# Expire data in DocumentDB collections automatically with time to live

Applications can produce and store vast amounts of data. Some of this data, like machine generated event data, logs, and user session information is only useful for a finite period of time. Once the data becomes surplus to the needs of the application it is safe to purge this data and reduce the storage needs of an application.

With “time to live” or TTL, Microsoft Azure DocumentDB provides the ability to have documents automatically purged from the system after a period of time. The default time to live can be set at the collection level, and overridden on a per-document basis. Once TTL is set, either as a collection default or at a document level, DocumentDB will automatically remove documents that exist after that period of time, in seconds, since they were last modified.

Time to live in DocumentDB uses an offset against when the document was last modified. To do this it uses the \_ts field which exists on every document. The \_ts field is a unix-style epoch timestamp representing the date and time. The \_ts field is updated every time a document is modified.

## TTL behavior

The TTL feature is controlled by TTL properties at two levels - the collection level and the document level. The values are set in seconds and are treated as a delta from the \_ts that the document was last modified at.

1. DefaultTTL for the collection

* If missing (or set to null), documents are not deleted automatically.
* If present and the value is “-1” = infinite – documents don’t expire by default
* If present and the value is some number (“n”) – documents expire “n” seconds after last modification

1. TTL for the documents:

* Property is applicable only if DefaultTTL is present for the parent collection.
* Overrides the DefaultTTL value for the parent collection.

As soon as the document has expired (ttl + \_ts >= current server time), the document is marked as “expired”. No operation will be allowed on these documents after this time and they will be excluded from the results of any queries performed. The documents are physically deleted in the system, and are deleted in the background opportunistically at a later time. This does not consume any [Request Units (RUs)](documentdb-request-units.md) from the collection budget.

The above logic can be shown in the following matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | DefaultTTL missing/not set on the collection | DefaultTTL = -1 on collection | DefaultTTL = “n” on collection |
| TTL Missing on document | Nothing to override at document level since both the document and collection have no concept of TTL. | No documents in this collection will expire. | The documents in this collection will expire when interval n elapses. |
| TTL = -1 on document | Nothing to override at the document level since the collection doesn’t define the DefaultTTL property that a document can override. TTL on a document is un-interpreted by the system. | No documents in this collection will expire. | The document with TTL=-1 in this collection will never expire. All other documents will expire after “n” interval. |
| TTL = n on document | Nothing to override at the document level. TTL on a document in un-interpreted by the system. | The document with TTL = n will expire after interval n, in seconds. Other documents will inherit interval of -1 and never expire. | The document with TTL = n will expire after interval n, in seconds. Other documents will inherit “n” interval from the collection. |

## Configuring TTL

By default, time to live is disabled by default in all DocumentDB collections and on all documents.

## Enabling TTL

To enable TTL on a collection, or the documents within a collection, you need to set the DefaultTTL property of a collection to either -1 or a non-zero positive number. Setting the DefaultTTL to -1 means that by default all documents in the collection will live forever but the DocumentDB service should monitor this collection for documents that have overridden this default.

## Configuring default TTL on a collection

You are able to configure a default time to live at a collection level.

To set the TTL on a collection, you need to provide a non-zero positive number that indicates the period, in seconds, to expire all documents in the collection after the last modified timestamp of the document (\_ts).

Or, you can set the default to -1, which implies that all documents inserted in to the collection will live indefinitely by default.

## Setting TTL on a document

In addition to setting a default TTL on a collection you can set specific TTL at a document level. Doing this will override the default of the collection.

To set the TTL on a document, you need to provide a non-zero positive number which indicates the period, in seconds, to expire the document after the last modified timestamp of the document (\_ts).

To set this expiry offset, set the TTL field on the document.

If a document has no TTL field, then the default of the collection will apply.

If TTL is disabled at the collection level, the TTL field on the document will be ignored until TTL is enabled again on the collection.

## Extending TTL on an existing document

You can reset the TTL on a document by doing any write operation on the document. Doing this will set the \_ts to the current time, and the countdown to the document expiry, as set by the ttl, will begin again.

If you wish to change the ttl of a document, you can update the field as you can do with any other settable field.

## Removing TTL from a document

If a TTL has been set on a document and you no longer want that document to expire, then you can retrieve the document, remove the TTL field and replace the document on the server.

When the TTL field is removed from the document, the default of the collection will be applied.

To stop a document from expiring and not inherit from the collection then you need to set the TTL value to -1.

## Disabling TTL

To disable TTL entirely on a collection and stop the background process from looking for expired documents the DefaultTTL property on the collection should be deleted.

Deleting this property is different from setting it to -1. Setting to -1 means new documents added to the collection will live forever but you can override this on specific documents in the collection.

Removing this property entirely from the collection means that no documents will expire, even if there are documents that have explicitly overridden a previous default.

## FAQ

**What will TTL cost me?**

There is no additional cost to setting a TTL on a document.

**How long will it take to delete my document once the TTL is up?**

The documents are marked as unavailable as soon as the document has expired (ttl + \_ts >= current server time). No operation will be allowed on these documents after this time and they will be excluded from the results of any queries performed. The documents are physically deleted by the system in the background. This will not consume any RUs from the collection’s budget.

**If it takes a period of time to delete the documents, will they count toward my quota (and bill) until they get deleted?**

No, once the document has expired you will not be billed for the storage of these documents and the size of the documents will not count toward the storage quota for a collection.

**Will TTL on a document have any impact on RU charges?**

No, there will be no impact on RU charges for operations performed on any document within DocumentDB.

**Will the deleting of documents impact on the throughput I have provisioned on my collection?**

No, serving requests against your collection will receive priority over running the background process to delete your documents. Adding TTL to any document will not impact this.

**When a document expires, how long will it remain in my collection until it’s deleted?**

As soon as the document expires it will no longer be accessible. The exact time a document will remain in your collection before actually being deleted is non-deterministic and will be based on when the background process is able to delete the document.

**Are expired documents deleted across all nodes, or is it “eventually consistent?”**

The document will become unavailable at the same time across all nodes and in all regions.

**Is there an RU cost for TTL-monitoring background tasks?**

No, there is no RU cost for this.

**How often are TTL expirations checked?**

Checking TTL expirations doesn’t happen as a background process. When responding to a request the backend service will do the checks inline and exclude any documents that have expired. The deletion of the physical document is the only process that is asynchronously run in the background. The frequency of this process is determined by the available RUs on the collection.

**Does the TTL feature only apply to entire documents, or can I expire individual document property values?**

TTL applies to the entire document. If you would like to expire just a portion of a document, then it is recommended that you extract the portion from the main document in to a separate “linked” document and then use TTL on that extracted document.

**Does the TTL feature have any specific indexing requirements?**

Yes. The collection must have [indexing policy set](documentdb-indexing-policies.md) to either Lazy or Consistent. Trying to set DefaultTTL on a collection with indexing set to None will result in an error, as will trying to turn off indexing on a collection that has a DefaultTTL already set.

## Next steps

To learn more about Azure DocumentDB, refer to the service [*documentation*](https://azure.microsoft.com/documentation/services/documentdb/) page.