|  |  |
|---|---|
| initFromFile: |  |
|  | initFromPasteboard: |
|  | initLinkedToFile: |
|  | initLinkedToSourceSelection:managedBy: supportingTypes:count: |
|  | copyFromZone: |
| Exporting a link | writeToPasteboard: |
|  | saveLinkIn: |
|  | writeToFile: |
| Information about the link manager |  |
|  | disposition |
|  | linkNumber |
| Information about the link's source |  |
|  | sourceAppName |
|  | sourceFilename |
|  | sourceSelection |
|  | openSource |
|  | lastUpdateTime |
|  | types |
| Information about the link's destination |  |
|  | destinationAppName |
|  | destinationFilename |
|  | destinationSelection |
| Updating the link's data | sourceEdited |
|  | updateDestination |
|  | setUpdateMode: |
|  | updateMode |
|  | break |

break

copyFromZone:(NXZone *)zone

Returns a copy of the receiving data link allocated from zone.  The copy is essentially linked to the
hooked up to the destination document.  The copy has a copy of the receiver's source selection, ha
selection, and its disposition is NX_LinkBroken.

addLink:at: (NXDataLinkManager)

(const char *)destinationAppName

Returns the name (as returned by Application's appName method) of the application containing th

sourceAppName

(const char *)destinationFilename

Returns the file name of the destination document, as set by any of several NXDataLinkManager
destination document.

sourceFilename

(NXSelection *)destinationSelection

Returns the destination selection, which describes how the linked data is represented in the destina

sourceSelection

(NXDataLinkDisposition)disposition

Identifies the link as a source link, a destination link, or a broken link by returning one of the follo

NX_LinkInDestination
NX_LinkInSource
NX_LinkBroken

addLink:at: (NXDataLinkManager), dataLinkManager:startTrackingLink: (NXDataLinkManage

initFromFile:(const char *)filename

Initializes a newly allocated NXDataLink instance from filename, a link that was previously saved
or writeToFile: method.  The new link is generally used by adding it to a destination document's li
addLink:at:.  Returns the link if it is successfully initialized otherwise frees the link and returns nil

saveLinkIn:, writeToFile:, addLink:at: (NXDataLinkManager) writeToPasteboard:, saveLinkIn:, addLink:at: (NXDataLinkManager)

initLinkedToFile:(const char *)filename

Initializes a newly allocated NXDataLink instance corresponding to the entire file filename. The l
link to a file because it has no source selection. Returns the link if it is successfully initialized othe
returns nil.

addLink:at: (NXDataLinkManager), writeToPasteboard:, sourceSelection

initLinkedToSourceSelection:(NXSelection *)selection
        managedBy:linkManager
        supportingTypes:(const char *const *)newTypes
        count:(int)numTypes

Initializes a newly allocated NXDataLink instance corresponding to a selection in the source docur
selection. linkManager is the source document's link manager. newTypes is an array (with size nu
the pasteboard types that linkManager's delegate is willing to provide (by copyToPasteboard:at: ch
when a user of the link requests the data described by selection.

Typically, when the user uses the Copy command to copy linkable data, this method should be inv
corresponding to the data. The new link should be added to the pasteboard (by writeToPasteboard
freed, since it will usually be of no further use to the source document. Many links so placed on th
unused and will simply be discarded when the pasteboard changes. If state identifying selection m
source document, the link manager's delegate should find out whether the selection is used by imp
dataLinkManager:startTrackingLink: (to discover if the selection gets used) and the Pasteboard ow
pasteboardChangedOwner: (to discover when the pasteboard has changed, precluding the use of a
there).

Returns the new link.

copyToPasteboard:at:cheapCopyAllowed: (NXDataLinkManager delegate), dataLinkManager:sta
(NXDataLinkManager delegate), pasteboardChangedOwner: (Pasteboard owner)

(time_t)lastUpdateTime

Returns the last time the link was updated. A link could be updated for many reasons for example,
sent to the source document's link manager telling it that its document was saved, or the link could
with an updateDestination message.

setLinksVerifiedByDelegate: (NXDataLinkManager),
documentSaved (NXDataLinkManager)

(NXDataLinkNumber)linkNumber

Returns a destination link's link-number, which may be useful in identifying the link. This numbe
the life of the document, and unique among the document's links it is not meaningful in source lin

destination link.  Returns self if the source document is successfully opened, nil otherwise.

app:openFile:type: (Application delegate),
showSelection: (NXDataLinkManager delegate)

saveLinkIn:(const char *)directoryName

Saves the link with a file name provided by the user.  This method should be invoked through the I
command.  It runs the SavePanel to prompt the user for a filename to save the link to.  The SavePa
provided in directoryName.  It then writes the link using the writeToFile: method.  Returns self if t
saved nil otherwise.

initFromFile:

setUpdateMode:(NXDataLinkUpdateMode)mode

Sets the link's update mode to mode, which must be one of the following values:

NX_UpdateContinuously
NX_UpdateWhenSourceSaved
NX_UpdateManually
NX_UpdateNever

A mode of NX_UpdateContinuously updates the link's destination data every time a sourceEdited
source link.  A mode of NX_UpdateWhenSourceSaved updates the link's destination data every ti
(or related) message is sent to the source link's link manager.  A mode of NX_UpdateManually up
destination data every time a updateDestination message is sent to the destination link this messag
programmatically or by the data link panel.  A mode of NX_UpdateNever makes the link never up
destination link has been set to this mode, it can't be set back to any other mode until it is broken.
for link buttons, for example.)

This message only has meaning when sent to a destination link or a broken link.  Returns self.

updateMode, break

(const char *)sourceAppName

Returns the name (as returned by Application's appName method) of the application containing th

destinationAppName

sourceEdited

Sent to a source link to inform it that the data referred to by its source selection has changed.  If th
link has been set to update continuously, the destination will be updated.

This message only has meaning if sent to a source link.  An application will only know of the sour
used in its document if the data link manager's delegate tracks links individually and responds to d
startTrackingLink: messages.  However, an application doesn't need to track source links individu
allow continuous updating.

document.

destinationFilename

(NXSelection *)sourceSelection

Returns the source selection, or nil if the link refers to an entire file (in which case the source file c
sourceFilename).

destinationSelection

(const NXAtom *)types

Returns the pasteboard types that the source document can provide when the data for the link's sou
required.

copyToPasteboard:at:cheapCopyAllowed: (NXDataLinkManager delegate)

updateDestination

Updates the data referred to by the link's destination selection. This message can be sent to a sour
destination link. If it's sent to a destination link, it will usually open the source document if it isn't
sent to a source link, it will usually force the destination data to be immediately updated (which is
desirable than sending the source link a sourceEdited message, since that would allow the update t
time). If the destination must be updated, it will be done by sending a pasteFromPasteboard:at: or
message to the destination link manager's delegate.

pasteFromPasteboard:at: (NXDataLinkManager delegate)

(NXDataLinkUpdateMode)updateMode

Returns the link's update mode, which determines when the data referred to by the link's destinatio
updated. Possible return values are:

NX_UpdateContinuously
NX_UpdateWhenSourceSaved
NX_UpdateManually
NX_UpdateNever

A description of these values can be found in the method description for setUpdateMode:.

writeToFile:(const char *)filename

Writes the link into the file filename. This allows selections to be published by the file system the
using initFromFile:. Returns self if the link is successfully written, nil otherwise.

initLinkedToSourceSelection:managedBy:supportingTypes:count:, writeToPasteboard:, saveLin