Pruning Flex Storage Data from PingPlotter 5 - Windows Instructions

Saved From:

https://www.pingman.com/kb/article/pruning-flex-storage-data-from-pingplotter-5-windows-instructions-141.html

THE FOLLOWING ONLY APPLIES TO INSTALLATIONS USING FLEX STORAGE (.PPSAMPLE FILES) AND VERSIONS 5.8.1 AND NEWER

Question

My PingPlotter 5 data is getting pretty large. Is there a way on Windows to prune data from PingPlotter 5 Flex Storage?

Solution

Yes! In fact, it"s pretty easy. We"ll go over a couple of ways to do that below, starting with manual deletion, then deletion using a script, then putting the script on a Task Schedule. If you're hoping to speed through this and you want to save time, we have a script in the Attachments section at the bottom of this article that you can customize to delete all data older than 1 month from Flex Storage.

Manual Deletion If you"re in a hurry, you might just want to manually delete some files yourself. Doing so with Flex Storage is simple!

All you"ll need to do is to close the app and stop the service, if applicable, and navigate to one of the following folders depending on your installation:

- ***%Programdata%\PingPlotter 5\SampleSets (If installed as a service)***
- ***%Localappdata%\PingPlotter 5\SampleSets (if installed as an application)***

In this folder, you"ll see a bunch of folders (we heard you like folders...).

These folders are organized by the date range specified in *Edit â†' Options â†' Auto-Save Data* (we"ve elected to save by month).

If you"d like, you can go ahead and remove entire folders, starting with the oldest first. Alternatively, for more precision, you can head into the folders and remove certain sample set files.

You have the option of sorting these files by modified date and deleting the oldest ones, or you can read through all the files to figure out which ones you don"t want. Here"s a breakdown of what the files look like

and how to read them:

pingplotter_com 2019-04 5xk77hznemdrbpg6955wu8ta1h.pp_sample

Pingplotter_com - This is the target, with underscores substituted for periods. **2019-04** - This is the date that the file started (YYYY-MM). If you"ve saved by day, you"ll probably see it look like â€œ2019-04-17â€• (YYYY-MM-DD). **5xk77hznemdrbpg6955wu8ta1h** - This is the GUID. This ties the session to the data. Make sure that you don't change this value. Otherwise, the data will no longer be tied to the session.

Pruning the data automatically

If you"ve got bigger fish to fry than going through your files and deleting them manually, you can absolutely have a script do it for you. We"ve done it here using a Windows PowerShell script. You can use the one we've made or you can make something to your own liking!

The download for our Windows PowerShell script is available at the bottom of this page. Here"s what we"ve put into it:

```
param ([int] $keepdays = 0, [int] $keepmonths = 1, $datadir = "", $dryrun = 1);
 if ($dryrun) {
 Write-Host â€œDRYRUN - data isn"t actually being deleted.â€•
 } else {
 Write-Host "LIVE - data is being deleted."
 }


# Figure out the date beforehand that it is OK to delete:
 #
 # -keepdays Is in the format of days to keep. For example, on monthly
flexstore, a value of 30 days
 # won"t delete the month of January on March 1st because #January 31st is less
than 30 days away.
 #
 # -keepmonths Is in the format of months to keep. On monthly flexstore, 1 month
will always keep at least
 1
 # full month, so January would be deleted on March 1st, but February won"t be
deleted
 # until there"s a complete full month available - February would be deleted on
April 1st though
 # (March is complete as well)
 #
 # -datadir If a custom data directory is in use, this is where the data will be
deleted from.
 # Should *not* include the "SampleSets" part of the directory (just the root
storage directory).
 #
 # -dryrun Just show what"s going to be deleted and don"t actually delete
anything. Defaults to 1.
 # Change the script or pass 0 as a parameter to actually delete data.
```

```
#

 if ($keepdays -gt 0) {
$keepdate = (Get-Date).AddDays(-$keepdays);
} elseif ($keepmonths -gt 0) {
$keepdate = (Get-Date).AddMonths(-$keepmonths);
} else {
throw "Both keepdays and keepmonths are 0 or negative. Must specify something
to delete."
}
$keepdate = $keepdate.ToUniversalTime();
Write-Host "Deleting data files that only contain data collected before:
$keepdate UTC"

#
# If datadir is passed in, then use that. Otherwise, check if running as a
service and use that.
#
# It"s possible that a data file is open and if that"s the case it won"t be
deletable. It should be
# deleted next time through. Open files have recently had something done to
them, but they get closed
# after a few minutes of access.
#
# Get a list of all directories that may be storing data in:
# As application: %localappdata%\PingPlotter 5\SampleSets
# As service: %programdata%\PingPlotter 5\SampleSets
#

$folders = @();
if ($datadir) {
$folders += $datadir;}
else {
if (Get-Service "PingPlotter5" -ErrorAction SilentlyContinue) {
$folders += ($env:ProgramData + "\PingPlotter 5")
Write-Host "PingPlotter is installed as a service."
} else {
$folders += ($env:LocalAppData + "\PingPlotter 5")
Write-Host "PingPlotter is installed as an application, not a service."
}
}

#
# For customization, you can add more directories here, but PingPlotter really
only collects data
# in a single folder.
#

 Foreach ($folder in $folders)
{
$folder = Join-Path $folder "SampleSets"
Write-Host "Deleting in $folder";
```

```
 #
 # Note: All file dates are UTC, not local time. Timestamps on files are local
time, so they may not match
 # exactly. Also, files may be written to after no more data is being collected
in to it - in the case
 # of a comment being created in the past, or if aggregation is run (or rerun)
for some reason.
 #
 # Iterate all date-looking directories in that folder. Date-looking directories
are in this format:
 # * Annual flexstore yyyy
 # * Monthly flexstore yyyy-mm
 # * Daily flexstore yyyy-mm-dd
 # Check the most recent piece of data that could be stored in this folder (end
of each period)
 # If the date is before "Keep" date, delete it.
 #

  $subfolders = Get-ChildItem -Directory $folder
 Foreach ($deletable in $subfolders) {
 $deletable = $deletable.Name
 if ($deletable -match â€œ^(\d\d\d\d)-(\d\d)-(\d\d)$â€œ) {


 # Daily file.
 $folderDate = (Get-Date -Year $Matches[1] -Month $Matches[2] -Day
$Matches[3]).Date.AddDays(1).AddMilliseconds(-1)
 } elseif ($deletable -match â€œ^(\d\d\d\d)-(\d\d)$â€œ) {

 # Monthly file.
 $folderDate = (Get-Date -Year $Matches[1] -Month $Matches[2] -Day
1).Date.AddMonths(1).AddMilliseconds(-1)
 } elseif ($deletable -match â€œ^(\d\d\d\d)$â€œ) {

 # Annual file.
 $folderDate = (Get-Date -Year $Matches[1] -Month 1 -Day
1).Date.AddYears(1).AddMilliseconds(-1)
 } else {
 Write-Host â€œ$deletable isn"t a date folderâ€•
 continue;
 }

 if ($folderDate -lt $keepdate) {
 Write-Host â€œ$deletable holds OLD data before $folderDate UTC and can be
deletedâ€•;
 if (!$dryrun) {
 $dirToDelete = Join-Path $folder $deletable
 Write-Host â€œDeleting $dirToDeleteâ€•;
 Remove-Item -Path $dirToDelete -Recurse
 }
 } else {
 Write-Host â€œ$deletable holds NEW data and will be kept ($folderDate) UTCâ€•
 }
```

```
}
}
```

Once you"ve got your handy dandy script downloaded, you"re ready to run it! First, check out the $dryrun output by pasting the code or running the script file using Windows Powershell command prompt to ensure your parameters are accurate. Then, edit the script file and set $dryrun = 0 - the script will then take action and prune your data directory.

Running the Script on a Schedule

You can easily set up an automated task to run the Powershell script with Windows Task Scheduler.

First, open Task Scheduler. You'll want to choose the option Create Task; using Create Basic Task will not contain a function that allows automation to complete successfully.

General

Here, you can name the task and give it a description. We've chosen to run whether the user is logged on or not; you can make your selection based on appropriate organizational protocol. It is imperative that "Run with highest privileges" is selected; the automation will not function unless this is selected.

Triggers

The task can be set to run as frequently as you'd like. We'd recommend setting a trigger to recur on a monthly basis.

Actions

Set the action to start a program, and browse for the script file you've downloaded and edited. This will all be unattended and will close upon completion.

As seen in the image, you'll want to pass powershell in the Program/script field, then enter the location of the script file within the Add arguments field.

Conditions, Settings

For simplicity's sake, we have left these values at defaults. You're welcome to alter these values to fit your needs.

Finishing up

Once you've performed all these steps, the automation should run successfully on the schedule you've dictated. Feel free to alter settings and experiment as needed; you will always know what's best for your data and your organization. Should you find yourself in need of support, please don't hesitate to reach out to support@pingman.com and we'll gladly help however we can.