3.3 Release Notes: Indexing Kit

This file contains release notes for the 3.2, 3.1, and 3.0 releases of the Indexing Kit. There are no changes for Release 3.3. Items specific to the 3.2 release are listed first, and the Release 3.1 and 3.0 notes follow.

Notes Specific to Release 3.2

New Features

New UNIX program

This functionality of the **ixdomain** program has been replaced by **ixparse**, which offers more options.   See the UNIX manual page for more information.

New method for thread-safe filtering

A new class method had been added to IXAttributeParser to handle problems in multi-threaded access to the Application Kit's Filter Services:

getFilterHandler:
+ (DPSPortProc) **getFilterHandler:**(port_t)*filterPort*

Returns a DPSPortProc for *filterPort*, which can be used with the Application Kit's Filter Services to prevent thread conflicts. Your code should create *filterPort* using **port_allocate()** (a standard Mach operating system function) and then invoke this method within its main thread of execution. *filterPort* and the returned DPSPortProc should then be passed to **DPSAddPort()** (a Display PostScript function). This allows background threads executing through the IXAttributeParser class to dispatch file filtering requests to the main thread of the application.

Bugs Fixed

These (and a number of other minor bugs) have been fixed in Release 3.2:

Reference	37361
Problem	RTF parsing/generation was broken in 3.1
Description	A change made in Release 3.1 skewed the RTF token numbering.

	Consequently, RTF parsing and generation doesn't work correctly.
Reference	37360
Problem	IXWeightingDomain wrote bogus cookie in attribute value
Description	When building a weighting domain from an attribute table, the weighting domain constructs a hash table, chaining the attribute values through their cookie fields. The chaining pointers are never cleared, so when the cookies are freed, the heap gets corrupted, and the application crashes.
Reference	36369, 37013
Problem	IXStoreFile would grow too large
Description	Certains patterns of usage of an IXStoreFile would cause it to grow without limit (a file equivalent of a memory leak). This has been fixed.
Reference	36683
Problem	Searching in Help Panel could cause application to crash
Description	A shared library conflict between the Indexing Kit and other

	software caused the same memory location to be used to different purposes. In the conflict, data could be corrupted and the application could crash.
Reference	35955
Problem	IXLanguageReader didn't search **~/Library** and **/LocalLibrary**
Description	Prior to Release 3.2, IXLanguageReader looked only in **/NextLibrary/Readers** for installed reader bundles, making it difficult for third parties to install their own readers. It now follows the standard resource search path.
Reference	35954
Problem	IXPostingCursor usage of alloca smashes when out of stack space
Description	This problem caused seemingly random crashes in applications using the Indexing Kit.
Reference	35909
Problem	The language specific weighting domain wasn't used by default
Description	When IXAttributeParser allocated a default reader, it allocated a

reader for the system default language. This was appropriate, but insufficient. It now also allocates a weighting domain for the system default language.

Reference 35764

Problem IXFileFinder reported attributes incorrectly when there was no store

Description This had the effect of rendering file name queries empty, since information in the store is used to determine whether or not recursive descent is needed.

Reference 35762

Problem Use of Pasteboard in IXFileConverter is not thread safe

Description This would cause applications to intermittently hang or crash. A new method has been added to IXAttributeParser to correct this. See ªNew Featuresº above for a description.

Reference 35760

Problem Failure to lock around calls to **NXUniqueString()** hangs application

Description The indexing kit called **NXUniqueString()** without locking against concurrent access by other threads. Since this function is not thread safe, bad things would happen.

Reference 35759

Problem Raising an exception while parsing locked the application

Description The yacc-generated parser used a mutex to protect access to the yacc-generated statics. The lock is not protected by an NX_DURING. Raising an exception within the parser left the parser locked, so the next call to the parser would hang.

Reference 35758

Problem Improper error reporting by IXStore made debugging difficult

Description An internal component of the indexing kit was raising the wrong type of exception during normal usage, making it look like an error.

Reference 35756

Problem Compaction didn't work properly on Intel machines

Description On the i486, compaction didn't converge. If there was a time limit, it would work until the time limit, but make no progress. If there was no time limit, it would hang indefinitely.

Reference	35329
Problem	A logic error prevents floppy ejection when store file is freed

Description Freeing a store file didn't close the file itself, causing Workspace Manager to think that the application was still using it, and preventing it from ejecting a floppy disk.

Reference	34978
Problem	IXPostingSet.h didn't import **remote/transport.h**

Description As stated.

Reference	34885
Problem	**ixbuild** didn't invoke man page title filter when indexing user added man pages

Description This bug caused descriptions not to be properly generated for man pages.

Reference	33215
Problem	IXStore typed stream unarchiving can cause error

Description If an object couldn't initialize itself during unarchiving and returned **nil**, the unarchiving code still sent a message to the freed object (for example, **awake**) which caused the application to crash.

Reference	31437
Problem	**IXAbsolutePath()** returns incorrect results for nonexisting files

Description An internal variable wasn't initialized if a **stat()** call failed, making subsequent uses of the variable incorrect.

Notes Specific to Release 3.1

Known Problems

Reference:	34403
Problem:	Store files created on Release 3.1 cannot be read on Release 3.0.

Description:	Because of file format changes needed to fix bugs, store files created by applications running under NEXTSTEP Release 3.1 can't be read by applications running under 3.0. Applications running under 3.1 can still read 3.0 store files, though.
Workaround:	None.

Bugs Fixed

These (and a number of other minor bugs) have been fixed in Release 3.1:

Reference	31594
Problem	IXStoreFile's **commitTransaction** would hang for several minutes on large files.
Description	Committing a transaction on a large store file causes pages to be synchronized in VM. When modified pages aren't in sorted clusters, this takes a very long time. Sorting modified pages and then synchronizing them improved commit time dramatically.

Reference	31573, 31232, 29636
Problem	Several crashes related to the use of attributes in

IXRecordManager have been repaired, including one that appeared to occur when freeing the record manager.

Description Various errors in managing freed or uniqued objects would cause the application to crash. One bug occurred when freeing an IXRecordManager which had applied an attribute to two or more classes which didn't respond to the attribute's selector.

Reference 31563

Problem IXStoreFile didn't set the close-on-exec for the store file to prevent the lock from being held by a child process.

Description A child process shouldn't have an open reference to a store file, since they have to be locked and in use by only one process at a time. Setting the close-on-exec flag for the store file solved this problem.

Reference 31554

Problem IXPostingList's **copyFromZone:** would produce a populated but unusable copy of the original.

Description Some instance variables weren't copied, resulting in a corrupt copy. This has been fixed.

Reference	31112, 30179, 30137
Problem	Several crashes related to modifying blocks in IXStore have been repaired.
Description	Certain internal errors would corrupt the IXStore and cause the application to crash. These have been fixed.

Reference	31110
Problem	IXPostingCursor with large contents didn't work.
Description	Under 3.0, IXPostingCursor could crash the application when more than four pages worth of postings were created for any key. This limit has been corrected.

Reference	30902
Problem	IXRecordManager couldn't store objects with **Nil** instance variables of type Class.
Description	Attempting to add an object that has an instance variable of type Class, where the value of that instance variable is **Nil**, would cause an exception.

Reference	30569
Problem	IXFileFinder could crash the application when generating descriptions for indexed files.
Description	An uninitialized variable would cause the application to crash. This has been repaired.

Reference	30525
Problem	IXStoreFile didn't flush modified pages all the way to the disk.
Description	The VM got flushed, but the store file itself didn't. Adding a call to **fsync()** for the file solved the problem.

Reference	30306
Problem	**ixbuild** would crash when the final component target directory name begins with `.'.
Description	One part of **ixbuild** was assuming that `hidden' directories shouldn't be indexed, while a later part did. Since certain state for the hidden directory hadn't been established, the application would crash.

Reference	30304
Problem	Searching by file name with IXFileFinder would invoke file filtering.
Description	IXFileFinder would actually look at the contents of files when searching by file name. It doesn't bother doing this now.

Reference	30209
Problem	Attributes on selectors implemented by private attribute classes would include instances of those classes created by IXRecordManager.
Description	IXAttribute objects would be implicitly added to attributes within the IXRecordManager. This is no longer the case.

Reference	30195, 28338, 26985
Problem	Various crashes would occur when committing or aborting transactions.
Description	Several crashes related to committing and aborting transactions have been repaired. Some of these appeared at run time to originate in IXStore's **free** method.

Reference	30135
Problem	A crash related to repeated modifications of small blocks in IXStore has been repaired.
Description	After a certain number of operations, an IXStore or IXStoreFile would cause an internal variable to wrap from its maximum possible value, causing an invalid value to be returned or the application to crash. This also affected storage files, since the variable is kept there for IXStoreFiles.
Reference	30068
Problem	IXRecordManager's **addRecord:** would crash when storing objects with a very large number of instance variables.
Description	IXRecordManager assumed that run-time data for an object would never occupy more the **vm_page_size**, an incorrect assumption.
Reference	29958, 29957, 29244
Problem	Various IXBTreeCursor bugs.
Description	Several crashes related to the synchronization IXBTreeCursors have been repaired. These may have appeared to the user of

IXBTreeCursor to be related to using two or more IXBTreeCursors simultaneously, or to emptying the IXBTree.

Reference	29915
Problem	IXPostingCursor's **count** and **empty** methods would choke on empty attributes.
Description	When used with an empty IXBTree, IXPostingCursor's **count** and **empty** methods raised exceptions. This has been fixed.

Reference	29913
Problem	IXRecordManager's **count** returns the wrong value.
Description	In order to maintain backward compatibility with existing applications, this method still returns the number of user objects plus the number of attributes. A new method, **attributeCount**, has been added to permit computation of the user object count.

Reference	29912
Problem	IXRecordManager's **removeRecord:** always returned **self**, although the documentation states that it returns **nil** if the record does not exist.

Description	As stated above. This has been fixed, and the method behaves as documented.

Reference	29911
Problem	IXPostingList would crash the application when passed as an argument or return value with Distributed Objects.
Description	IXPostingList was incorrectly encoding/decoding data passed across the network.

Reference	29712, 24101, 23248
Problem	Several crashes related to thread safety have been repaired. All of these affected Digital Librarian.
Description	Note that care must still be exercised when using background threads with IXFileFinder, since any use of the Pasteboard by the main thread while the background threads are running can still cause thread safety problems.

Reference	29644, 29637
Problem	Several crashes relating to the use of blobs in IXRecordManager

	have been repaired.
Description	IXRecordManager's **removeRecord:** method would not remove blobs associated with the target record unless blobs have been used since the IXRecordManager was initialized. Also, sometimes an exception would occur when the IXRecordManager did try to remove a blob.
Reference	29383
Problem	Several crashes relating to initializing an IXStoreFile or a store client like IXFileFinder have been repaired.
Description	A zone **malloc()** library bug occasionally caused IXStore/IXStoreFile initialization methods to crash the application.
Reference	26984
Problem	Having IXStore allocate a block of length 0 would crash the application.
Description	IXStore incorrectly recorded block information if a block of size 0 is requested. This could cause the application to crash.
Reference	28321

Problem	Invoking **clean** on an IXFileFinder or IXRecordManager would crash the application.
Description	An uninitialized variable would cause the application to crash.
Reference	27119
Problem	IXPostingList's **objectAt:** occasionally returns an invalid object identifier, usually a small integer value.
Description	A memory-allocation error caused a buffer not to be properly zeroed, causing data to be corrupted.
Reference	25542
Problem	IXRecordManager's **selectorForAttributeNamed:** method can crash the application.
Description	An internal object was being sent a message it didn't respond to. This condition is now checked for.
Reference	25512
Problem	If an IXRecordManager was created in a file and then freed, it

	couldn't be reopened.
Description	IXRecordManager's **initWithName:inFile:** method didn't record the fact that it had created and opened an IXStoreFile, so that when a method that needed to open the storage file was invoked again, it noticed that the file was already opened by someone else, and failed to open the storage file.
Reference	25335
Problem	The **fileFinder** instance variable of IXFileRecord was not set on retrieval from an IXFileFinder.
Description	Not setting **fileFinder** caused files not to be retrievable, since their full paths couldn't be reconstructed without the IXFileFinder's root path.
Reference	24483
Problem	IXRecordManager's **classNames** returned a single class name, usually ªObjectº, instead of the names of the stored classes.
Description	The length of a string was being improperly calculated, causing an error in the returned string array.

Reference	29061, 24490
Problem	Using regular expressions would produce incorrect results or crash the application.
Description	Regular expressions in the IXAttributeQuery object resulted in a memory overrun. This has been corrected.

Other Changes

Most, but not all, of changes made to the Indexing Kit for 3.1 are documented here. Comments in the header files may provide additional information on some subjects. Some additional changes made to the Indexing Kit for Release 3.1 include:

- The IXBlobWriting and IXRecordDiscarding protocols are no longer supported, although they are still declared in **indexing/protocols.h**.   The methods they declared have been moved to the interface declaration for IXRecordManager.

- The IXAttributeReading protocol is no longer supported, although it is still declared in **indexing/IXAttributeReader.h**.   The method it declared has been moved to the interface declaration for IXAttributeReader.

- Store file locking with **flock**(2) is not longer implemented, due to a lock reclamation problem in the kernel affecting any process that calls both **flock**(2) and **fork**(2).

Known Problems

Due to schedule limitations, not all of the known Indexing Kit problems have been addressed for Release 3.1. Remaining problems, not identified in the 3.0 Release Notes, include:

- IXAttributeQuery implements a subset of the Indexing Kit query language. In particular, only the search operators can bind attribute values.

- IXAttributeQuery handles a **prefix** or **within** search on two or more terms incorrectly, resulting in very slow turn around.

- Your application will crash if you abort (a) the first transaction that adds a record to a newly created record manager, (b) a transaction that adds an instance of a class that has not been seen before by the record manager, or (c) the first transaction that creates a blob. In general, you should schematize the file before aborting any transactions.

- The Indexing Kit does not support concurrent updates to a store file by multiple processes, either on a single machine or over a network. If a store file must be shared by multiple processes, and at least one of them may modify the file, then a server must be introduced that provides a single point of access to the file through the Indexing Kit.

Notes Specific to Release 3.0

These notes were included with the Release 3.0 version of the Indexing Kit. Sections that are no longer relevant have been marked with an italicized comment.

The following paragraphs describe the goals and purpose of the IndexingKit, its feature set, and its programming interface.

The IndexingKit is a library of Objective C classes designed to simplify the management of structured persistent data. With the IndexingKit, it is easy to build fast, lightweight databases that store structured data and invert its attributes.

Salient features of the IndexingKit include fast storage and retrieval primitives, transaction management that guarantees data integrity, fast associative access methods, Objective C based record management, tight integration with the NEXTSTEP © programming environment, and fast file system searching. Storage managed by the IndexingKit may reside in files or in virtual memory; the IndexingKit uses low level virtual memory primitives to attain excellent paging performance. Most of the classes in the IndexingKit are fully thread safe.

There are three foundation classes in the IndexingKit: IXStore, IXBTree and IXRecordManager; these classes address storage management, associative access, and data modelling, respectively. In addition, there are two BTree cursor classes and a class that manages set operations; IXPostingList, for the lazy instantiation of retrieved record objects; IXFileFinder, for searching files; IXAttributeQuery, a query language interface to IXFileFinder and IXRecordManager, and several lexical analysis classes for harnessing textual data.

IXStore is a storage manager that provides a stable heap abstraction, with transaction management to guarantee data integrity. Callers allocate relocatable blocks of storage from IXStore using block handles as addresses. Blocks can be resized and modified in whole or in part. IXStore shadows the modified portions of each block to ensure a consistent view in the event of unexpected interruption. IXStore compacts free space on demand, and automatically garbage collects the storage discarded by shadowing.

IXBTree implements indexed sequential access on top of IXStore. It is very fast, and is an excellent tool for building custom data structures. Two cursor classes, IXBTreeCursor and IXPostingCursor, provide cursoring for primary and secondary keys, respectively. A utility class, IXPostingSet, implements set combinatorial operations on lists of postings returned by IXPostingCursor, and another utility class called IXStoreDirectory uses IXBTree to implement a simple recursive naming facility for organizing the contents of an IXStore.

IXRecordManager is a a simple record manager based on IXBTree; it lets callers store and retrieve records in the form of Objective C objects, and obtain cursors on indexes that invert method return values. Callers use a sub- class of

IXPostingCursor to enumerate the indexes, and to locate record objects by key. IXPostingList is a lazy list that manages large numbers of retrieved record objects efficiently; instances of IXPostingList may be initialized directly from instances of IXPostingCursor. The IXPostingSet and IXPostingList classes work efficiently with Distributed Objects®.

IXFileFinder is a client of IXRecordManager that answers queries against UNIX® directories. As a replacement for the libtext library distributed in prior releases, this class builds instances of the IXFileRecord class to describe files, and stores them in an instance of IXRecordManager. The lexical analysis classes, IXAttributeReader, IXLanguageReader, IXAttributeParser and IXWeightingDomain, make it easy for developers to reduce unstructured text into attribute value lists that are directly queryable by IXAttributeQuery. IXAttributeQuery implements a query language for selecting objects from several evaluation contexts including IXRecordManager.

Known Problems

The following are known problems in the Indexing Kit. They will be corrected in a future release.

- When an attribute is added to IXRecordManager, pre-existing records eligible for inversion by the attribute are not included.
  This was not changed for 3.1, since it is now straightforward for the caller to update all records by passing over the entire set of records and replacing each

one with itself. This gives the caller the opportunity to perform one pass over the records to update several new attributes at once.

- IXAttributeQuery implements a subset of the IndexingKit query language. In particular, only the search operators can bind attribute values.
  This is still true for 3.1.