Timing of the Delete Operation When you build a TTL index in the *background* (page 508), the TTL thread can begin deleting documents while the index is building. If you build a TTL index in the foreground, MongoDB begins removing expired documents as soon as the index finishes building.

The TTL index does not guarantee that expired data will be deleted immediately upon expiration. There may be a delay between the time a document expires and the time that MongoDB removes the document from the database.

The background task that removes expired documents runs *every 60 seconds*. As a result, documents may remain in a collection during the period between the expiration of the document and the running of the background task.

Because the duration of the removal operation depends on the workload of your `mongod` instance, expired data may exist for some time *beyond* the 60 second period between runs of the background task.

Replica Sets On *replica sets*, the TTL background thread *only* deletes documents on the *primary*. However, the TTL background thread does run on secondaries. *Secondary* members replicate deletion operations from the primary.

Support for Queries A TTL index supports queries in the same way non-TTL indexes do.

Record Allocation A collection with a TTL index has `usePowerOf2Sizes` enabled, and you cannot modify this setting for the collection. As a result of enabling `usePowerOf2Sizes`, MongoDB must allocate more disk space relative to data size. This approach helps mitigate the possibility of storage fragmentation caused by frequent delete operations and leads to more predictable storage use patterns.

Restrictions

- TTL indexes are a single-field indexes. *Compound indexes* (page 489) do not support TTL and ignores the `expireAfterSeconds` option.
- The `_id` field does not support TTL indexes.
- You cannot create a TTL index on a *capped collection* (page 209) because MongoDB cannot remove documents from a capped collection.
- You cannot use `createIndex()` to change the value of `expireAfterSeconds` of an existing index. Instead use the `collMod` database command in conjunction with the `index` collection flag. Otherwise, to change the value of the option of an existing index, you must drop the index first and recreate.
- If a non-TTL single-field index already exists for a field, you cannot create a TTL index on the same field since you cannot create indexes that have the same key specification and differ only by the options. To change a non-TTL single-field index to a TTL index, you must drop the index first and recreate with the `expireAfterSeconds` option.

Additional Information

For examples, see *Expire Data from Collections by Setting TTL* (page 211).

Unique Indexes

A unique index causes MongoDB to reject all documents that contain a duplicate value for the indexed field.

To create a unique index, use the `db.collection.createIndex()` method with the `unique` option set to `true`. For example, to create a unique index on the `user_id` field of the `members` collection, use the following operation in the `mongo` shell: