initInStore:
initFromBlock:inStore:
freeFromStore
+ freeFromBlock:inStore:

Retrieving the block and store getBlock:andStore:

freeFromStore

getBlock:(unsigned int *)aHandle andStore:(IXStore **)aStore

Returns by reference the handle of the receiver's boot block, and its IXStore. Also returns self.

Since a store client needs to record its boot block handle and its IXStore to function properly, impl
is simply a matter of putting those values into aHandle and aStore.

initFromBlock:(unsigned int)aHandle inStore:(IXStore *)aStore

Initializes the receiver using existing data from the boot block identified by aHandle in aStore. Th
been created by a previous invocation of the initInStore: method on the original instance of the sto
isn't required to be of the same class as the original creator of the store data, but it must be able to
Returns self if successful, or nil if the receiver can't be initialized (for example, if aHandle doesn't

A block handle of zero should be interpreted as a request for the creation of a new store client. Th
class' implementation of the initInStore: method to simply send initFromBlock:inStore: to self wit
zero.

To implement this method, simply access the data in aHandle to set up a usable state for the client
involve opening other blocks whose handles are stored in the boot block. For example IXBTree in
to read the block at aHandle and to check that the contents of that block form the root node of a BT

initInStore:

initInStore:(IXStore *)aStore

Initializes the receiver, creating a new boot block in store. After initialization, the boot block can b
receiver's data. That block's handle can be retrieved with getBlock:andStore:. Returns self if succ
receiver can't initialize itself.

To implement this method, simply create a block in aStore, record its handle as the boot block, and
initialization values your client may need there. If your client needs to use several blocks within a
those, and store their handles in its boot block. This allows a later instance to retrieve those blocks
initFromBlock:inStore: message. For example, IXBTree implements this method by creating a blo
the root node of a BTree it creates more blocks only as it needs them.

This method is usually implemented to send initFromBlock:inStore: to self with a block handle of
handle of zero is interpreted as a request to create a new store client.

initFromBlock:inStore: