osmtask

Version 1.2

September 2017

The MIT License

SPDX short identifier: MIT

Introduction

osmtask is a utlity designed to enhance the Synology DSM 6 task scheduler by providing batch sequencing of a list of tasks in a task file including optional conditional task starting.

With osmtask you may have a single task scheduler start time that runs osmtask, osmtask then triggers a grouped list of tasks. For example:

- Run backup preparation task (e.g. delete temporary files, fetch backup data from remote systems etc).
- Wait for preparation task to complete
- Start cloud upload backup task
- In parallel, if and only if the right USB disk is connected, start USB disk backup task

osmtask is designed to start a sequence of other tasks in the task scheduler, not to run programs directly. This has the benefit of using the maximum amount of standard Synology code with the email-on-error processing provided by DSM 6.

osmtask must be run with root privilege as the DSM task scheduler ignores task start commands from any other id. I would be uncomfortable running some unknown code as root but I hope this is mitigated by the fact that osmtask is less than 700 lines of easily readable and commented Python 2.7 source code so an assurance review should be a relatively easy and quick process.

Please Note

This software is not known to Synology and is not endorsed by Synology.

This software uses Synology DSM commands included with DSM 6 to list and to start tasks in the task scheduler, and to send DSM desktop notifications, but makes no modification to any Synology software. The osmtask command itself is a single file of Python source code and may be installed on a regular file share.

Caution: This tool requires “root” privilege to operate. As with any program with this privilege, bugs or misuse could cause significant damage. This software is provided to you in readable Python source code for your inspection prior to execution so use of this code is entirely at your own risk. The terms of use are spelled out formally in the MIT License included in this repository and by using the --license option in each program. If you are not willing to accept these terms of use, do not run this code.

Overview of Task File Syntax

Anything after a # in the task file is considered a comment.

Start a Task

To run a named task from the DSM task scheduler, just place the name of the task on a line in the task file:

```
MYPREPTASK              # Start MYPREPTASK task in the task scheduler
```

Wait for a Process to Complete

osmtask does not wait for tasks to complete before moving on to the next line of the task file. If you wish to wait for completion, use the @waitno directive.

@waitno is followed by some search text that is checked against the output of a ps command listing the CMD text for every process running on the system. osmtask only procedes when the search fails to find the provided text.

@waitno always waits a minimum of 5 seconds to avoid the race condition of the @waitno directive being checked before the prior task has properly started. Following that, @waitno rechecks every 10 seconds up to a maximum of 23 hours.

```
@waitno searchtext    # Wait until no process CMD contains searchtext
```

The command used to list the CMD portion of all processes running on the system is:

```
ps -eo '%a'
```

Sleep for Integer Seconds

A simple directive allows a pause in processing for the requested (integer) number of seconds.

```
@sleep 10             # Pause for 10 seconds
```

Conditionally Start Task

Only start task if a file or directory matching the condition name exists. The condition is separated from the task name by a question mark.

```
condition ? task      # Only start task if condition exists
```

If condition contains a forward slash anywhere it is assumed to be an absolute path and the existence of that path is checked.

```
/path/conditionfile ? task  # Run task if /path/conditionfile exists
```

If the condition does not contain a forward slash then it represents a file or folder name to be searched for. For example:

```
usb_dir ? usb_backup  # Search for usb_dir before starting usb_backup
```

By default osmtask will search in the top directory of any attached USB device and one level below the top directory. This may be changed with command line options. The default search is run with the command:

```
find “/volumeUSB?” -maxdepth 2 -name “usb_dir” -print
```

If this command finds something then the task is started.

More generally, the search is:

```
find DIR_SEARCH -maxdepth MAX_DEPTH -name “condition” -print
```

with DIR_SEARCH defaulting to “/volumeUSB?” and MAX_DEPTH defaulting to 2. Both may be changed on the command line.

More complex behaviour is easy to create from these basic operations. It is possible for earlier tasks to create or delete condition files/folders that are used later in the batch sequence.

Task Configuration for Task File Tasks

Due to the function of the DSM task scheduler, tasks to be started by osmtask must have “enabled” selected. If you wish only osmtask to start these tasks, and never the task scheduler on its own, the easy set-up is to schedule the task to “Run on the following date” and then pick a date in the past.

Task Configuration for osmtask Task

osmtask is normally be placed into it's own user-defined DSM task scheduler task. This task needs user to be root. Without root privilege the DSM task scheduler will ignore task start requests from osmtask.

The task settings run command will be something like:

```
/path/to/osmtask --task_file "/path/to/taskfile.txt"
```

I prefer a slight enhancement of this which sends detailed output to a log file as well as allowing the task scheduler email-on-error to pick up output.

```
/path/to/osmtask --task_file "/path/to/taskfile.txt" -verbose -time_stamp |
tee "/path/to/taskfile.log"
```

Command Line

The following information is available by typing "osmtask -h" at a command line:

```
usage: osmtask [-h] [-D DIR_SEARCH] [-l] [--max_depth MAX_DEPTH] [-q]
               [-t TASK_FILE] [-T] [-v]

Synology DSM multi-task runner: process a task list with some simple process
control.

optional arguments:
  -h, --help            show this help message and exit
  -D DIR_SEARCH, --dir_search DIR_SEARCH
                        search for conditional files below (default:
                        "/volumeUSB?")
  -l, --licence, --license
                        display program licence and exit
  --max_depth MAX_DEPTH
                        max search depth below dir_search (default: 2)
  -q, --quiet           suppress intro and summary unless alerting
  -t TASK_FILE, --task_file TASK_FILE
                        task file containing task list
  -T, --time_stamp      add time stamps to output
  -v, --verbose         show progress, -vv for debug level

Copyright (c) 2017, Chris Baker <osdr@ctac.me.uk> This software is made
available under the MIT License, please use -l to display.
```

Example Task File

This example file is used with other OSDR utilities for pre-processing some cloud updates, then uploading to cloud using Hyper Backup, then copying backup data to USB if one of 3 off-site USB disks is connected. Finally we record the last updated time and free space of those off-site disks.

```
OSRFCLOUD             # Start the task that updates recent file cloud

@waitno osrfcloud     # Wait until it is done

rfcloud to s3         # Sync the recent file cloud to s3 via Hyper Backup

# Now we copy backup image files to USB via rsync to a USB directory

osersync ? OSERSYNC   # If USB connected, send backup images to USB dir

@waitno osersync      # Wait until the task completes

# Send the NAS share data to USB with individual Hyper Backup tasks.

DS_Orange.hbk ? HB Orange   # Hyper Backup to USB disk "Orange"
DS_Yellow.hbk ? HB Yellow   # Hyper Backup to USB disk "Yellow"
DS_Purple.hbk ? HB Purple   # Hyper Backup to USB disk "Purple"

@waitno HyperBackup         # Wait for all HB processes to end

# Copy Hyper Backup explorer to USB disk using HB rsync

recovery ? recovery to usb  # HB explorer for Win, Mac & Linux

@waitno HyperBackup         # Wait for all HB processes to end

# Record the update times and check/record free space for the USB disk

Track USB             # Run osutrack, to record last update & free space

@waitno osutrack      # Wait until the task completes
```