



Ubiquiti Support and Help Center > UniFi Network Archived Documentation > UniFi Network Troubleshooting
> Troubleshooting: Application

UniFi - How to Remove (Prune) Older Data and Adjust Mongo Database Size

A UniFi Network application managing many devices or logging many clients may become very large over time if left unchecked. The following article explains how to use the prune script in order to remove statistics that are older than X number of days, including alarms, events, guest entries, detected rogue APs, known clients, expired vouchers, and traffic statistics. The prune script will not remove data that is essential for the correct functionality of the UniFi Network applicationo, such as users that have been blocked.

NOTES & REQUIREMENTS:

The UniFi Network application needs to be running prior to executing the script. If it is not running, please start the application before performing the steps outlined in this article. Always remember to [create a backup](#) of your Network application prior to performing any major changes, such as this one.

Table of Contents

- [Introduction](#)
- [How to Prune a UniFi Network Application Hosted on Windows](#)
- [How to Prune a UniFi Network Application Hosted on macOS](#)
- [How to Prune a UniFi Network Application Hosted on Linux \(Ubuntu, Debian, and Cloud Key\)](#)

- How to Prune a UniFi Network Application Hosted on UniFi OS (UDM-Pro)
 - Pruning Script
-

Introduction

UniFi uses MongoDB to store relevant information about connected devices, application configuration, clients, and statistics. The UniFi Network application offers the option to compact the database from within the application itself. Assuming you are able to log into the UniFi Network application, please use the compact database button under the **Services** section in **Settings > Maintenance** instead of following the manual pruning instructions in this article.

The compact database option will not remove data from the database, only shrink the dataSize of what is currently in the database. Compacting the database involves defragmentation, but not error correcting. If there is suspected invalid or corrupt entries in your database, please follow the instructions below.

How to Prune a UniFi Network Application Hosted on Windows

1. Download mongo. The Windows UniFi installer does not include the mongo binary. Visit the [MongoDB official download website](#), and download the .zip release that corresponds to your OS and the MongoDB that matches the one your UniFi Network application version is using, as seen in the MongoDB log, or stated in the application release notes on the Community.

2. Extract mongo. Extract \bin\mongo.exe to a working directory of your choice. In this example, we C:\prune\ is used. You may ignore all other files included in the package.

3. Download Script. Download the script to the server by [clicking here](#), and save it to your working directory.

4. Open Command Prompt. Open the command prompt by pressing WINDOWS + R. In the popup, type cmd, and press ENTER.

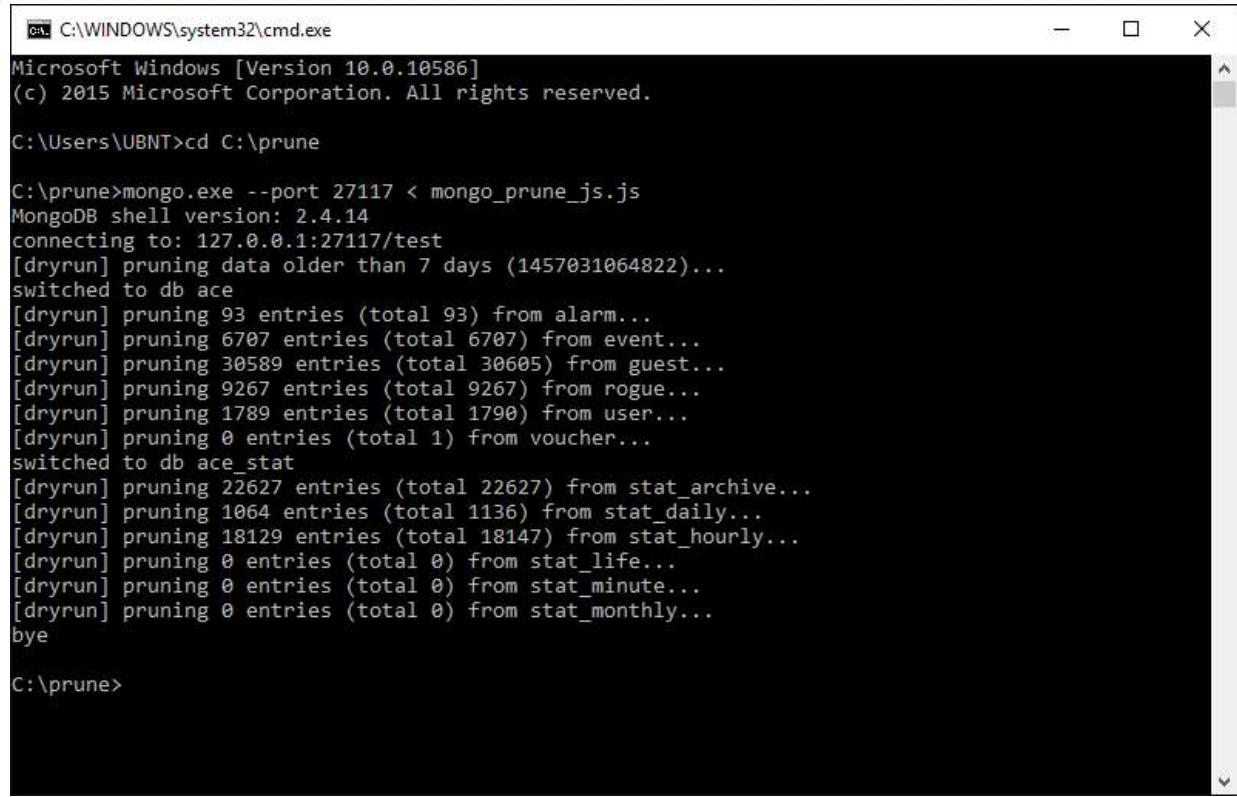
5. Enter the Working Directory. In the command prompt, change to your working directory by typing in the following command and hitting enter:

```
cd C:\prune\
```

6. Perform Test Run. By default, the script is in "dry run" mode and will indicate what will be pruned from the database without actually doing it. This step is to make sure the script runs as expected. Type in the following command and hit enter:

```
mongo.exe --port 27117 < mongo_prune_js.js
```

The output should look similar to this:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

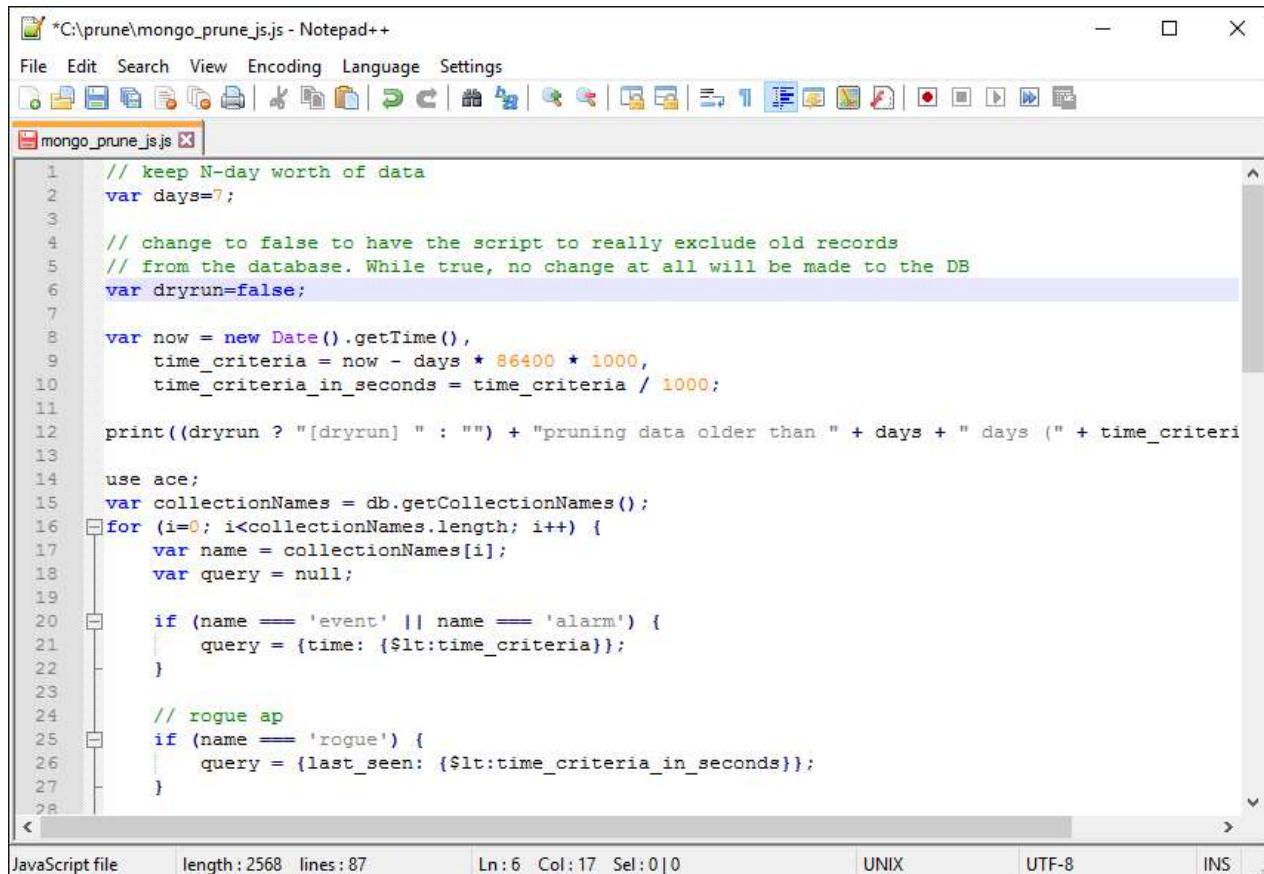
C:\Users\UBNT>cd C:\prune

C:\prune>mongo.exe --port 27117 < mongo_prune_js.js
MongoDB shell version: 2.4.14
connecting to: 127.0.0.1:27117/test
[dryrun] pruning data older than 7 days (1457031064822)...
switched to db ace
[dryrun] pruning 93 entries (total 93) from alarm...
[dryrun] pruning 6707 entries (total 6707) from event...
[dryrun] pruning 30589 entries (total 30605) from guest...
[dryrun] pruning 9267 entries (total 9267) from rogue...
[dryrun] pruning 1789 entries (total 1790) from user...
[dryrun] pruning 0 entries (total 1) from voucher...
switched to db ace_stat
[dryrun] pruning 22627 entries (total 22627) from stat_archive...
[dryrun] pruning 1064 entries (total 1136) from stat_daily...
[dryrun] pruning 18129 entries (total 18147) from stat_hourly...
[dryrun] pruning 0 entries (total 0) from stat_life...
[dryrun] pruning 0 entries (total 0) from stat_minute...
[dryrun] pruning 0 entries (total 0) from stat_monthly...
bye

C:\prune>
```

7. Disable "dry run". Edit the script to disable dry run and configure the number of days worth of data to keep. By default, the script will keep 7 days worth of data. Use [Notepad++](#) or a similar plain-text editor to modify the script and save it. Do *not* use a rich text editor such as Wordpad or Word that can add hidden characters and won't respect line breaks.

Edit `var days=7;` to change how many days worth of data to keep, and change `var dryrun=true;` to `var dryrun=false;` to allow the script to actually prune the database rather than just do a test run.



The screenshot shows a Notepad++ window with the file `C:\prune\mongo_prune_js.js` open. The code is a MongoDB shell script for pruning old data from a database. It includes variables for keeping 7 days of data and setting `dryrun` to false for actual pruning. The script iterates through collection names, defining queries based on collection names like 'event' or 'alarm'. It also handles a 'rogue' collection by querying for documents last seen before the time criteria.

```
// keep N-day worth of data
var days=7;

// change to false to have the script to really exclude old records
// from the database. While true, no change at all will be made to the DB
var dryrun=false;

var now = new Date().getTime(),
    time_criteria = now - days * 86400 * 1000,
    time_criteria_in_seconds = time_criteria / 1000;

print((dryrun ? "[dryrun] " : "") + "pruning data older than " + days + " days (" + time_criteria_in_seconds + ")");

use ace;
var collectionNames = db.getCollectionNames();
for (i=0; i<collectionNames.length; i++) {
    var name = collectionNames[i];
    var query = null;

    if (name == 'event' || name == 'alarm') {
        query = {time: {$lt:time_criteria}};
    }

    // rogue ap
    if (name == 'rogue') {
        query = {last_seen: {$lt:time_criteria_in_seconds}};
    }
}
```

8. Prune the Database. Once changes have been made and saved, run the modified script, actually purging the database this time:

```
mongo.exe --port 27117 < mongo_prune_js.js
```

NOTE: As the database is actually being modified, this step may take considerably longer compared to the test (dry) run. Do not interrupt the process until you receive a message that says **bye**.

9. Verify Completion. Verify that the operation completed successfully and that no errors were reported. The output should be similar to this:

```
C:\WINDOWS\system32\cmd.exe
[dryrun] pruning 18129 entries (total 18147) from stat_hourly...
[dryrun] pruning 0 entries (total 0) from stat_life...
[dryrun] pruning 0 entries (total 0) from stat_minute...
[dryrun] pruning 0 entries (total 0) from stat_monthly...
bye

C:\prune>mongo.exe --port 27117 < mongo_prune_js.js
MongoDB shell version: 2.4.14
connecting to: 127.0.0.1:27117/test
pruning data older than 7 days (1457031303046)...
switched to db ace
pruning 93 entries (total 93) from alarm...
pruning 6707 entries (total 6707) from event...
pruning 30589 entries (total 30605) from guest...
pruning 9267 entries (total 9267) from rogue...
pruning 1789 entries (total 1790) from user...
pruning 0 entries (total 1) from voucher...
{ "ok" : 1 }
switched to db ace_stat
pruning 22627 entries (total 22627) from stat_archive...
pruning 1064 entries (total 1136) from stat_daily...
pruning 18129 entries (total 18147) from stat_hourly...
pruning 0 entries (total 0) from stat_life...
pruning 0 entries (total 0) from stat_minute...
pruning 0 entries (total 0) from stat_monthly...
{ "ok" : 1 }
{ "ok" : 1 }
bye

C:\prune>
```

10. **OPTIONAL** **Cleanup.** Delete the directory you created and the files inside if you don't intend to use them again.

How to Prune a UniFi Network Application Hosted on macOS

1. Create a Working Directory. Create a working directory on the computer. For the purpose of this article, we will be creating a directory named /prune.

2. Download mongo. The macOS UniFi installer does not include the mongo binary. Visit the [MongoDB official download website](#), and download the .tgz release that corresponds to your server's CPU architecture. The correct MongoDB version will appear in the release notes of the UniFi Network application found in our [Community](#).

3. Extract mongo. Move the downloaded package to your working directory, and extract it by double-clicking it or using the application of your choice.

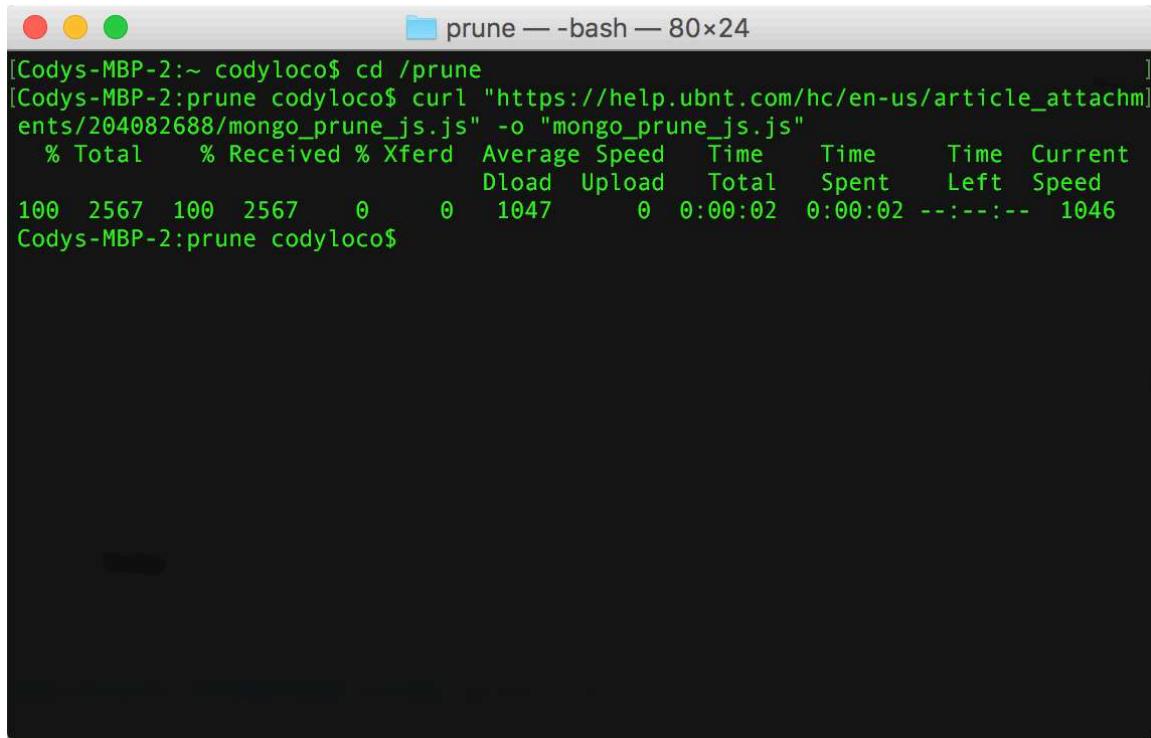
4. Move mongo to Working Directory. Locate the bin/mongo binary file, and copy it to your working directory. At this point, the downloaded .tgz package and any extra extracted files can be deleted as we only need the mongo binary.

5. Enter the Working Directory. Open a Terminal window, and change directories to the working directory created in Step 1 with the following command:

```
cd /prune
```

6. Download Script. Download the pruning script with the following command:

```
curl -kO "https://help.ui.com/hc/article_attachments/115024095828/mongo_prune_js.js" -o "mongo
```



The screenshot shows a terminal window with the title bar 'prune — bash — 80x24'. The window contains the following text:

```
[Cody's-MBP-2:~ codyloco$ cd /prune
[Cody's-MBP-2:prune codyloco$ curl "https://help.ubnt.com/hc/en-us/article_attachments/204082688/mongo_prune_js.js" -o "mongo_prune_js.js"
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total   Spent   Left  Speed
100  2567  100  2567    0     0  1047      0  0:00:02  0:00:02  --:--:--  1046
Cody's-MBP-2:prune codyloco$
```

7. Perform Test Run. Perform a test run of the script. By default, the script is in "dry run" mode and will indicate what will be pruned from the database without actually doing it. This step ensures the script runs as expected:

```
./mongo --port 27117 < mongo_prune_js.js
```

The output should look similar to this:

```
prune — bash — 80x24
ents/204082688/mongo_prune_js.js" -o "mongo_prune_js.js"
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100  2567  100  2567    0     0    772      0  0:00:03  0:00:03 --:--:--   772
[Codys-MBP-2:prune codyloco$ ./mongo --port 27117 < mongo_prune_js.js]
MongoDB shell version: 2.4.14
connecting to: 127.0.0.1:27117/test
[dryrun] pruning data older than 7 days (1457022364812)...
switched to db ace
[dryrun] pruning 93 entries (total 93) from alarm...
[dryrun] pruning 7099 entries (total 7099) from event...
[dryrun] pruning 30586 entries (total 30605) from guest...
[dryrun] pruning 9267 entries (total 9267) from rogue...
[dryrun] pruning 1789 entries (total 1790) from user...
[dryrun] pruning 0 entries (total 1) from voucher...
switched to db ace_stat
[dryrun] pruning 22627 entries (total 22627) from stat_archive...
[dryrun] pruning 1064 entries (total 1101) from stat_daily...
[dryrun] pruning 18129 entries (total 18149) from stat_hourly...
[dryrun] pruning 0 entries (total 0) from stat_life...
[dryrun] pruning 0 entries (total 0) from stat_minute...
[dryrun] pruning 0 entries (total 0) from stat_monthly...
bye
Codys-MBP-2:prune codyloco$ ]
```

8. Disable "dry run". If all seems correct, edit the script to disable dry run and configure the number of days worth of data to keep. By default, the script will keep 7 days worth of data. Use nano or a similar editor to modify the script, as such:

```
nano mongo_prune_js.js
```

Assuming you are using nano to edit the file, use the arrow keys to navigate the text. Edit `var days=7;` to change how many days worth of data to keep, and change `vardryrun=true;` to `var dryrun=false;` to allow the script to actually prune the database rather than just do a test run. When ready, press CTRL + O to save the file (pressing Enter to confirm) and CTRL + X to exit.

```
// keep N-day worth of data
var days=7;

// change to false to have the script to really exclude old records
// from the database. While true, no change at all will be made to the DB
var dryrun=false;

var now = new Date().getTime(),
    time_criteria = now - days * 86400 * 1000,
    time_criteria_in_seconds = time_criteria / 1000;

print((dryrun ? "[dryrun] " : "") + "pruning data older than " + days + " days $"

use ace;
var collectionNames = db.getCollectionNames();
for (i=0; i<collectionNames.length; i++) {
    var name = collectionNames[i];
    var query = null;

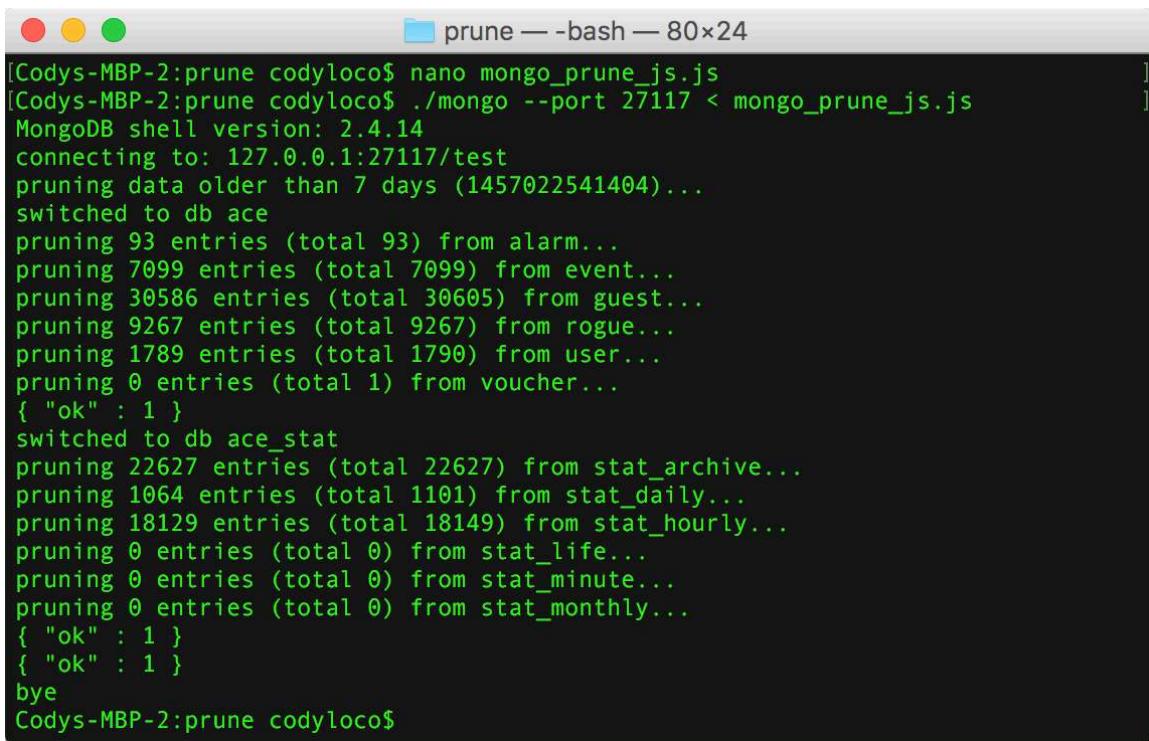
File Name to Write: mongo_prune_js.js
^G Get Help      ^T To Files      M-M Mac Format      M-P Prepend
^C Cancel        M-D DOS Format    M-A Append       M-B Backup File
```

9. Prune the Database. Run the modified script, purging the database this time:

```
./mongo --port 27117 < mongo_prune_js.js
```

NOTE: As the database is actually being modified, this step may take considerably longer compared to the test (dry) run. Do not interrupt the process until you receive a message that says **bye**.

10. Verify Completion. Verify that the operation completed successfully and that no errors were reported. The output should be similar to this:



A screenshot of a macOS Terminal window titled "prune — bash — 80x24". The window shows the execution of a MongoDB script named "mongo_prune_js.js" to remove old data from the "test" database. The output includes statistics for various collections like "alarm", "event", "guest", "rogue", "user", and "voucher", and then moves to the "ace_stat" collection to prune "stat_archive", "stat_daily", "stat_hourly", "stat_life", "stat_minute", and "stat_monthly" collections. It ends with a "bye" command and exits the session.

```
[Cody's-MBP-2:prune codyloco$ nano mongo_prune_js.js
[Cody's-MBP-2:prune codyloco$ ./mongo --port 27117 < mongo_prune_js.js
MongoDB shell version: 2.4.14
connecting to: 127.0.0.1:27117/test
pruning data older than 7 days (1457022541404)...
switched to db ace
pruning 93 entries (total 93) from alarm...
pruning 7099 entries (total 7099) from event...
pruning 30586 entries (total 30605) from guest...
pruning 9267 entries (total 9267) from rogue...
pruning 1789 entries (total 1790) from user...
pruning 0 entries (total 1) from voucher...
{ "ok" : 1 }
switched to db ace_stat
pruning 22627 entries (total 22627) from stat_archive...
pruning 1064 entries (total 1101) from stat_daily...
pruning 18129 entries (total 18149) from stat_hourly...
pruning 0 entries (total 0) from stat_life...
pruning 0 entries (total 0) from stat_minute...
pruning 0 entries (total 0) from stat_monthly...
{ "ok" : 1 }
{ "ok" : 1 }
bye
Cody's-MBP-2:prune codyloco$
```

11. End Session. End the Terminal session by typing the following command and hitting enter. Then close the Terminal window.

```
exit
```

12. OPTIONAL Cleanup. Remove the working directory if you don't intend to use it again.

User Tip: If you see an error that reads:

```
WARNING: shell and server versions do not match
```

Please download an older version of MongoDB. Instead of current version, try v3.4.x:

https://fastdl.mongodb.org/osx/mongodb-osx-ssl-x86_64-3.4.24.tgz

How to Prune a UniFi Network Application Hosted on Linux (Ubuntu, Debian, and Cloud Key)

The Network application must be running prior to executing the script. If it is not running, please start the application before performing the following steps issuing the following command after Step 2: Enter the Working Directory.

```
sudo service unifi start
```

1. Establish Connection. Connect to your server via SSH using your preferred client and authenticate. The screenshots in this section of the article are from PuTTY on Windows. If using a Linux or Mac client, you can connect to the server using the built-in Terminal application.

2. Enter the Working Directory. Change to your home directory, or create a working directory of your choice. For this article, the home directory will be used.

```
cd ~
```

3. Download Script. Download the pruning script to your server:

```
curl -k0 https://help.ui.com/hc/article_attachments/115024095828/mongo_prune_js.js
```

The screenshot shows a PuTTY terminal window titled "192.168.1.224 - PuTTY". The session path is "/". The URL "http://www.ubnt.com" is displayed at the top. The terminal output is as follows:

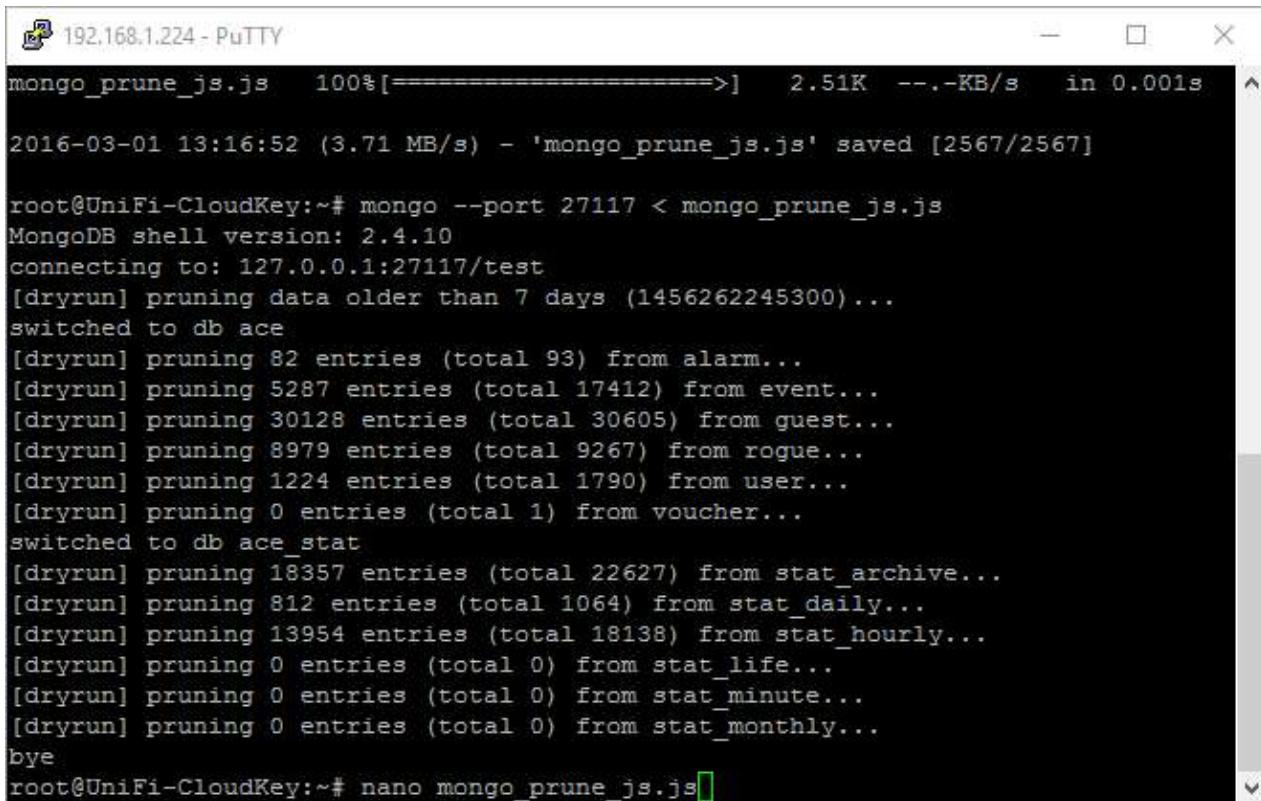
```
Welcome to UniFi CloudKey!
Last login: Tue Mar  1 13:04:02 2016 from 192.168.1.124
root@UniFi-CloudKey:~# cd ~
root@UniFi-CloudKey:~# wget https://help.ubnt.com/hc/en-us/article_attachments/204082688/mongo_prune_js.js
converted 'https://help.ubnt.com/hc/en-us/article_attachments/204082688/mongo_prune_js.js' (ANSI_X3.4-1968) -> 'https://help.ubnt.com/hc/en-us/article_attachments/204082688/mongo_prune_js.js' (UTF-8)
--2016-03-01 13:16:51-- https://help.ubnt.com/hc/en-us/article_attachments/204082688/mongo_prune_js.js
Resolving help.ubnt.com (help.ubnt.com) ... 104.218.207.196
Connecting to help.ubnt.com (help.ubnt.com)|104.218.207.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2567 (2.5K) [text/javascript]
Saving to: 'mongo_prune_js.js'

mongo_prune_js.js 100%[=====] 2.51K --.-KB/s in 0.001s
2016-03-01 13:16:52 (3.71 MB/s) - 'mongo_prune_js.js' saved [2567/2567]
root@UniFi-CloudKey:~# mongo --port 27117 < mongo_prune_js.js
```

4. Perform Test Run. Perform a test-run of the script. By default, the script is in "dry run" mode and will indicate what will be pruned from the database without actually doing it. This step verifies that the script runs as expected:

```
mongo --port 27117 < mongo_prune_js.js
```

The output should look similar to this:



A screenshot of a PuTTY terminal window titled "192.168.1.224 - PuTTY". The window shows the execution of a MongoDB script named "mongo_prune_js.js". The output indicates a dry run of the pruning process, showing the number of entries being pruned from various database collections. The script is then edited using the nano text editor.

```
mongo_prune_js.js 100%[=====] 2.51K ---KB/s in 0.001s
2016-03-01 13:16:52 (3.71 MB/s) - 'mongo_prune_js.js' saved [2567/2567]

root@UniFi-CloudKey:~# mongo --port 27117 < mongo_prune_js.js
MongoDB shell version: 2.4.10
connecting to: 127.0.0.1:27117/test
[dryrun] pruning data older than 7 days (1456262245300)...
switched to db ace
[dryrun] pruning 82 entries (total 93) from alarm...
[dryrun] pruning 5287 entries (total 17412) from event...
[dryrun] pruning 30128 entries (total 30605) from guest...
[dryrun] pruning 8979 entries (total 9267) from rogue...
[dryrun] pruning 1224 entries (total 1790) from user...
[dryrun] pruning 0 entries (total 1) from voucher...
switched to db ace_stat
[dryrun] pruning 18357 entries (total 22627) from stat_archive...
[dryrun] pruning 812 entries (total 1064) from stat_daily...
[dryrun] pruning 13954 entries (total 18138) from stat_hourly...
[dryrun] pruning 0 entries (total 0) from stat_life...
[dryrun] pruning 0 entries (total 0) from stat_minute...
[dryrun] pruning 0 entries (total 0) from stat_monthly...
bye
root@UniFi-CloudKey:~# nano mongo_prune_js.js
```

5. Disable "dry run". Edit the script to disable dry run and configure the number of days worth of data to keep. By default, the script will keep 7 days worth of data. Use nano or a similar editor to modify the script, as such:

```
nano mongo_prune_js.js
```

Assuming you are using nano to edit the file, use the arrow keys to navigate the text. Edit `var days=7;` to change how many days worth of data to keep, and change `var dryrun=true;` to `var dryrun=false;` to allow the script to actually prune the database rather than just do a test run. When ready, press CTRL + O to save the file (pressing Enter to confirm) and CTRL + X to exit.

```
// keep N-day worth of data
var days=7;

// change to false to have the script to really exclude old records
// from the database. While true, no change at all will be made to the DB
var dryrun=false;

var now = new Date().getTime(),
    time_criteria = now - days * 86400 * 1000,
    time_criteria_in_seconds = time_criteria / 1000;

print((dryrun ? "[dryrun] " : "") + "pruning data older than " + days + " days $"

use ace;
var collectionNames = db.getCollectionNames();
for (i=0; i<collectionNames.length; i++) {
    var name = collectionNames[i];
    var query = null;

        [ Read 86 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

NOTE: If you receive an error indicating nano is not installed, you can install it with the following command, then run the previous command again:

```
sudo apt-get update && sudo apt-get -y install nano
```

6. Prune Database.

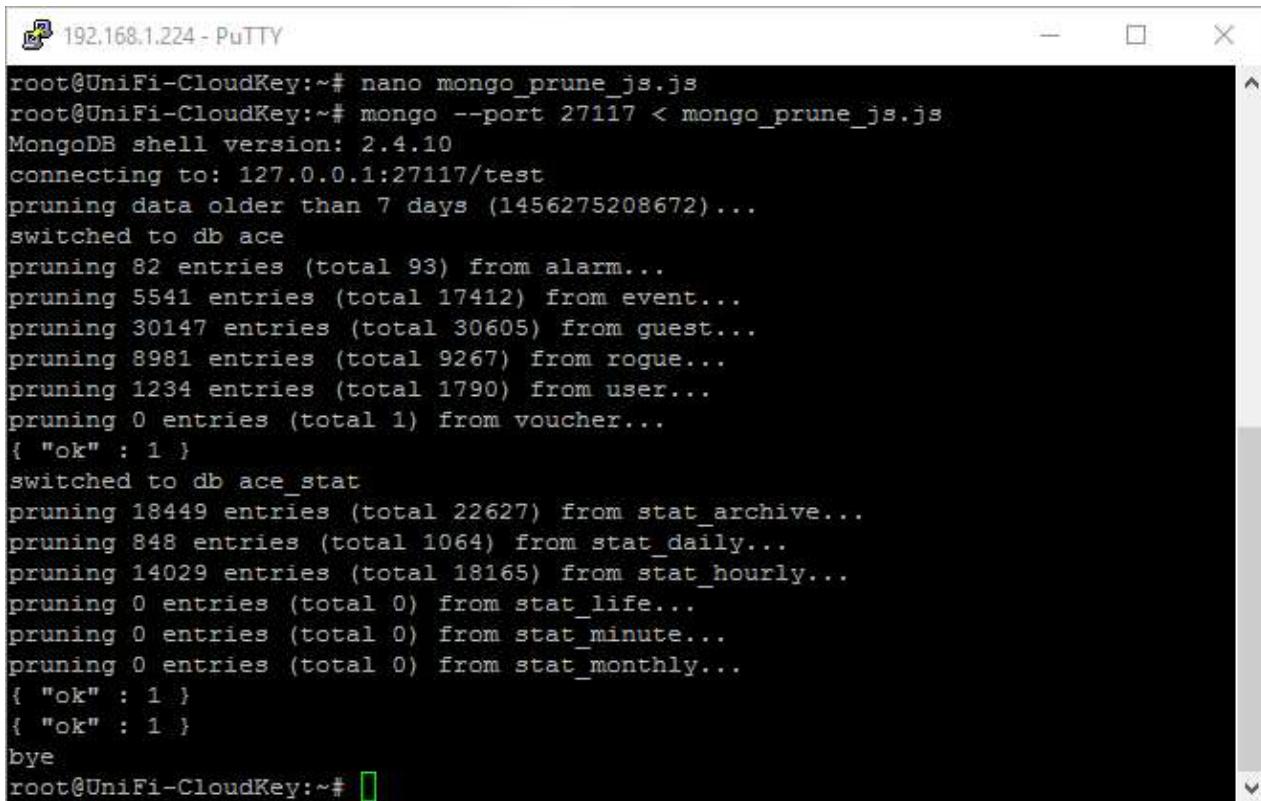
Run the modified script, actually purging the database this time:

```
mongo --port 27117 < mongo_prune_js.js
```

NOTE: As the database is actually being modified, this step may take considerably longer compared to the test (dry) run. Do not interrupt the process until you receive a message that says **bye**.

7. Verify Completion.

Verify that the operation completed successfully and that no errors were reported. The output should be similar to this:



A screenshot of a PuTTY terminal window titled "192.168.1.224 - PuTTY". The session is running as root on a UniFi CloudKey device. The user has run a script named "mongo_prune_js.js" which connects to the MongoDB database at port 27117. The script performs a series of pruning operations across various databases and collections, removing documents older than 7 days. It shows the number of entries pruned from "alarm", "event", "guest", "rogue", "user", and "voucher" collections in the "ace" database, as well as "stat_archive", "stat_daily", "stat_hourly", "stat_life", "stat_minute", and "stat_monthly" collections in the "ace_stat" database. The process concludes with a final "ok" response and a "bye" command.

```
root@UniFi-CloudKey:~# nano mongo_prune_js.js
root@UniFi-CloudKey:~# mongo --port 27117 < mongo_prune_js.js
MongoDB shell version: 2.4.10
connecting to: 127.0.0.1:27117/test
pruning data older than 7 days (1456275208672)...
switched to db ace
pruning 82 entries (total 93) from alarm...
pruning 5541 entries (total 17412) from event...
pruning 30147 entries (total 30605) from guest...
pruning 8981 entries (total 9267) from rogue...
pruning 1234 entries (total 1790) from user...
pruning 0 entries (total 1) from voucher...
{ "ok" : 1 }
switched to db ace_stat
pruning 18449 entries (total 22627) from stat_archive...
pruning 848 entries (total 1064) from stat_daily...
pruning 14029 entries (total 18165) from stat_hourly...
pruning 0 entries (total 0) from stat_life...
pruning 0 entries (total 0) from stat_minute...
pruning 0 entries (total 0) from stat_monthly...
{ "ok" : 1 }
{ "ok" : 1 }
bye
root@UniFi-CloudKey:~#
```

8. **OPTIONAL** **Cleanup.** Remove the script from your home directory, if you don't intend to use it again:

```
rm mongo_prune_js.js
```

9. **End Session.** End the terminal session:

```
exit
```

How to Prune a UniFi Network Application Hosted on UniFi OS (UDM-Pro)

1. Enter shell and download script:

```
unifi-os shell
curl https://help.ui.com/hc/article_attachments/115024095828/mongo_prune_js.js -o /tmp/mongo_p
```

2. [OPTIONAL] Define days to keep data. Issue the following commands to change the amount of days that data will be kept. The example below sets the 'days to keep data' to 14 days; the default value is 7. For example, by changing days_to_keep="14" to days_to_keep="21", you will be setting the 'days to keep data' to 21, and any data older than 21 days will be pruned.

```
current_days=$(awk '/var days/{print$2}' /tmp/mongo_prune_js.js | sed -e 's/days//g' -e 's/;/'
days_to_keep="14"
sed -i "s/days=${current_days}/days=${days_to_keep}/g" /tmp/mongo_prune_js.js
```

3. Perform a test run. To run a "dry run" issue the following commands:

```
sed -i 's/dryrun=false/dryrun=true/g' /tmp/mongo_prune_js.js
mongo --port 27117 < /tmp/mongo_prune_js.js
```

4. Run the script. This will run the script without "dry run".

```
sed -i 's/dryrun=true/dryrun=false/g' /tmp/mongo_prune_js.js
mongo --port 27117 < /tmp/mongo_prune_js.js
```

5. Remove script and exit shell:

```
rm /tmp/mongo_prune_js.js
exit
```

Prune Script for Reference

This is the pruning script for reference, or you can download it from the attachments at the base of the article:

```
// keep N-day worth of data
var days=7;

// change to false to have the script to really exclude old records
// from the database. While true, no change at all will be made to the DB
var dryrun=true;
```

```

var now = new Date().getTime(),
time_criteria = now - days * 86400 * 1000,
time_criteria_in_seconds = time_criteria / 1000;

print((dryrun ? "[dryrun] " : "") + "pruning data older than " + days + " days (" + time_crite

use ace;
var collectionNames = db.getCollectionNames();
for (i=0; i<collectionNames.length; i++) {
  var name = collectionNames[i];
  var query = null;

  if (name === 'event' || name === 'alarm') {
    query = {time: {$lt:time_criteria}};
  }

  // rogue ap
  if (name === 'rogue') {
    query = {last_seen: {$lt:time_criteria_in_seconds}};
  }

  // removes vouchers expired more than '$days' ago
  // active and unused vouchers are NOT touched
  if (name === 'voucher') {
    query = {end_time: {$lt:time_criteria_in_seconds}};
  }

  // guest authorization
  if (name === 'guest') {
    query = {end: {$lt:time_criteria_in_seconds}};
  }

  // if a user was only seen ONCE, $last_seen will not be defined
  // so, if $last_seen not defined, lets use $first_seen instead
  // also check if $blocked or $use_fixedip is set. If true, do NOT purge the
  // entry no matter how old it is. We want blocked/fixed_ip users to continue
  // blocked/fixed_ip. Also noted users should not be deleted.
  if (name === 'user') {
    query = { blocked: { $ne: true}, use_fixedip: { $ne: true}, noted: { $ne: true}, $or: [
      {last_seen: {$lt:time_criteria_in_seconds} },
      {last_seen: { $exists: false}, first_seen: {$lt:time_criteria_in_seconds} }
    ]};
  };

}

if (query) {
  count1 = db.getCollection(name).count();
  count2 = db.getCollection(name).find(query).count();
}

```

```

print((dryrun ? "[dryrun] " : "") + "pruning " + count2 + " entries (total " + count1 + ") fr
if (!dryrun) {
db.getCollection(name).remove(query);
db.runCommand({ compact: name });
}
}
}

if (!dryrun) db.repairDatabase();

use ace_stat;
var collectionNames = db.getCollectionNames();
for (i=0; i<collectionNames.length; i++) {
var name = collectionNames[i];
var query = null;

// historical stats (stat.*)
if (name.indexOf('stat')==-0) {
query = {time: {$lt:time_criteria}};
}

if (query) {
count1 = db.getCollection(name).count();
count2 = db.getCollection(name).find(query).count();
print((dryrun ? "[dryrun] " : "") + "pruning " + count2 + " entries (total " + count1 + ") fr
if (!dryrun) {
db.getCollection(name).remove(query);
db.runCommand({ compact: name });
}
}
}

if (!dryrun) db.repairDatabase();

```

[mongo_prune_js.js](#) (3 KB)

Was this article helpful?

Yes

No

207 found this article helpful



© 2022 Ubiquiti Inc. All Rights Reserved.

[Terms of Service](#) | [Privacy Policy](#) | [Legal](#)