init
copy
free

Creating, copying, and freeing blocks

createBlock:ofSize:
copyBlock:atOffset:forLength:
freeBlock:

Opening and closing blocks openBlock:atOffset:forLength:

readBlock:atOffset:forLength:

Accessing the contents getContents:andLength:
setContents:andLength:

Reducing memory consumption compact

abortTransaction

Reverts the IXStore to the state it was in before the last time it received a startTransaction message enabled. Discards all changes made to blocks that were opened by the current transaction (even if closes those blocks if necessary, and makes them available to other contexts. Any blocks created b transaction are destroyed, any blocks freed are reclaimed, and any blocks resized are restored to th current transaction is terminated, and the transaction in effect, if any, when the current transaction current transaction. If the nesting level is 1 (that is, no transaction is pending), the state reverts to commitTransaction was received. Returns self.

Blocks opened by an enclosing transaction are not affected, even if their contents have been chang the last startTransaction message. If transactions aren't enabled, only the block creations and freei the last commitTransaction message are reverted changes made to the contents of blocks aren't und code never uses startTransaction, it should periodically send commitTransaction to establish a chec abortTransaction.

This method increases the change count of the IXStore, indicating that a change in state has occurr closed blocks.

commitTransaction, startTransaction, nestingLevel, changeCount, closeBlock:

(BOOL)areTransactionsEnabled

Returns YES if transactions are enabled for the IXStore (that is, if the IXStore was ever sent a start Otherwise, it returns NO. You should use this method if you're not sure whether or not to send sta messages, or when invoked by higher-level code that establishes the transaction management polic

The transaction management policy is a property of the contents of an IXStore. If your code copie IXStore that has transactions enabled into an IXStore that doesn't, transactions will be enabled for

startTransaction, nestingLevel

(unsigned int)changeCount

Returns the number of commitTransaction and abortTransaction messages received by the IXStore That is, this number indicates the number of changes made to the IXStore's contents since the run- initialized.

This method is useful for determining if cached pointers to the contents of opened blocks are still v the block opening methods can be avoided. For example, if an object needs to repeatedly access th transaction, it can cache the pointer to the block's contents when it opens the block, along with the then on, whenever the object needs to access the block, it can check the IXStore's change count if increased, then no commits or transactions have occurred since the block was opened, which mean pointer is still valid, and the object can use the pointer safely without having to open the block aga

don't take effect until the transaction that opened it is committed similarly, changes aren't undone that opened the block is aborted. Open blocks are automatically closed when the transaction that o committed or aborted. Returns self.

openBlock:atOffset:forLength:, readBlock:atOffset:forLength:, startTransaction, commitTransac

commitTransaction

Commits all changes made to blocks opened since the last startTransaction, closes the blocks. If th becomes 0, makes the blocks available to other contexts. Any creations, freeings or resizes perfor startTransaction are also committed. The current transaction is terminated, and the enclosing trans the current transaction. Returns self.

Your code may use this message even if transactions aren't enabled the reversal of block-level ope freeing) is supported even in the absence of transactions. commitTransaction commits all such cha last commitTransaction, and abortTransaction cancels all such changes made since the last commit transactions aren't enabled, this method closes all open blocks, making them available to other con outstanding creates and frees.

This method increases the change count of the IXStore, indicating that a change in state has occurr closed blocks.

abortTransaction, startTransaction, changeCount, closeBlock:

compact

Compacts the contents of the IXStore so that they consume as little storage as possible. This meth around physically within the IXStore, and so may take some time to complete. The amount of stor reduced by as much as 50%. Returns self.

If this method is invoked while transactions are pending, the actual compaction will be postponed transactions outstanding. When used with IXStoreFile, this method actually reduces the size of the may occur automatically. This won't occur unless the IXStore consumes at least 16 MB of storage until much more storage is actually consumed.

copy

Creates and returns a new store context, which addresses the same storage as the original. Change context will affect the shared storage, and will be reflected in both contexts.

If you want to create a completely independent duplicate of an IXStore, you can use getContents:a setContents:andLength: as follows:

anOffset and aLength.

If there is no block identified by aHandle, IX_NotFoundError is raised. If the block has been open
IX_LockedError is raised. See the class description for more information on when a block become
contexts.

openBlock:atOffset:forLength:, readBlock:atOffset:forLength:, abortTransaction, commitTransa

createBlock:(unsigned int *)aHandle ofSize:(unsigned int)size

Creates a new block of size bytes and returns its handle by reference in aHandle. The new block is
zeroed. If you create a block of size vm_page_size or more, it's guaranteed to be page-aligned (vm
in the header file mach/mach_init.h). It isn't possible to create a block of size 0. Returns self.

free

Frees the IXStore. The storage substrate is also freed if there are no other store contexts addressing

freeBlock:

freeBlock:(unsigned int)aHandle

Removes and frees the block identified by aHandle. Returns self.

If there is no block identified by aHandle, IX_NotFoundError is raised. If the block has been open
IX_LockedError is raised. See the class description for more information on when a block become
contexts.

free, abortTransaction, commitTransaction, closeBlock:

getContents:(vm_address_t *)theContents andLength:(vm_size_t *)aLength

Returns by reference the address and length of a copy of the IXStore's contents. theContents is a c
the original (vm_address_t is declared in the header file mach/mach_types.h). Returns self.

Your code can use this method along with setContents:andLength: to create an independent copy o
copy method description for an example). Be sure to compact the IXStore before invoking this me
amount of memory copied is as small as possible. These methods also provide an efficient means
of an IXStore into an IXStoreFile.

getContents:andLength: must not be invoked when transactions are pending if it is, IX_ArgumentE
code should also not invoke this method while any blocks are open outside the scope of a transacti
have been changed).

setContents:andLength:, copy

init

abortTransaction, commitTransaction, areTransactionsEnabled, startTransaction

(void *)openBlock:(unsigned int)aHandle
atOffset:(unsigned int)anOffset
forLength:(unsigned int)aLength

Returns a pointer to a region of the block identified by aHandle, beginning at anOffset and of aLen
it for writing. If your code writes outside of the opened area, your data may become corrupt. Neit
nor commitTransaction will restore data damaged in this manner.

readBlock:atOffset:forLength:, freeBlock:, abortTransaction, commitTransaction, closeBlock:

(void *)readBlock:(unsigned int)aHandle
atOffset:(unsigned int)anOffset
forLength:(unsigned int)aLength

Returns a pointer to a region in the block identified by aHandle, beginning at anOffset and of aLen
it for reading. It's assumed that your code won't alter the block. If your code does alter the block,
corrupt, and neither abortTransaction nor commitTransaction will restore data damaged in this ma

openBlock:atOffset:forLength:, freeBlock:, abortTransaction, commitTransaction, closeBlock:

resizeBlock:(unsigned int)aHandle toSize:(unsigned int)aSize

Resizes the block identified by aHandle to aSize. Returns self.

If there is no block identified by aHandle, IX_NotFoundError is raised. If the block has been open
IX_LockedError is raised. See the class description for more information on when a block become
contexts.

sizeOfBlock:, openBlock:atOffset:forLength:, readBlock:atOffset:forLength, abortTransaction,
closeBlock:

setContents:(vm_address_t)someContents andLength:(vm_size_t)aLength

Replaces the contents of the IXStore with the contents specified by someContents and aLength. T
the IXStore are lost. someContents should be a virtual memory image retrieved by getContents:an
(vm_address_t is declared in the header file mach/mach_types.h). The IXStore assumes responsib
virtual memory image, and may simply use it directly. Contents copied in this manner between ins
shared as copy-on-write data. Returns self.

(unsigned int)sizeOfBlock:(unsigned int)aHandle

Returns the size, in bytes, of the block identified by aHandle.

If there is no block identified by aHandle, IX_NotFoundError is raised. If the block has been open IX_LockedError is raised. See the class description for more information on when a block become contexts.

resizeBlock:toSize:, openBlock:atOffset:forLength:, readBlock:atOffset:forLength, abortTransa commitTransaction, closeBlock:

(unsigned int)startTransaction

Begins a new transaction, which will be aborted or committed before all other outstanding transact context. If transactions aren't enabled for the IXStore, they're permanently enabled. Returns a nu new transaction, and indicating the number of transactions outstanding, including the new one. Th returned by the nestingLevel method. For example, if the nesting level is 0 and the IXStore receiv three times, the invocations of the method will return, in order, 1, 2, 3.

abortTransaction, commitTransaction, areTransactionsEnabled, nestingLevel