# Using Time Machine with MongoDB

Time Machine® provides software virtual clocks that enable you to time travel your applications into the future or the past, facilitating time shift testing on your date and time sensitive application logic, such as month end, quarter end, year end processing, billing cycle, work flow, regulatory go live, and policy life cycle. Time Machine has a long history testing applications deployed to relational databases. This document describes using Time Machine with the NoSQL engine MongoDB.

## ENVIRONMENT

Windows Server 2012 R2 64 bit

Time Machine for Windows 2012 version 11.1R7
MongoDB version 3.2 x86 64 bit single node deployment instance running under the local Windows user *mongouser*.

The following parameters were used to startup single MongoDB instance:

```
C:\MongoDB3.2\bin\mongod.exe --port 27017 --noauth  --storageEngine wiredTiger --dbpath C:\MongoDB3.2\db --directoryperdb –journal
```

## TEST CASE: VIRTUAL CLOCK SET FOR MONGODB INSTANCE PROCESS

1. Create virtual clock for mongod.exe instance process (8964 is PID of mongod.exe process):

```
C:\MongoDB3.2\bin> tmuser -a -p 8964 -y 10

C:\MongoDB3.2\bin> tmuser –l

Listing all virtual clocks:

ACCOUNT/pid/tid                    CLOCK INFORMATION
--------------------------------------------------------------------------------
 (pid)8964                         Wed Sep 30 2026 10:32:26

                                   Clock runs in normal speed.
--------------------------------------------------------------------------------
```

2. Connect to MongoDB:

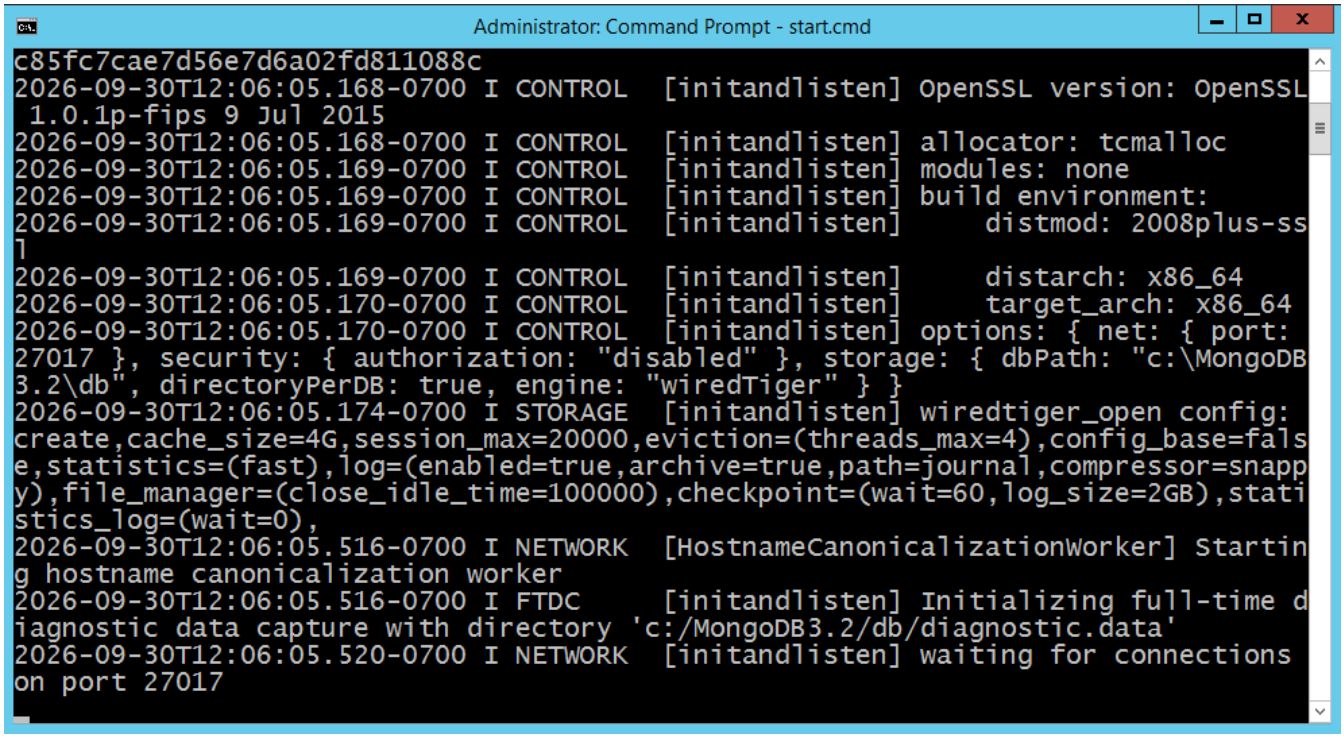
```
C:\MongoDB3.2\bin>mongo.exe

connecting to: test

> db
Test
```

The MongoDB log file shows time in the future:

**2026-09-30T10:33:30.435** I NETWORK [initandlisten] connection accepted from 127.0.0.1:58592 #3 (1 connection now open)




3. Insert a document into a MongoDB collection and calling $currentDate system function:

```
> db.col_1.update({ "name": "bar_future"},  { "$currentDate": { "date": { "$type": "date" }}},  { upsert: true } )

WriteResult({
    "nMatched" : 0,
    "nUpserted" : 1,
    "nModified" : 0,
    "_id" : ObjectId("6abcc9b75137ec8682c0bc50")
})
```

4. Retrieve the inserted document:

```
> db.col_1.find()

{ "_id" : ObjectId("6abd6564dfab4d8ac76f2b53"), "name" : "bar_future", "date" :
ISODate("2026-09-30T19:39:16.682Z") }
```

```
Administrator: Command Prompt - mongo.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd c:\MongoDB3.2\bin

c:\MongoDB3.2\bin>mongo.exe
MongoDB shell version: 3.2.9
connecting to: test
>
> db.col_1.update({ "name": "bar_future"},  { "$currentDate": { "date": { "$type
": "date" }}},  { upsert: true } )
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("6abd7098917a9930dab5b36d")
})
> db.col_1.find()
{ "_id" : ObjectId("6abd7098917a9930dab5b36d"), "name" : "bar_future", "date" :
ISODate("2026-09-30T20:27:04.696Z") }
>
```

**Test Result**: If the virtual clock is created for the MongoDB instance process mongod.exe then system function $currentDate returns correct virtual time.

## TEST CASE: VIRTUAL CLOCK FOR CLIENT CONNECTION WINDOWS THREAD

1. Create virtual clock for client connection Windows thread (6784 is the thread number of the database session):

C:\MongoDB3.2>tmuser -a -t 6784 -y 10

C:\MongoDB3.2>tmuser –l

Listing all virtual clocks:

ACCOUNT/pid/tid CLOCK INFORMATION

---

**(tid)6784** **Wed Sep 30 2026 10:43:58**

Clock runs in normal speed.

---

2. Insert a document into a MongoDB collection and calling $currentDate system function:

```
> db.col_1.update({ "name": "bar_future"},
...     { "$currentDate": { "date": { "$type": "date" }}},
...     { upsert: true }
... )

WriteResult({
      "nMatched" : 0,
      "nUpserted" : 1,
      "nModified" : 0,
      "_id" : ObjectId("6abcccbb5137ec8682c0bc51")
})
```

3. Retrieve the inserted document:

```
> db.col_1.find()

{ "_id" : ObjectId("6abcccbb5137ec8682c0bc51"), "name" : "bar_future", "date" :
ISODate("2026-09-30T19:45:31.487Z") }
```

**Test Result**: If the virtual clock is created for a client connection Windows thread then the system function $currentDate returns the correct virtual time.

## TEST CASE: VIRTUAL CLOCK FOR OS USER RUNNING MONGODB INSTANCE

1. Create a virtual clock for OS user *mongouser* under which the MongoDB instance is running:

```
C:\MongoDB3.2> tmuser -a -u mongouser -y 10

C:\MongoDB3.2\bin> tmuser –l

Listing all virtual clocks:

ACCOUNT/pid/tid                    CLOCK INFORMATION
--------------------------------------------------------------------------------

(U) WIN2012--ORA\mongouser         Wed Sep 30 2026 12:02:49

                                   Clock runs in normal speed.
--------------------------------------------------------------------------------
```

2. Insert a document into a MongoDB collection and calling $currentDate system function:

```
> db.col_1.update({ "name": "bar_future"},     { "$currentDate": { "date": { "$type": "date" }}},     { upsert: true } )

WriteResult({
      "nMatched" : 0,
      "nUpserted" : 1,
      "nModified" : 0,
      "_id" : ObjectId("679e3887d0d84fce09745be3")
```

3. Retrieve the inserted document:

```
> db.col_1.find()

{ "_id" : ObjectId("6abd6564dfab4d8ac76f2b53"), "name" : "bar_future", "date" :
ISODate("2026-09-30T19:39:16.682Z") }
```

**Test Result**: If the virtual clock created for the OS user under which the MongoDB instance is running then the system function $currentDate returns the correct virtual time.

## SUMMARY

Time Machine supports time shift testing applications using MongoDB 3.2 and successfully time travels the following components:

- Entire MongoDB instance;
- A particular client connection Windows thread;
- An OS user under which the MongoDB Instance is running.

In all the above cases, the MongoDB uses virtual time for calls of $currentDate system function which returns system time or timestamp at the server side.

Based on the above results, any complex enterprise application that uses MongoDB can be successfully time traveled if synched virtual clocks are created for both parts of application: MongoDB database instance or session connection thread together with the application components that may be written in Java or C#. This is because date/time functions can be called from the database or a function call in the programming language. For this reason, it would be useful to use the Time Machine Sync Server to time travel all the nodes in the application deployment architecture. Read more at: http://www.solution-soft.com/products/time-machine/tmsync

It is also possible to use Time Machine for MongoDB cluster deployments, which consist of multiple MongoDB nodes. In this case, Time Machine Sync Server is also used to time travel the MongoDB cluster and application components in a consistent way.

# Using Time Machine with MongoDB



SolutionSoft Systems, Inc.
2350 Mission College Blvd.
Suite #777
Santa Clara, CA 95054
U.S.A.

Worldwide Inquiries:

Toll Free: 1.888.884.7337
Phone: 1.408.346.1400
Fax: 1.408.346.1499

www.solution-soft.com



WP106_20160930_v.1_6