touch
getDocumentPathIn:
Managing the object hierarchy attachObject:to:
attachObjects:to:
deleteObject:
deleteObjects:
copyObject:type:inPasteboard:
copyObjects:type:inPasteboard:
pasteType:fromPasteboard:parent:
objectIsMember:
getObjects:
getParentForObject:
Setting object names setName:for:
getNameIn:for:
Managing connectors addConnector:
removeConnector:
listConnectors:forSource:
listConnectors:forDestination:
listConnectors:forSource:filterClass:
listConnectors:forDestination:filterClass:
Managing editors setSelectionFrom:
editorDidClose:for:
getEditor:for:
openEditorFor:
Updating the display redrawObject:

addConnector:aConnector

Adds a connector object to the list maintained by Interface Builder. (See the IBConnectors protocol for more information. ) This is the message a custom connection inspector sends Interface Builder's document object to register a connection.

addConnector:

attachObjects:(List *)objectList to:parent

Adds the objects in objectList to the document's object hierarchy by attaching them to parent. Thi
related method attachObject:to:) lets you keep the document's object hierarchy informed of change
the control of your custom editor.

attachObject:to:

copyObject:anObject
type:(NXAtom)type
inPasteboard:(Pasteboard *)aPasteboard

Copies anObject to the specified pasteboard. The type argument can be one of the following:

IBObjectPboardType
IBCellPboardType
IBMenuPboardType
IBMenuCellPboardType
IBViewPboardType
IBWindowPboardType

An editor should send the document object a copyObject:type:inPasteboard: or copyObjects:type:i
as part of its implementation of its copySelection method.

copyObjects:type:inPasteboard:

copyObjects:(List *)objectList
type:(NXAtom)type
inPasteboard:(Pasteboard *)aPasteboard

Copies the objects in objectList to the specified pasteboard. The type argument can be one of the f

IBObjectPboardType
IBCellPboardType
IBMenuPboardType
IBMenuCellPboardType
IBViewPboardType
IBWindowPboardType

An editor should send the document object a copyObject:type:inPasteboard: or copyObjects:type:i
as part of its implementation of its copySelection method.

copyObject:type:inPasteboard:

deleteObject:anObject

Removes anObject from the document's object hierarchy. An editor should send the document obj
deleteObjects: message as part of its implementation of the deleteSelection method. This will keep
accounting of the objects in the nib document in agreement with the actual state of the document.

deleteObjects:

editorDidClose:anEditor for:anObject

Informs the document object that anEditor is no longer active. By sending this message to the doc
close an editor, you keep Interface Builder's record of the active editor up to date. Interface Build
method whenever an editor is closed because of a user action, such as the closing of a window.

getDocumentPathIn:(char *)buffer

Places the document's path in buffer. This is the path displayed as the title of Interface Builder's F
sure that buffer is sufficiently larger to hold the path. Returns the document object.

getEditor:(BOOL)createIt for:anObject

Returns the editor object for anObject. If createIt is YES and the editor hasn't been instantiated, it
returned. If createIt is NO, the editor is returned only if it has already been instantiated. If createIt
hasn't been instantiated, this method returns nil.

getNameIn:(char *)buffer for:anObject

Places the name associated with anObject in buffer. Make sure the buffer you pass in is sufficientl
accommodate the name. Returns the document object.

setName:for:

getObjects:(List *)objectList

Places the objects from the document's object hierarchy into objectList. The object's are not arran
order.

getParentForObject:anObject

Returns the object above anObject in the document's object hierarchy. The top object is the File's
anObject is the File's owner.

listConnectors:(List *)aList forDestination:aDestination

Places in aList all connector objects whose destinations are aDestination. Returns aList.

Since a given object can be the destination of multiple connections, the connection information is r
objects. Each object in the list conforms to the IBConnectors protocol and contains the informatio

aList is a List object that you provide. When you're done with aList, free it but don't free the conn
since they're managed by Interface Builder.

listConnectors:forDestination:filterClass:, listConnectors:forSource:

objects. Each object in the list conforms to the IBConnectors protocol and contains the informatio

aList is a List object that you provide. When you're done with aList, free it but don't free the conn since they're managed by Interface Builder.

listConnectors:forDestination:, listConnectors:forSource:

listConnectors:(List *)aList forSource:aSource

Places in aList all connector objects whose sources are aSource. Returns aList.

Since a given source can have multiple connections, the connection information is returned as a lis object in the list conforms to the IBConnectors protocol and contains the connection information fo

aList is a List object that you provide. When you're done with aList, free it but don't free the conn since they're managed by Interface Builder.

listConnectors:forSource:filterClass:, listConnectors:forDestination:

listConnectors:(List *)aList
forSource:aSource
filterClass:aClass

Places in aList the connector objects of class aClass whose sources are aSource. Returns aList.

Since a given source can have multiple connections, the connection information is returned as a lis object in the list conforms to the IBConnectors protocol and contains the connection information fo

aList is a List object that you provide. When you're done with aList, free it but don't free the conn since they're managed by Interface Builder.

listConnectors:forSource:, listConnectors:forDestination:

(BOOL)objectIsMember:anObject

Returns YES if anObject is a part of the document's object hierarchy NO otherwise. You might se message to the document object before attempting to open a subeditor for anObject.

(BOOL)openEditorFor:anObject

Opens the editor object for anObject. This method ensures that editors for all the objects above an hierarchy are open before opening anObject's editor.

(List *)pasteType:(NXAtom)type
fromPasteboard:(Pasteboard *)thePasteboard
parent:theParent

Alerts the document object that one or more objects were pasted. Returns a List containing the ids pasted. The pasteboard and the type being pasted are identified by thePasteboard and type.

hierarchyDand" sending each editor a resetObject: message.

removeConnector:aConnector

Removes aConnector from the list of connectors maintained by Interface Builder. (See the IBCon more information on connectors.) This is the message a custom connection inspector sends Interfa object to break a connection.

Interface Builder doesn't free aConnector it's your responsibility to do so.

addConnector:

(BOOL)setName:(const char *)name for:anObject

Sets the name associated with the anObject. For objects in the File window, this is the name displa image. Except for objects in the File window, setting an object's name is generally not needed.

getNameIn:for:

setSelectionFrom:anEditor

Registers anEditor as the editor that owns the selection.

When you activate an editor or change the selection, make sure you send this message to the docu Interface Builder informed of the selection's owner. In this way, when the user switches from one from one document to another, Interface Builder can inform the proper editor to display its selectio Builder uses the selection information to determine which inspector to display in its Inspector pane

touch

Marks the document as edited by causing the File window's close box to display a broken ªXº. Re object.