ReleaseNote50

```
20 February 2015, Apache Lucene™ 5.0.0 available

The Lucene PMC is pleased to announce the release of Apache Lucene 5.0.

Apache Lucene is a high-performance, full-featured text search engine
library written entirely in Java. It is a technology suitable for nearly
any application that requires full-text search, especially cross-platform.

This release contains numerous bug fixes, optimizations, and
improvements, some of which are highlighted below. The release
is available for immediate download at:
  http://lucene.apache.org/core/mirrors-core-latest-redir.html

See the CHANGES.txt file included with the release for a full list of
details.

Lucene 5.0 Release Highlights

Stronger index safety
 * All file access now uses Java’s NIO.2 APIs which give Lucene stronger index safety in terms of better error
handling and safer commits.
 * Every Lucene segment now stores a unique id per-segment and per-commit to aid in accurate replication of
index files.
 * During merging, IndexWriter now always checks the incoming segments for corruption before merging. This can
mean, on upgrading to 5.0.0, that merging may uncover long-standing latent corruption in an older 4.x index.

Reduced heap usage
 * Lucene now supports random-writable and advance-able sparse bitsets (RoaringDocIdSet and SparseFixedBitSet),
so the heap required is in proportion to how many bits are set, not how many total documents exist in the index.
 * Heap usage during IndexWriter merging is also much lower with the new Lucene50Codec, since doc values and
norms for the segments being merged are no longer fully loaded into heap for all fields; now they are loaded
for the one field currently being merged, and then dropped.
 * The default norms format now uses sparse encoding when appropriate, so indices that enable norms for many
sparse fields will see a large reduction in required heap at search time.
 * 5.0 has a new API to print a tree structure showing a recursive breakdown of which parts are using how much
heap.

Other features
 * FieldCache is gone (moved to a dedicated UninvertingReader in the misc module). This means when you intend
to sort on a field, you should index that field using doc values, which is much faster and less heap consuming
than FieldCache.
 * Tokenizers and Analyzers no longer require Reader on init.
 * NormsFormat now gets its own dedicated NormsConsumer/Producer
 * SortedSetSortField, used to sort on a multi-valued field, is promoted from sandbox to Lucene's core.
 * PostingsFormat now uses a "pull" API when writing postings, just like doc values. This is powerful because
you can do things in your postings format that require making more than one pass through the postings such as
iterating over all postings for each term to decide which compression format it should use.
 * New NumberRangePrefixTreeStrategy & DateRangePrefixTree in the spatial module enables indexing and searching
of date ranges, particularly multi-valued ones.
 * A new ExitableDirectoryReader extends FilterDirectoryReader and enables exiting requests that take too long
to enumerate over terms.
 * Suggesters from multi-valued field can now be built as DocumentDictionary now enumerates each value
separately in a multi-valued field.
 * ConcurrentMergeScheduler detects whether the index is on SSD or not and does a better job defaulting its
settings. This only works on Linux for now; other OS's will continue to use the previous defaults (tuned for
spinning disks).
 * Auto-IO-throttling has been added to ConcurrentMergeScheduler, to rate limit IO writes for each merge
depending on incoming merge rate.
 * CustomAnalyzer has been added that allows to configure analyzers like you do in Solr's index schema. This
class has a builder API to configure Tokenizers, TokenFilters, and CharFilters based on their SPI names and
parameters as documented by the corresponding factories.
 * Memory index now supports payloads.
 * Added a filter cache with a usage tracking policy that caches filters based on frequency of use.
 * The default codec has an option to control BEST_SPEED or BEST_COMPRESSION for stored fields.
 * Stored fields are merged more efficiently, especially when upgrading from previous versions or using
SortingMergePolicy
```

NOTE: Lucene 5 no longer supports the Lucene 3.x index format. Opening indexes will result in IndexFormatTooOldException. It is recommended to either reindex all your data, or upgrade the old indexes with the IndexUpgrader tool of latest Lucene 4 version (4.10.x). Those indexes can then be read with Lucene 5.

To read more about the changes, also see: http://blog.mikemccandless.com/2014/11/apache-lucene-500-is-coming.html

Please read CHANGES.txt (https://lucene.apache.org/core/5_0_0/changes/Changes.html) and MIGRATE.txt for a full list of new features and notes on upgrading.

Please report any feedback to the mailing lists (http://lucene.apache.org/core/discussion.html)