The Wayback Machine - https://web.archive.org/web/20220928221056/https://help.ui.com/hc/en-us/articles/2049114...

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

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.

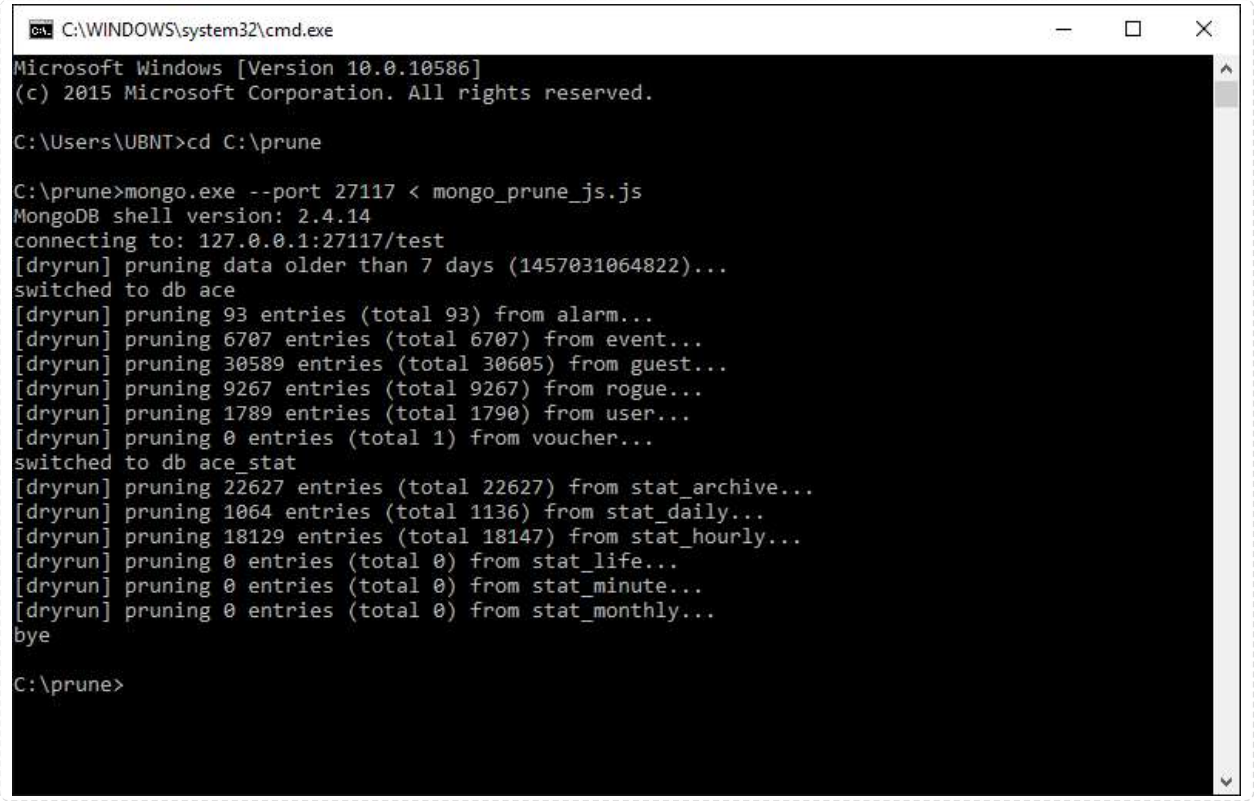
5. Enter the Working Directory. In the command prompt, change to your working directory by typing in the following command and hitting enter:

```
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:

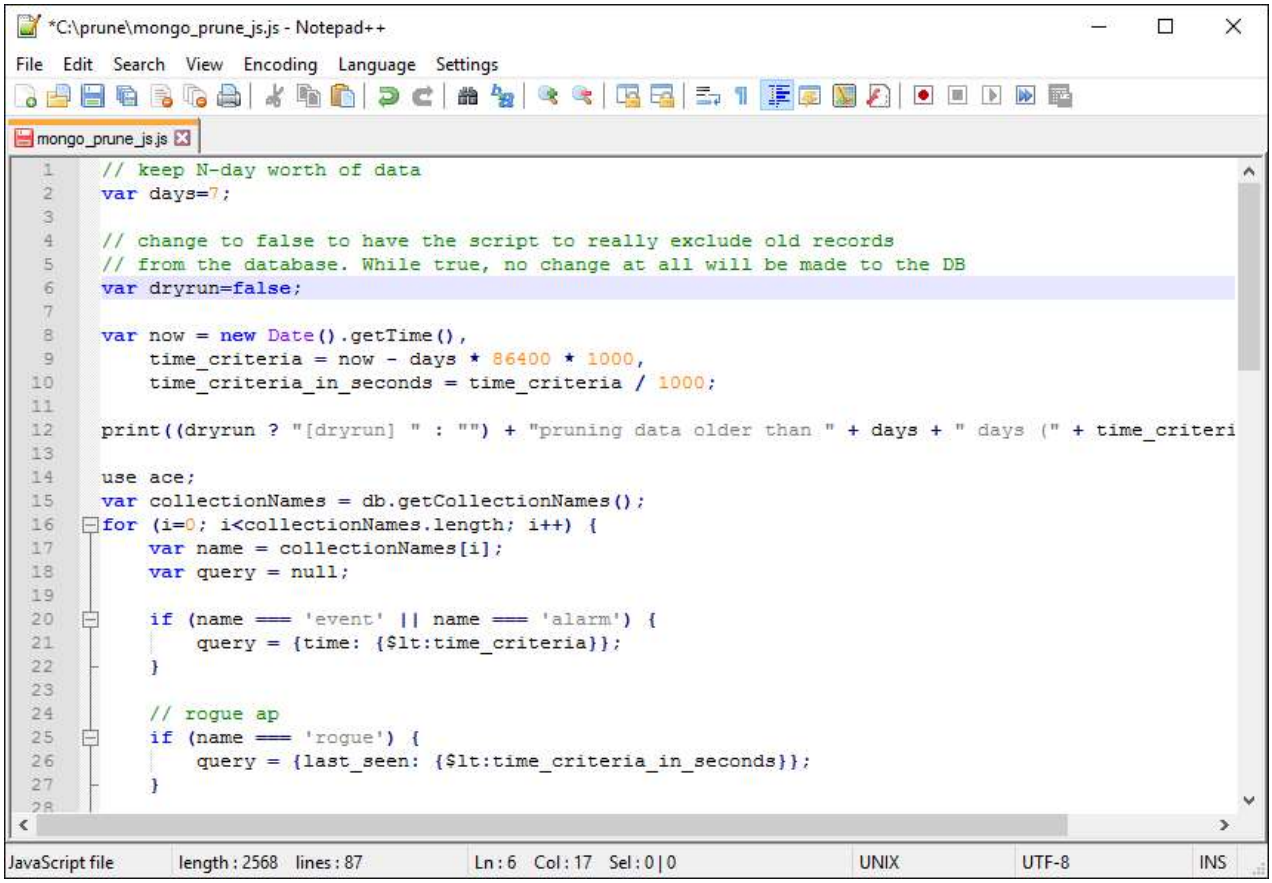
```
mongo.exe --port 27117 < mongo_prune_js.js
```

The output should look similar to this:



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.

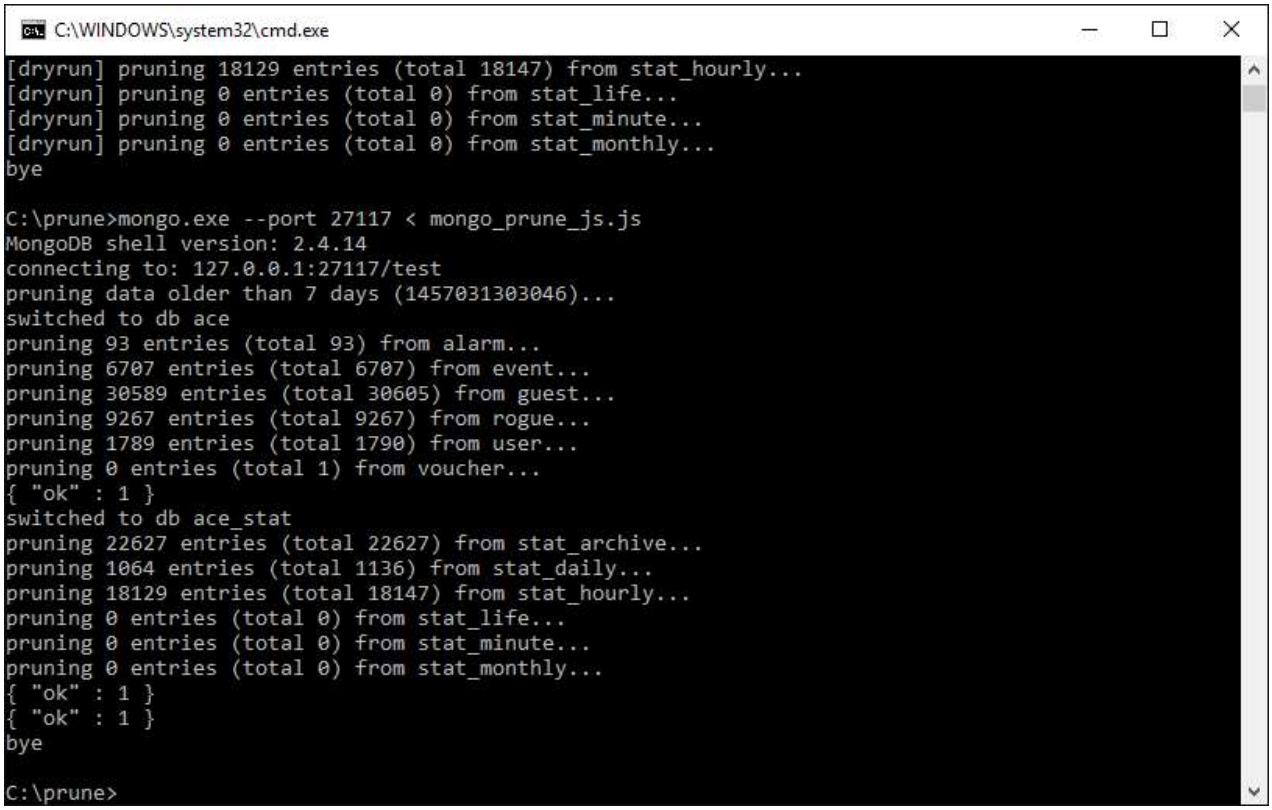


8. Prune the Database. Once changes have been made and saved, run the modified script, actually purging the database this time:

```
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:



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.

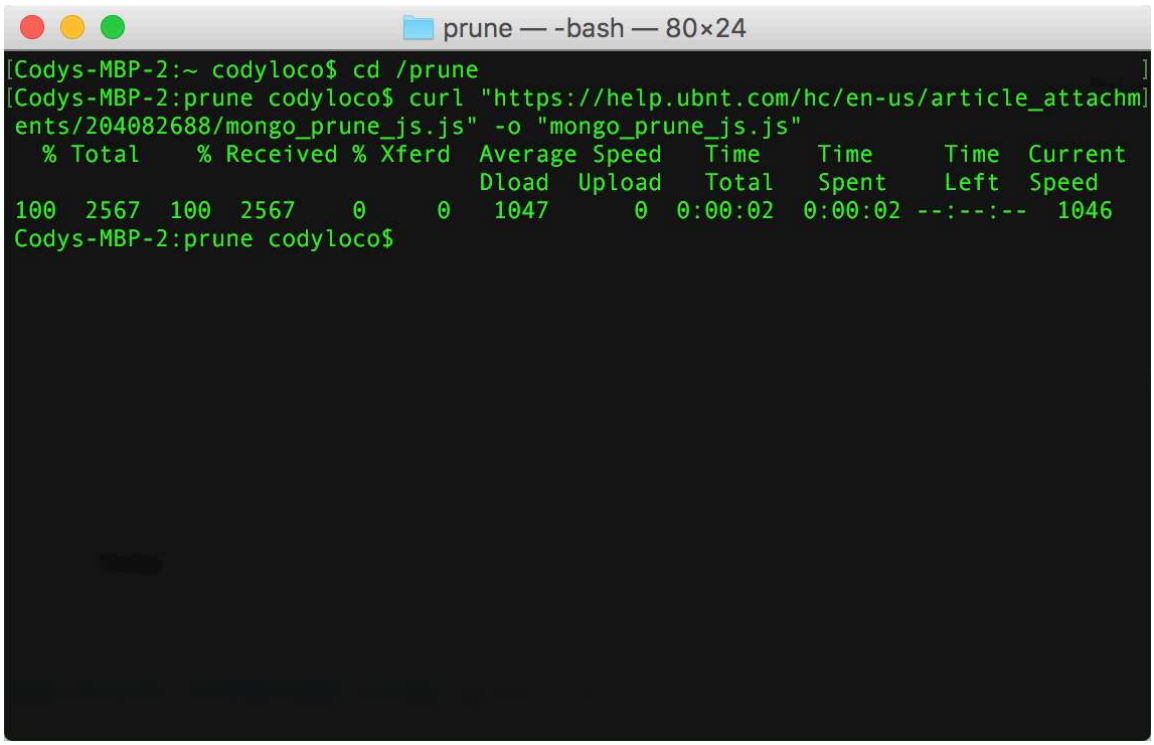
4. Move mongo to Working Directory. Locate the bin/mongo binary file, and copy it to your working directory. At this point, the downloaded .tgz package and any extra extracted files can be deleted as we only need the mongo binary.

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:

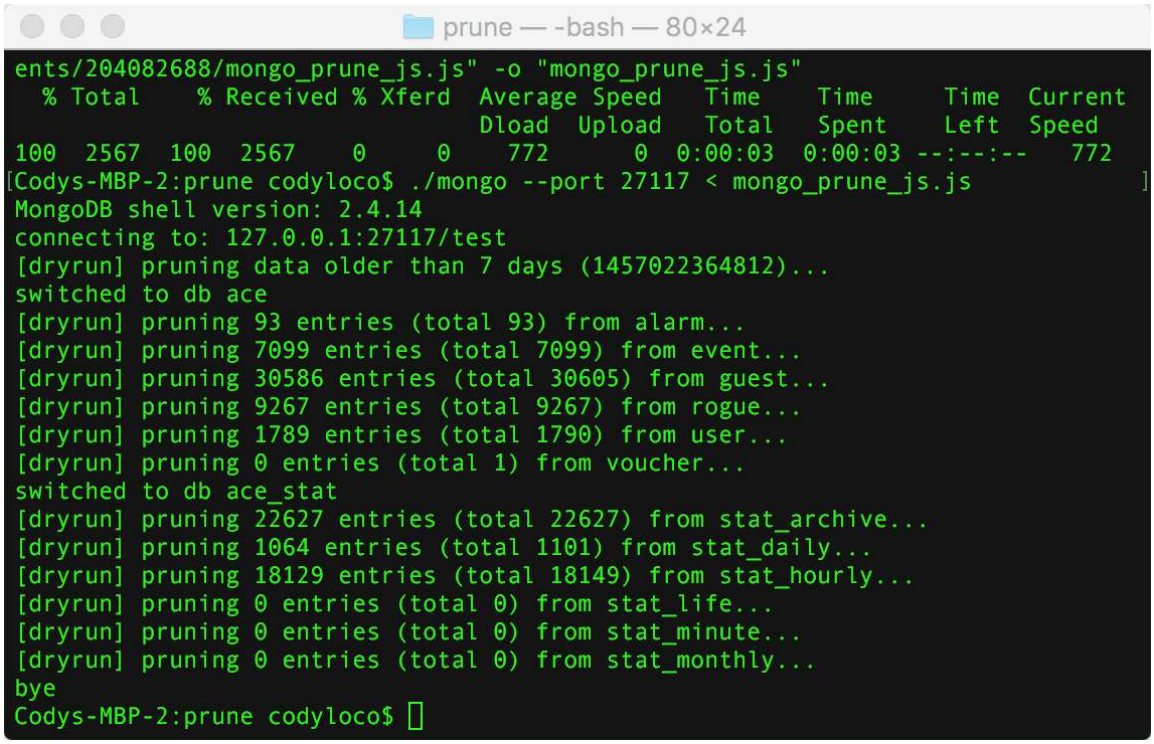
```
curl -kO "https://help.ui.com/hc/article_attachments/115024095828/mongo_prune_js.js" -o "mongo
```



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
```

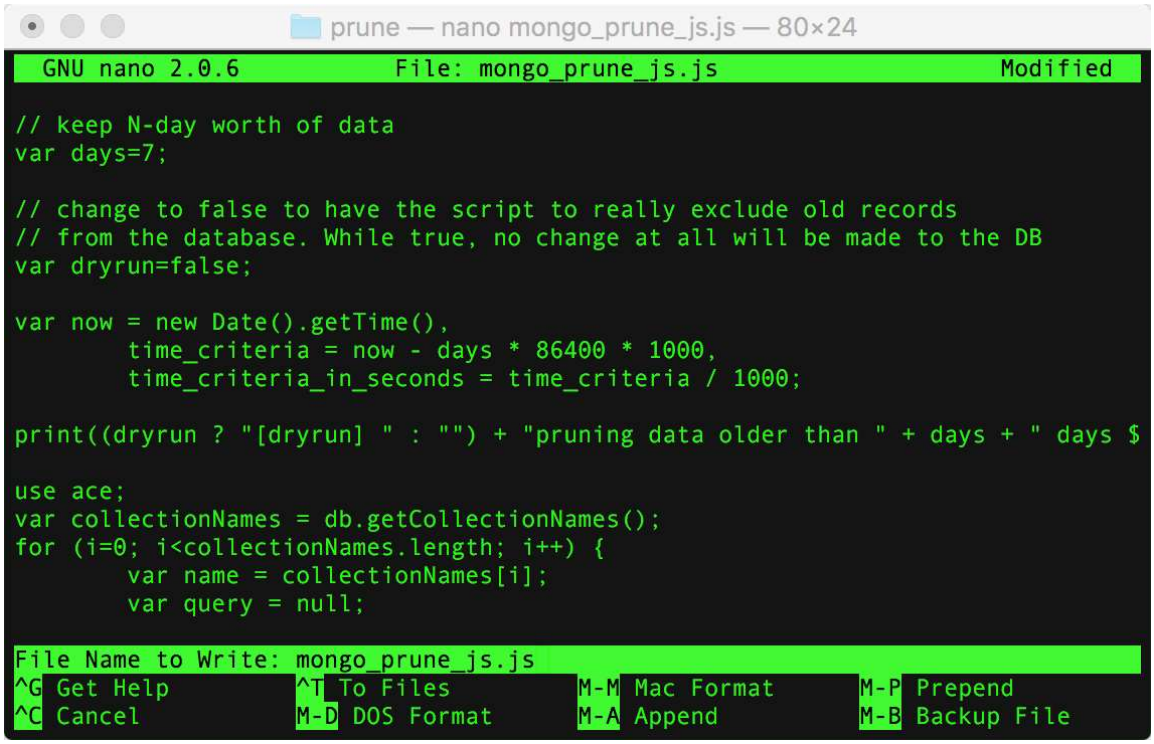
The output should look similar to this:



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.

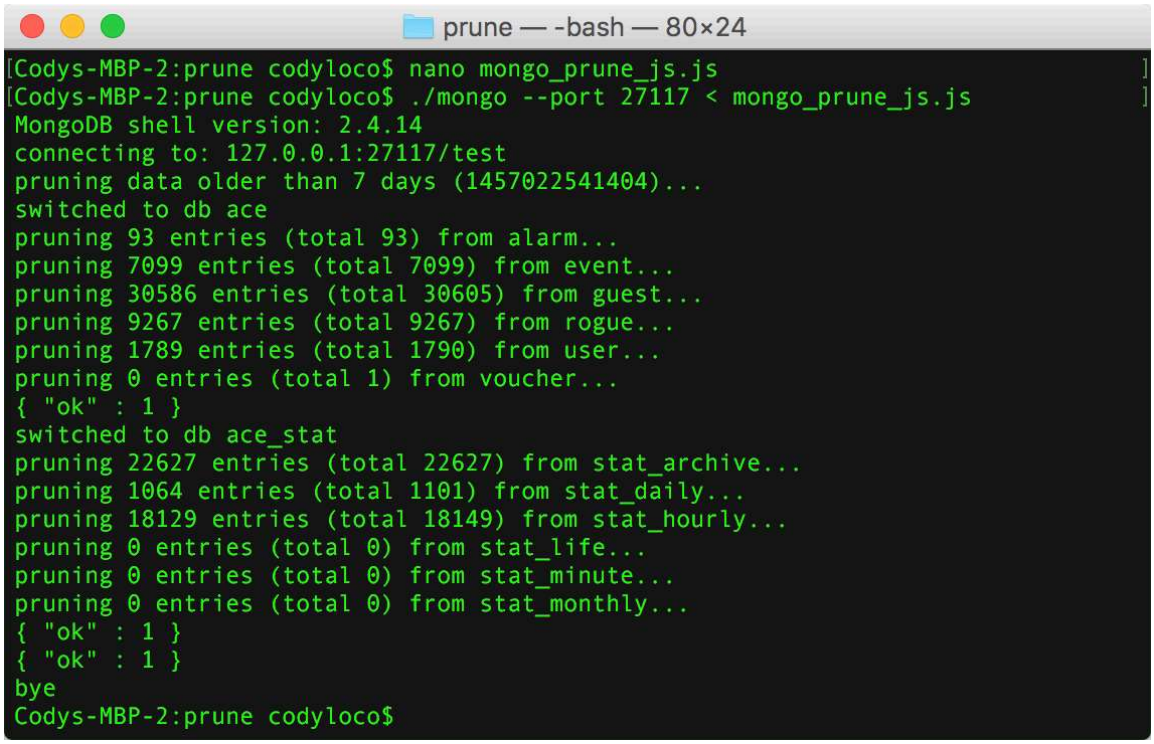


9. Prune the Database. Run the modified script, purging the database this time:

```
./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:



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
```

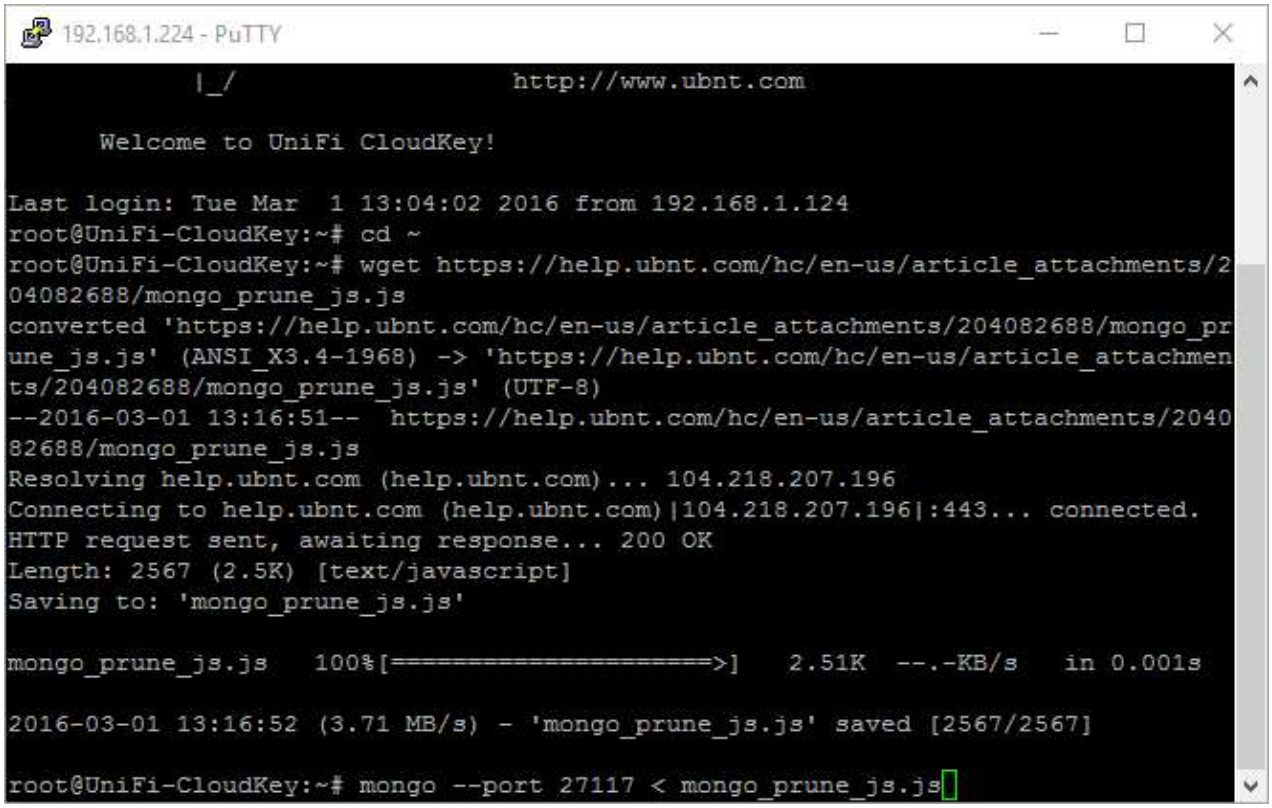
1. Establish Connection. Connect to your server via SSH using your preferred client and authenticate. The screenshots in this section of the article are from PuTTY on Windows. If using a Linux or Mac client, you can connect to the server using the built-in Terminal application.

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:

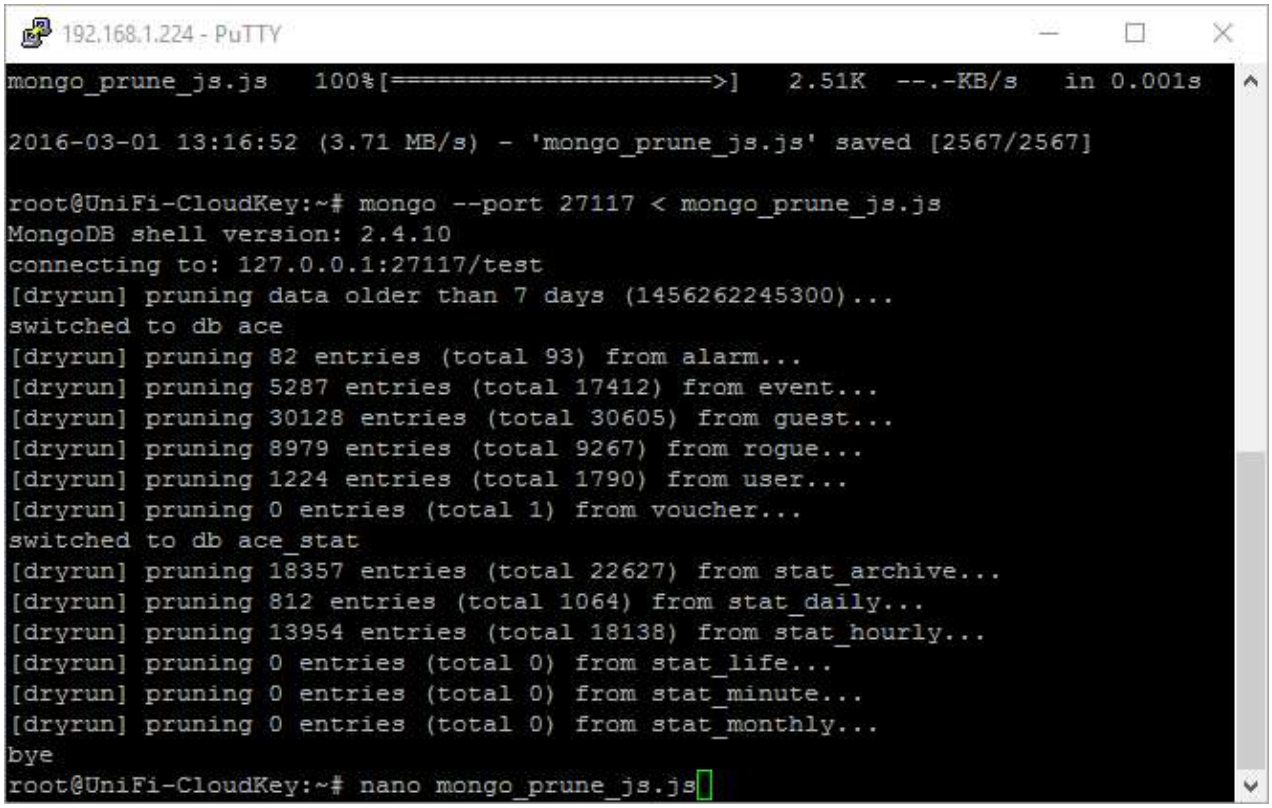
```
curl -kO https://help.ui.com/hc/article_attachments/115024095828/mongo_prune_js.js
```



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
```

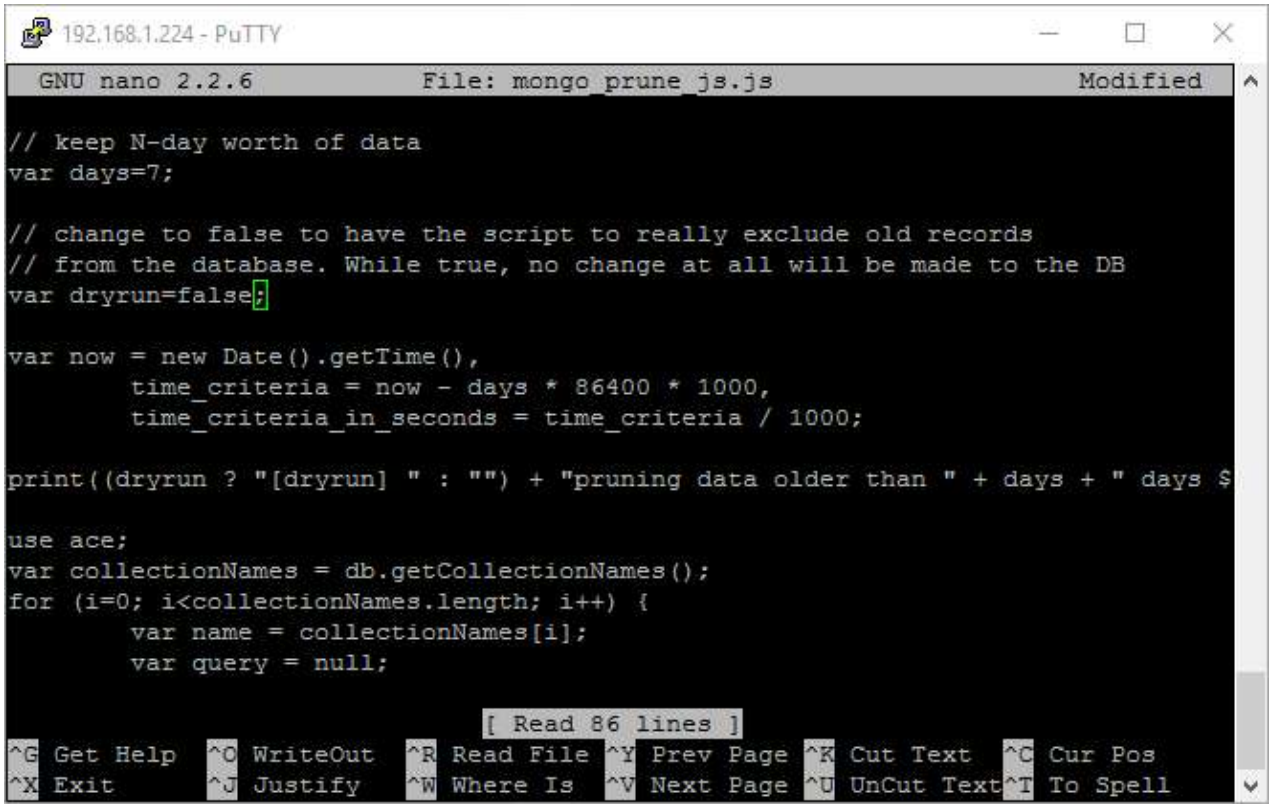
The output should look similar to this:



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 *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.



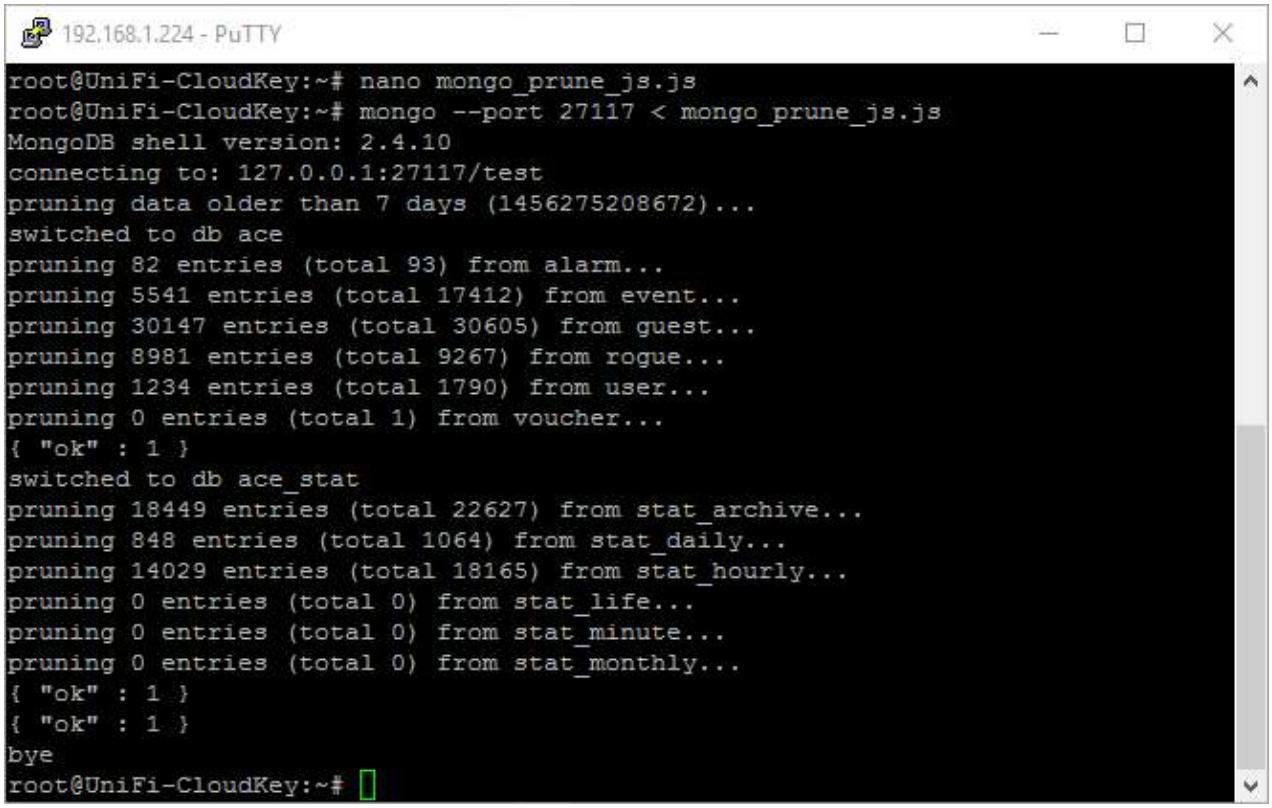
> **NOTE:** If you receive an error indicating nano is not installed, you can install it with the following command, then run the previous command again:
>
> ```
> sudo apt-get update && sudo apt-get -y install nano
> ```

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:



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.