initInStore:	
	initFromBlock:inStore:
	freeFromStore
	+ freeFromBlock:inStore:
	getBlock:andStore:
IXNameAndFileAccess	initWithName:inFile:
	initFromName:inFile:
	freeFromStore
	+ freeFromName:inFile:
	getName:andFile:
IXRecordWriting	addRecord:
	removeRecord:

	getStringValue:inLength:ofIvar:forRecord:
	getObjectValue:ofIvar:forRecord:
IXTransientMessaging	getOpaqueValue:ofMessage:forRecord:
	getIntValue:ofMessage:forRecord:
	getFloatValue:ofMessage:forRecord:
	getDoubleValue:ofMessage:forRecord:
	getStringValue:ofMessage:forRecord:
	getStringValue:inMessage:ofIvar:forRecord:
	getObjectValue:ofMessage:forRecord:

	addAttributeNamed:forSelector:
	hasAttributeNamed:
	removeAttributeNamed:
	attributeCount
Key comparison	setComparisonFormat:forAttributeNamed:
	comparisonFormatForAttributeNamed:
	setComparator:andContext:forAttributeNamed:
	getComparator:andContext:forAttributeNamed:
Setting attribute targets	setTargetClass:forAttributeNamed:
	getTargetName:andVersion:forAttributeNamed:
Accessing attributes	cursorForAttributeNamed:
Getting attribute information	selectorForAttributeNamed:
	attributeNames
Accessing classes	classNames
	attributeNamesForClass:
	recordsForClass:
Discarding records	discardRecord:
	reclaimRecord:
	discards
	clean
Setting attribute descriptions	setDescription:forAttributeNamed:
	getDescription:forAttributeNamed:
Setting parsers	setParser:forAttributeNamed:
	parserForAttributeNamed:
Writing blobs	setValue:andLength:ofBlob:forRecord:
	getValue:andLength:ofBlob:forRecord:

addAttributeNamed:(const char *)aName forSelector:(SEL)aSelector

Creates an attribute for records that respond to aSelector, associates it with name aName, and buil
attribute. Note that records already passivated by the IXRecordManager that respond to aSelector
new index automatically. This may change in a future release. If an attribute already exists with n
otherwise returns non-nil.

removeAttributeNamed:, selectorForAttributeNamed:

(char *)attributeNames

Returns a newline-separated list of the names of all attributes in the IXRecordManager. The sende
responsible for freeing the string returned.

addAttributeNamed:forSelector:

(char *)attributeNamesForClass:aClass

Returns a newline-separated list of the names of all of the attributes maintained by the IXRecordM
for instances of aClass. This includes all of the attributes whose selectors are recognized by instan
whose target class is aClass or one of its superclasses. The sender of this message is responsible fo
returned.

setTargetClass:forAttributeNamed:

(char *)classNames

Returns a newline-separated list of the names of all the classes that have instances stored in the IX
sender of this message is responsible for freeing the string returned.

clean

Removes all discarded records from the receiver. Those records will no longer be reclaimable. Re

discardRecord:, reclaimRecord:, empty (IXRecordReading protocol)

(const char *)comparisonFormatForAttributeNamed:(const char *)aName

Returns a string defining the comparison format of keys in the index named aName, or NULL if on
is a string encoding the Objective C data types that comprise the key for example, ª [3i]º describes
An IXBTree uses this format to determine how to compare keys. For more information on compar
IXComparisonSetting protocol specification.

setComparisonFormat:forAttributeNamed:, getComparator:andContext:forAttributeNamed:

(unsigned int)count

Returns the number of records stored in the IXRecordManager, plus the number of attributes defin
number of records, subtract the return value of attributeCount from the return value of this method

attributeCount, count (IXRecordWriting protocol)

(IXPostingCursor *)cursorForAttributeNamed:(const char *)aName discardRecord:(unsigned int)aHandle

Discards the record identified by aHandle, so that the record can't be read, removed or replaced. r
discarded records, and clean removes all discarded records. Returns self.

reclaimRecord:, clean, discards, removeRecord: (IXRecordWriting)

(IXPostingList *)discards

Returns an IXPostingList containing all records that have been discarded (by sending discardReco
IXRecordManager). This IXPostingList can be used to reclaim the discarded records with reclaim

If the IXRecordManager is asked to read a discarded record (with the IXRecordReading protocol's
FromZone: method), the result will be nil for most purposes the record no longer exists. However,
still have references in the IXRecordManager's attribute indexes. If your code doesn't deal gracef
you can filter posting sets before using them by subtracting the discards from them.

discardRecord:, reclaimRecord:, clean

getComparator:(IXComparator **)aComparator
andContext:(const void **)aContext
forAttributeNamed:(const char *)aName

Returns by reference the function used to compare attribute values, and the context associated with
attribute named aName. If the attribute has a comparison format set instead, the comparator and co
comparator function takes two data items and returns an answer indicating whether the first is less
greater than the second. The context is arbitrary data for use by that function. Returns self.

For more information on comparators, see the IXComparatorSetting protocol specification and the
specification.

setComparator:andContext:forAttributeNamed:, comparisonFormat:forAttributeNamed:

getDescription:(char **)aDescription forAttributeNamed:(const char *)aName

Returns by reference the description for the attribute named aName. The description can be used t
information pertaining to the attribute. Returns self.

setDescription:forAttributeNamed:, addAttributeNamed:forSelector:

getTargetName:(const char **)aName
andVersion:(unsigned int *)targetVersion
forAttributeNamed:(const char *)aName

Returns by reference the name and version of the class that the attribute named aName is defined f
none has been set. If an attribute has a target class set, it will be defined only for records of that cl
Returns self.

setTargetClass:forAttributeNamed:

(BOOL)hasAttributeNamed:(const char *)aName

Returns YES if the IXRecordManager has an attribute named aName, NO if it doesn't.

(IXAttributeParser *)parserForAttributeNamed:(const char *)aName

Returns the parser, if any, assigned to the attribute named aName. The parser will break the return
selector into separate words when the attribute is evaluated.

setParser:forAttributeNamed:

reclaimRecord:(unsigned int)aHandle

Reclaims a record previously discarded with discardRecord:. aHandle is the identifier of the discar
discarded record must be reclaimed in order to access it or remove it completely from the archive (
removes all discarded records at once). Returns self.

discardRecord:, discards, clean

(IXPostingList *)recordsForClass:aClass

Returns an IXPostingList containing all of the records in the IXRecordManager that are direct inst
not of any subclasses of aClass).

removeAttributeNamed:(const char *)aName

Removes the attribute named aName from the IXRecordManager. Records referenced by the attril
affected. Returns self.

addAttributeNamed:forSelector:

(SEL)selectorForAttributeNamed:(const char *)aName

Returns the selector for the message that defines the attribute named aName. Unless the attribute i
class, this message is sent to any record that responds to it in order to evaluate the attribute. Other
records of the attribute's target class (or a subclass of the target class).

addAttributeNamed:forSelector:

setComparator:(IXComparator *)aComparator
andContext:(const void *)aContext

setComparisonFormat:(const char *)aFormat forAttributeNamed:(const char *)aName

Installs a string defining the comparison format of keys in the index named aName. This is a string
Objective C data types that comprise the key for example, ª[3i]º describes an array of 3 integers (a
currently ignored). An IXBTree uses this format to determine how to compare keys. For more inf
comparison formats, see the IXComparisonSetting protocol specification.

comparisonFormat:forAttributeNamed:, setComparator:andContext:forAttributeNamed:

setDescription:(const char *)aDescription forAttributeNamed:(const char *)aName

Sets the description for the attribute named aName to aDescription. The description can be used to
information pertaining to the attribute. Returns self.

getDescription:forAttributeNamed:

setParser:(IXAttributeParser *)aParser forAttributeNamed:(const char *)aName

Assigns the parser aParser to the attribute named aName. The parser will break the return value of
into separate words when the attribute is evaluated. Returns self.

parserForAttributeNamed:

setTargetClass:aClass forAttributeNamed:(const char *)aName

Sets the target class for the attribute named aName to aClass. The attribute will be defined only fo
aClass or any of its subclasses. Your code should set the target class before any records have been
IXRecordManager otherwise, the index for the named attribute may collect references to instances
the restriction is imposed. This behavior may change in a future release, so that records that aren't
from the index when the target class is set. Returns self.

getTargetName:andVersion:forAttributeNamed:

getValue:andLength:ofBlob:forRecord: