Dotnet Core Documentation

Contents

[Setup Dotnet Core on CentOS 2](#_Toc520186925)

[Setting up Dotnet Core Watch 3](#_Toc520186926)

[Configuration Settings 4](#_Toc520186927)

[MySQL: 4](#_Toc520186928)

[Payeezy: 4](#_Toc520186929)

[Quickbooks 4](#_Toc520186930)

[Other: 5](#_Toc520186931)

[Apache/Forwarding 6](#_Toc520186932)

[Setting up cron job 7](#_Toc520186933)

[Startup Dotnet Services from Shellscript 9](#_Toc520186934)

[Requirements: 9](#_Toc520186935)

[Debugging: 9](#_Toc520186936)

[Shellscript Example: 10](#_Toc520186937)

# Setup Dotnet Core on CentOS

sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc

sudo sh -c 'echo -e "[packages-microsoft-com-prod]\nname=packages-microsoft-com-prod \nbaseurl=https://packages.microsoft.com/yumrepos/microsoft-rhel7.3-prod\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/dotnetdev.repo'

sudo yum update

sudo yum install libunwind libicu

sudo yum install dotnet-sdk-2.0.0

export PATH=$PATH:$HOME/dotnet

dotnet --version

# Setting up Dotnet Core Watch

* Have a directory on your machine which will include the dotnet core project files (where the shell script file will later point to for rebooting the service).  
  (dotnetcore/office.starsupplies.com/Stairsupplies/Stairsupplies) -> This is the folder we need to copy
* Create a configuration file in the root of the created folder for settings and name it: “**stairsupplies.cfg**”.

# Configuration Settings

Each setting in the “**stairsupplies.cfg**” file should be in the following format on a new line: “**Setting\_Name:Value\_Here**”. Empty lines will be ignored, along with lines starting with “**#**”.

## MySQL:

* mysql\_username
* mysql\_password
* mysql\_server
* mysql\_port
* mysql\_db\_name

## Payeezy:

* gateway\_id
* gateway\_password
* key\_id
* hmac\_key
* payeezy\_mode
  + “**live**” or “**test**” (use ‘live’ for submitting transactions live, or ‘test’ for using a demo account)

## Quickbooks

* quickbooks\_client\_id
  + Go to “**My Apps**” on QB, select app, and click “**Keys**” to retrieve this value.
* quickbooks\_client\_secret
  + Go to “**My Apps**” on QB, select app, and click “**Keys**” to retrieve this value.
* quickbooks\_redirect\_uri
  + The redirect URI that is being used for the QB login. This URI must be set on QB as well --- under your “**App > Keys**”.
* quickbooks\_realm\_id
  + This will be required when we use client credentials with OAuth, opposed to requiring a user login. You can find this value by going to your app’s dashboard, followed by pressing the following key combination: “**CTRL + ALT + /**”. This will open a dialog containing the “**Company ID**”.
* quickbooks\_mode
  + “**live**” or “**test**”.
* In the AppCommon.cfc file, set the global variable: “**Application.QUICKBOOKS\_API\_CLIENT\_ID**”.

## Other:

* log\_path
  + The directory path to store all log files, ex: “**C:\StairSupplies\**”. A sub directory will be created underneath for each of the APIs, along with a directory for exceptions, and failures which will contain the text file logs.

# Apache/Forwarding

**\*\*\*By default, Apache should already be installed.**

**Edit the following file:**

/etc/httpd/conf/vhosts/office\_stairsupplies.conf

**Within the ‘VirtualHost’ tag, input the following:**

ProxyPass "/ironbaluster/app1" "http://localhost:5000"

ProxyPassReverse "/ironbaluster/app1" "http://localhost:5000"

# Setting up cron job

* Create a shell script file in a different directory (such as script.sh). Include the following code for the shell script: (/home/peakey/cron/script.sh)

#!/bin/bash

SERVICE\_PATH="/home/peakey/dotnetcore";

LOG\_PATH="/home/peakey/cron/test.txt";

OUTPUT=$(curl "http://localhost:5000/api/hubspot/")

RESPONSE="$?";

if [ $RESPONSE -eq 0 ]; then

echo 'SUCCESSFULLLLLL';

echo "$OUTPUT" > $LOG\_PATH

elif [ $RESPONSE -eq 7 ]; then

echo 'Oh no, the connection was refused!';

cd $SERVICE\_PATH

pkkill dotnet

dotnet restore

dotnet watch run &

else

echo 'Other error';

echo "$?";

fi

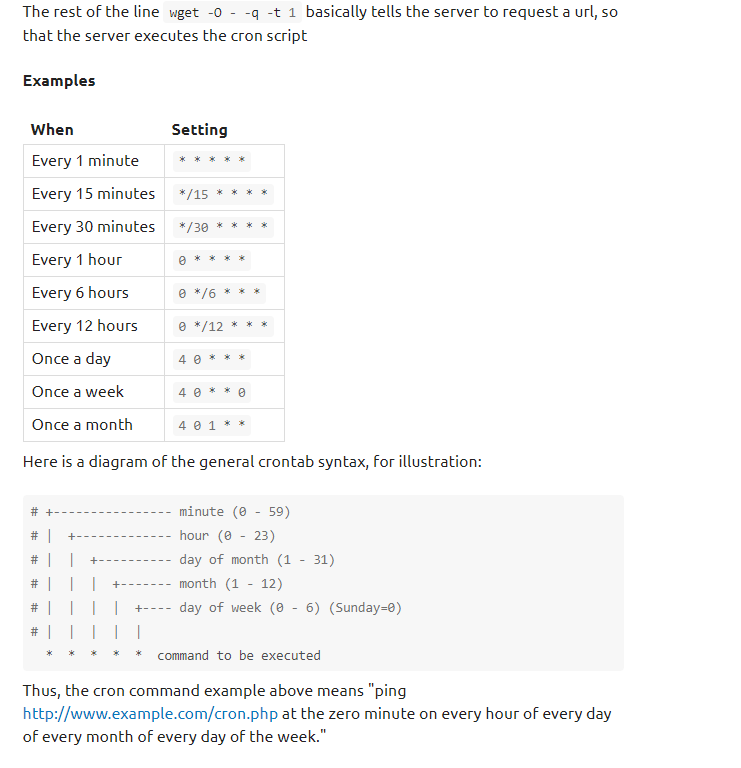
* To create a cron job (scheduled task) for your script, navigate to **‘/etc’**. Using **‘ls’**, you should find a file titled **‘cron.hourly’**. Running the following command will create/link to your shell script file:

sudo ln -s /home/peakey/cron/script.sh dotnet\_watch

* To specify a time for the cron job, you can use the command **‘crontab -e’**. For running it hourly, you can input the following to this file:

\* \* \* \* \* /home/peakey/cron/script.sh

**Time examples:**



* Finally, navigate to **‘/etc/init.d’**. Run the following command as before to create/link to your shell script file:

sudo ln -s /home/peakey/cron/script.sh dotnet\_watch

# Startup Dotnet Services from Shellscript

## Requirements:

1. Create a directory under the root titled ‘**nonexistent**’. This will be required for the Nuget configuration, when the shellscript runs the ‘**dotnet restore**’ command.
   1. Be sure to set the user/group permission on this directory to ‘**nobody:nogroup**’ (or whichever one you are using). Example: ‘***chown nobody:nogroup /nonexistent***’. If not, an error will be received with ‘**Permission Denied**’.
2. In the dotnet directory (where the source code is), navigate to ‘**bin/Debug/netcoreapp2.0**’. Set the user/group permission to ‘**nobody:nogroup**’ for all files under this directory. Failure to do so will result in a ‘**Permission Denied**’ error.
3. If you encounter an error like ‘**Unable to obtain lock file access on '/tmp/NuGetScratch/lock**’, navigate to the ‘**/tmp/NuGetScratch/lock**’directory from the root. Either remove all files under this folder, or set the user/group permission to ‘**nobody:nogroup**’.

## Debugging:

* Use the command ‘**ps ax | grep dotnet**’ to view the running processes for dotnet.
* If you need to kill all dotnet processes, use the command ‘**killall dotnet**’.
* If everything is set/working properly, there should be roughly 4 dotnet services running after executing the shellscript. The running services should look something like:
  + Dotnet (always runs)
  + Dotnet exec
  + Dotnet watch run
  + Exec –depsfile

## Shellscript Example:

#!/bin/bash

DOTNET\_PROCESS\_COUNT=$(ps -C dotnet | wc -l)

SERVICE\_PATH="/ironbaluster.dev1/office/html/dotnetcore";

SERVICES\_RUNNING\_FILE="/ironbaluster.dev1/office/html/dotnetcore/dotnet\_services\_running.txt";

#LOG\_PATH="/ironbaluster.dev1/office/html/dotnetcore/test.txt";

# There will always be 1 running

if [ $DOTNET\_PROCESS\_COUNT -gt 1 ]; then

echo "Dotnet processes running.";

# exit 1

else

echo "Dotnet processes not found, restarting service...";

cd $SERVICE\_PATH

echo "Changed to dotnetcore path";

killall dotnet

#ps ax | grep dotnet | awk '{print $1}' | xargs kill -9 $1

echo "Stopped dotnet processes";

export DOTNET\_SKIP\_FIRST\_TIME\_EXPERIENCE=true

export DOTNET\_CLI\_TELEMETRY\_OPTOUT=1

rm -f $SERVICES\_RUNNING\_FILE # If exists, delete file created from dotnet (which is created after services are running)

dotnet restore

echo "Started dotnet restore";

echo "Preparing to execute dotnet watch run command...";

nohup dotnet watch run > /dev/null 2>&1 &

while true; do

if [ -f $SERVICES\_RUNNING\_FILE ]; then

# Services are running

echo "STARTED";

break

else

# Services are not running yet

sleep 2

fi

done

exit

fi