startTrackingLink:
– dataLinkManager:stopTrackingLink:
– dataLinkManagerCloseDocument:
– dataLinkManagerDidEditLinks:
– dataLinkManagerRedrawLinkOutlines:
– dataLinkManagerTracksLinksIndividually
– importFile:at:
– pasteFromPasteboard:at:
– showSelection:
– windowForSelection:

Instance Methods

addLink:at:

– (BOOL)**addLink:**(NSDataLink *)*link*
at:(NSSelection *)*selection*

Adds *link* to the document, indicating that the data in the document described by *selection* is dependent upon the link. This method is invoked as part of the Paste and Link command to actually link in the data that was just pasted. It can also be used at other times; for example, to link to files that are dragged into the document.

This method makes *link* a destination link and sets *selection* as *link*'s destination selection. When the link's source is modified, the link manager's delegate will be sent a **pasteFromPasteboard:at:** or **importFile:at:** message with *selection* as an argument, indicating that the destination data must be updated.

Returns a boolean indicating whether the link was successfully added. There are several situations that will result in failure to add the link, such as an inability to resolve the link to its source, so it's important to check the return value of this method and undo the requested operation if the linking fails.

See also: – addLinkAsMarker:at:

addLinkAsMarker:at:

– (BOOL)**addLinkAsMarker:**(NSDataLink *)*link*
at:(NSSelection *)*selection*

Incorporates *link* into the document as a marker. This method is used to implement link buttons that allow access to the link's source, but are never asked to receive data from the source document. The link button

in the destination document is described by *selection*. This method adds the link and, upon success, sets its the link's update mode to NSUpdateNever. Returns a boolean indicating whether the link was successfully added.

See also: – addLink:at:

addLinkPreviouslyAt:fromPasteboard:at:

– (NSDataLink *)**addLinkPreviouslyAt:**(NSSelection *)*oldSelection*
fromPasteboard:(NSPasteboard *)*pasteboard*
at:(NSSelection *)*selection*

Creates and adds a new destination link corresponding to the same source data as the link described by the destination selection *oldSelection*. The new link's destination selection is provided in *selection*. This method is useful if you paste data that is already linked. It's similar to copying the old link and adding it using **addLink:at:**, except you specify the old destination selection rather than the old link. Before invoking this method, the document's links must be written to *pasteboard* using **writeLinksToPasteboard:**. Returns the new link if it's successfully added, or **nil** if the link can't be added.

See also:

areLinkOutlinesVisible

– (BOOL)**areLinkOutlinesVisible**

Used to inform the link manager's delegate of whether link outlines should be drawn around linked destination data. When the delegate receives a **dataLinkManagerRedrawLinkOutlines:** message, it should query the link manager with an **areLinkOutlinesVisible** message. If this message returns YES, the delegate should call the **NSFrameLinkRect()** function to draw a distinctive link outline around the dependent data.

See also: – setLinkOutlinesVisible

breakAllLinks

– (void)**breakAllLinks**

Breaks all the destination links in the document by sending each link a **break** message. This method is typically invoked by the application's data link panel in response to user input.

See also: **– break** (NSDataLink), **– pickedBreakAllLinks:** (NSDataLinkPanel)

delegate

– (id)**delegate**

Returns the data link manager’s delegate. The delegate is sent messages to provide source data, paste destination data, and help the data link manager keep links up-to-date.

See also: – initWithDelegate:

delegateVerifiesLinks

– (BOOL)**delegateVerifiesLinks**

Return YES if the link manager's delegate will be asked to verify whether data based on the delegate's source links needs to be updated. If so, the delegate should implement the **dataLinkManager:isUpdateNeededForLink:** method.

See also: – setDelegateVerifiesLinks:

destinationLinkEnumerator

– (NSEnumerator *)**destinationLinkEnumerator**

Returns an enumerator of the destination’s source links.

See also: **– destinationLinkWithSelection:**, **– sourceLinkEnumerator**

destinationLinkWithSelection:

– (NSDataLink *)**destinationLinkWithSelection:**(NSSelection *)*destSel*

Returns the destination link for the selection *destSel*, or **nil** if the document has no link for that selection.

See also: **– destinationLinkWithSelection:**,

filename

– (NSString *)**filename**

Returns the filename of the link manager's document. This is the name that was set with the **initWithDelegate:fromFile:** or **documentSavedAs:** method.

initWithDelegate:

– (id)**initWithDelegate:**(id)*anObject*

Initializes and returns a newly allocated link manager for a new document. The link manager's delegate, specified by *anObject*, will be expected to provide source data, paste destination data, and help the data link manager keep links up-to-date. Before data in the document can be linked to, the document will have to be saved and the link manager will have to be informed of the document's name by a **noteDocumentSavedAs:** message.

See “Methods Implemented by the Delegate” at the end of this class specification for information about the methods the delegate should implement to assist the link manager.

See also: – initWithDelegate:fromFile:

initWithDelegate:fromFile:

– (id)**initWithDelegate:**(id)*anObject* **fromFile:**(NSString *)*path*

Initializes a newly allocated link manager for a new document. The link manager's delegate, specified by *anObject*, will be expected to provide source data, paste destination data, and help the data link manager keep links up-to-date. The document's file is specified by the full path *path*. The file must exist or initialization will fail.

Returns the new link manager upon success; frees the allocated storage and returns **nil** if initialization fails.

See “Methods Implemented by the Delegate” at the end of this class specification for information about the methods the delegate should implement to assist the link manager.

See also: – initWithDelegate:

interactsWithUser

– (BOOL)**interactsWithUser**

Returns a boolean indicating whether the link manager will display alert panels to the user when problems occur with links. The default value is YES.

See also: – setInteractsWithUser:

isEdited

– (BOOL)**isEdited**

Returns a boolean indicating whether the document has been edited since the last time it was saved. It’s the application’s responsibility to inform the link manager when the document’s edit state changes, using **noteDocumentEdited**, **noteDocumentSaved**, and related methods.

noteDocumentClosed

– (void)**noteDocumentClosed**

An application should send this message to the link manager to inform it that the manager’s document has been closed.

noteDocumentEdited

– (void)**noteDocumentEdited**

An application should send this message to the link manager to inform it that the manager's document has been edited. If the delegate doesn't track source links individually, this method marks all source links as dirty, indicating that the dependent destination data will eventually need to be updated.

See also: **– dataLinkManagerTracksLinksIndividually:** (NSDataLinkManager)

noteDocumentReverted

– (void)**noteDocumentReverted**

An application should send this message to the link manager to inform it that the manager's document has been reverted to the last saved copy. This method restores the link manager and its links to their last saved state, from the last time the link manager received a **noteDocumentSaved** or **noteDocumentSavedAs:** message.

noteDocumentSaved

– (void)**noteDocumentSaved**

An application should send this message to the link manager to inform it that the manager's document has been saved. This method writes the document's destination link information to a file associated with the document’s path, then initiates updates of other documents dependent upon the document's source links.

See also: **– noteDocumentSavedAs:**, **– noteDocumentSavedTo:**

noteDocumentSavedAs:

– (void)**noteDocumentSavedAs:**(NSString *)*path*

An application should send this message to the link manager to inform it that the manager's document has been saved as the file specified by the full pathname *path*. This method updates the document’s destination links to the new path and writes the destination link information to a file associated with *path*.

See also: **– noteDocumentSaved**, **– noteDocumentSavedTo:**

noteDocumentSavedTo:

– (void)**noteDocumentSavedTo:**(NSString *)*path*

An application should send this message to the link manager to inform it that a copy of the manager's document has been saved to the file specified by the full pathname *path*. This method writes the appropriate link information to a file associated with *path*.

See also: **– noteDocumentSaved**, **– noteDocumentSavedAs:**

setDelegateVerifiesLinks:

– (void)**setDelegateVerifiesLinks:**(BOOL)*flag*

Tells the link manager whether the link manager’s delegate will verify the update status of links. If YES, the delegate must implement the **dataLinkManager:isUpdateNeededForLink:** method to tell the link manager if data based on a source link needs to be updated.

By default, the update status of an individual link isn't verified by the delegate, so the link manager verifies a link based on its last update time. An example where this verification could be incorrect might be a link to a database query; if the query itself doesn't change, the link manager might return that data is up-to-date, even though the database referred to by the query might have changed.

See also: – delegateVerifiesLinks

setInteractsWithUser:

– (void)**setInteractsWithUser:**(BOOL)*flag*

Tells the link manager whether it should display alert panels to the user when link problems occur. The default value is YES.

See also: – interactsWithUser

setLinkOutlinesVisible:

– (void)**setLinkOutlinesVisible:**(BOOL)*flag*

Sets whether link outlines should be displayed, and sends **dataLinkManagerRedrawLinkOutlines** to the link manager’s delegate if the delegate implements it.

sourceLinkEnumerator

– (NSEnumerator *)**sourceLinkEnumerator**

Returns an enumerator of the link manager’s source links.

See also: – destinationLinkEnumerator

writeLinksToPasteboard:

– (void)**writeLinksToPasteboard:**(NSPasteboard *)*pasteboard*

Writes all the link manager's links to *pasteboard* in preparation for an invocation of **addLinkPreviouslyAt:fromPasteboard:at:**, which expects to find one link matching its specified selection.

The links are written with NSPasteboard's **addTypes:owner:** method, which doesn't change the pasteboard's owner or change count, using a private pasteboard type.

Methods Implemented By the Delegate

copyToPasteboard:at:cheapCopyAllowed:

– (BOOL)**copyToPasteboard:**(NSPasteboard *)*pasteboard*
at:(NSSelection *)*selection*
cheapCopyAllowed:(BOOL)*flag*

Implemented by the link manager’s delegate to supply the source data described by *selection* on *pasteboard*. *selection* was previously provided by the application when it created an NSDataLink using **initLinkedToSourceSelection:managedBy:supportingTypes:**.

NSPasteboard works lazily, so the delegate doesn't have to provide all data representations at this time; it simply has to declare the pasteboard types it's willing to provide for selection. Normally, the delegate must put at least one representation on the pasteboard in order to generate any of the specified types when one is requested. However, if *flag* is YES, the system guarantees that no events will be processed by the application before the delegate is requested to provide the specified data; in this case, the application doesn't necessarily have to write any data representations to the pasteboard.

This method should return YES upon success, or NO if the selection can't be resolved.

dataLinkManager:didBreakLink:

– (void)**dataLinkManager:**(NSDataLinkManager *)*sender*
didBreakLink:(NSDataLink *)*link*

If the delegate implements this method, it is invoked to inform the delegate that the destination link *link* was broken and thus data based on the link's destination selection will no longer be updated.

dataLinkManager:isUpdateNeededForLink:

– (BOOL)**dataLinkManager:**(NSDataLinkManager *)*sender*
isUpdateNeededForLink:(NSDataLink *)*link*

This method should return a boolean indicating whether the source data identified by *link*'s source selection has been modified since the link's last update time. If the link manager has been sent a **setLinksVerifiedByDelegate:** message indicating that the delegate will verify the update status of individual links, the delegate should implement this method.

dataLinkManager:startTrackingLink:

– (void)**dataLinkManager:**(NSDataLinkManager *)*sender*
startTrackingLink:(NSDataLink *)*link*

Informs the delegate that another document has established a data link to the link manager's document. The delegate need only implement this method if it returns YES in response to a **dataLinkManagerTracksLinksIndividually:** message. *link* is a newly added source link; the data that it applies to is identified by *link*'s source selection.

dataLinkManager:stopTrackingLink:

– (void)**dataLinkManager:**(NSDataLinkManager *)*sender*
stopTrackingLink:(NSDataLink *)*link*

Informs the delegate that *link* is no longer linked to the document. There are many reasons that a source link might be removed; the destination document could get closed, the link could be explicitly broken, or the destination application might have died.

See also: – **dataLinkManagerTracksLinksIndividually:**

dataLinkManagerCloseDocument:

– (void)**dataLinkManagerCloseDocument:**(NSDataLinkManager *)*sender*

Used for closing documents that were not opened through the user interface. This method is never invoked by the Application Kit classes.

dataLinkManagerDidEditLinks

– (void)**dataLinkManagerDidEditLinks:**(NSDataLinkManager *)*sender*

Informs the delegate that link data has been modified. The link data is stored alongside the document's data and should be considered part of the document, so the delegate should use this notification to mark the document as edited.

dataLinkManagerRedrawLinkOutlines

– (void)**dataLinkManagerRedrawOutlines:**(NSDataLinkManager)*sender*

If the delegate implements this method, it is invoked any time the manager is instructed to show or hide link outlines through **setLinkOutlinesVisible:**. This method should query the link manager with **areLinkOutlinesVisible** to find out if link outlines should be displayed. If so, it should invoke **NSFrameLinkRect()** to draw a distinctive outline around the linked data; otherwise it should display the data without outlines.

dataLinkManagerTracksLinksIndividually

– (BOOL)**dataLinkManagerTracksLinksIndividually:**(NSDataLinkManager *)*sender*

If the delegate wishes to track links individually, it should implement this method and return YES. If the delegate doesn't implement this method, links are not individually tracked.

If the delegate implements this method and returns YES, it should also implement **dataLinkManager:startTrackingLink:** and **dataLinkManager:stopTrackingLink:** to follow the links in use.

To support continuous updating of links, your application needs to track links individually. This is also useful if your application must store selection-state information in the document. It should only do so for selections (and their associated links) that actually get used; many links may be placed on the pasteboard when data is copied, but few of those links will actually ever get used. This method indicates whether the delegate wants to be informed when a link gets used.

importFile:at:

– (BOOL)**importFile:**(NSString *)*filename*
at:(NSSelection *)*selection*

If the application has added a link based on an entire file (that is, used **addLink:at:** to incorporate a link created by **initLinkedToFile:**), the delegate must implement this method to import the file *filename* at the destination described by *selection*.

This method should return YES upon success, or NO if the selection can't be resolved.

pasteFromPasteboard:at:

– (BOOL)**pasteFromPasteboard:**(NSPasteboard *)*pasteboard*
at:(NSSelection *)*selection*

If the application has added an ordinary destination link (that is, used **addLink:at:** to incorporate a link created by **initWithPasteboard:** or a related method), the delegate must implement this method to paste the updated data that has been made available on the pasteboard. The destination for the data is described by *selection*, which was supplied to the link manager as an argument to the **addLink:at:** method.

The data is read from the pasteboard just as it is for any ordinary paste operation; see the NSPasteboard class specification for more information on reading data from a pasteboard.

This method should return YES upon success, or NO if the selection can't be resolved.

showSelection

– (BOOL)**showSelection:**(NSSelection *)*selection*

In an application that serves as a link source, the delegate should implement this method to show the source data for *selection*. This method should scroll the document so the selected data is visible. It might additionally highlight the selected data using the function **NSFrameLinkRect()** with the argument *isDestination* set to NO.

This method should return YES upon success, or NO if the selection can't be resolved.

windowForSelection

– (NSWindow *)**windowForSelection:**(NSSelection *)*selection*

In an application that serves as a link source, the delegate should implement this method to return the NSWindow for the given selection, or **nil** if the selection can't be resolved.