initInStore:
initFromBlock:inStore:
freeFromStore
+ freeFromBlock:inStore:
getBlock:andStore:

IXNameAndFileAccess initWithName:inFile:
initFromName:inFile:forWriting:
freeFromStore
+ freeFromName:andFile:
getName:andFile:

addEntryNamed:ofClass:
addEntryNamed:ofClass:atBlock:
addEntryNamed:forObject:

Removing entries freeEntryNamed:
removeName:
empty
reset

Getting entries hasEntryNamed:
getBlock:ofEntryNamed:
getClass:ofEntryNamed:
openEntryNamed:
entries

addEntryNamed:(const char *)aName forObject:anObject

Associates anObject with aName. anObject must conform to the IXBlockAndStoreAccess protocol, and must be a client of the same IXStore as the IXStoreDirectory. Returns the newly created instance, or nil if an entry already exists with the specified name.

Use this method to associate a name with an existing and instantiated store client. If you want to associate a name with a store client that has already been created, but isn't currently instantiated (that is, its data exists in the IXStore, but there's no run-time object accessing it), use addEntryNamed:ofClass:atBlock:. If you want to immediately create a new store client and associate a name with it, use addEntryNamed:ofClass:.

and associates it with aName. Returns the newly created instance.

If an entry already exists for aName, IX_DuplicateError is raised. If aName is NULL or empty, or
don't respond to initInStore:, IX_ArgumentError is raised.

addEntryNamed:ofClass:atBlock:, openEntryNamed:, initInStore: (IXBlockAndStoreAccess pro

addEntryNamed:(const char *)aName
ofClass:aClass
atBlock:(IXBlockHandle)aHandle

Creates an instance of class aClass, reconstitutes it from the block at aHandle by sending initFroml
IXBlockAndStoreAccess protocol method), and associates it with aName. If aHandle is 0, this me
addEntryNamed:ofClass:, and creates a new instance of aClass. Returns the reconstituted or create

Use this method to associate a name with the data for a previously created store client. The stored
created by a previous instance of aClass.

If an entry already exists for aName, IX_DuplicateError is raised. If aName is NULL or empty, or
don't respond to initFromBlock:inStore:, IX_ArgumentError is raised.

addEntryNamed:forObject:, addEntryNamed:ofClass:, openEntryNamed:, initFromBlock:inStor
(IXBlockAndStoreAccess protocol)

empty

Removes all entries from the directory, instantiating the store clients, and freeing them from the st

freeEntryNamed:, freeFromBlock:inStore: (IXBlockAndStoreAccess protocol)

(const char **)entries

Creates and returns a NULL-terminated list of the names of all currently defined entries. The send
responsible for freeing the list, but not the strings in the list, which are NXAtoms.

If space for the array of entries can't be allocated, IX_MemoryError is raised.

freeEntryNamed:(const char *)aName

Removes the named entry from the directory by sending freeFromBlock:inStore to the named entr
Returns self.

empty, freeFromBlock:inStore: (IXBlockAndStoreAccess protocol)

getClass:(id *)aClass ofEntryNamed:(const char *)aName

Returns by reference the class object for the entry named aName, or nil if there is no such entry. R

is no such entry. It's possible to create multiple instances from the same entry your code should av
separate objects may corrupt the data they share in the IXStore if they try to change it.

addEntryNamed:ofClass:, addEntryNamed:ofClass:atBlock:, initFromBlock:inStore: (IXBlockA
protocol)

removeName:(const char *)aName

Removes aName as an entry in the IXStoreDirectory, but doesn't remove the store client. That is,
recovered by handle. Returns self.

reset, initFromBlock:inStore: (IXBlockAndStoreAccess protocol)

reset

Removes all entries in the IXStoreDirectory, but doesn't remove the store clients. That is, the clie
recovered by handle. Returns self.

removeName:, initFromBlock:inStore: (IXBlockAndStoreAccess protocol)