MONITORING UPS WITH NAGIOS AND Network UPS Tools (NUT)

Configuration on Debian10 Server

connected directly via serial interface to the UPS

Inhalt

A. Introduction

The solution for monitoring uninterruptible power supplies (UPS) with the Network UPS Tools is possible with the plugin ***check_ups*** which is included in Nagios Plugin Package.

Regarding the general rule that no plugin directly accesses the UPS interface the ***check_ups*** plugin rely on a corresponding **daemon upsd** that monitors the UPS and provides status information

B. With the Network UPS Tools (NUT) and the check_ups Plugin

B.1 Installation-Preparation of the Nut-Server

1. Installation of the „nut“ package

Tipp: Installing the nut package, should add the nut user and group. If not add those manually.

2. entry for the Nut-Server in the file /etc/hosts

```
vim etc/hosts
```

for example:

```
192.168.188.xxx servername
```

3. Stop the nut-server service

```
/etc/init.d/nut-server stop

[ ok ] Stopping nut-server (via systemctl): nut-server.service.
```

4. Stop the nut-client service

```
/etc/init.d/nut-client stop

[ ok ] Stopping nut-client (via systemctl): nut-client.service.
```

B.2 Configuration of the Nut-Server

1. Edit the /etc/nut/nut.conf file

```
vi /etc/nut/nut.conf
```

```
# Network UPS Tools: example nut.conf

#
##################################################################################
# General section
##################################################################################
# The MODE determines which part of the NUT is to be started, and which
# configuration files must be modified.
#
# This file try to standardize the various files being found in the field, like
# /etc/default/nut on Debian based systems, /etc/sysconfig/ups on RedHat based
# systems, ... Distribution's init script should source this file to see which
# component(s) has to be started.
#
# The values of MODE can be:
# - none: NUT is not configured, or use the Integrated Power Management, or use
#   some external system to startup NUT components. So nothing is to be started.
# - standalone: This mode address a local only configuration, with 1 UPS
#   protecting the local system. This implies to start the 3 NUT layers (driver,
#   upsd and upsmon) and the matching configuration files. This mode can also
#   address UPS redundancy.
# - netserver: same as for the standalone configuration, but also need
#   some more network access controls (firewall, tcp-wrappers) and possibly a
#   specific LISTEN directive in upsd.conf.
# IMPORTANT NOTE:
#  This file is intended to be sourced by shell scripts.
#  You MUST NOT use spaces around the equal sign!

MODE=standalone
```

2. Edit the /etc/nut/ups.conf file

```
vi /etc/nut/ups.conf

# Network UPS Tools: example ups.conf
[ups]
   driver = usbhid-ups
   port = /dev/ttyS0
  desc= “PR750ELCD”
# The section header ([upsname]) can be just about anything as long as
# it is a single word inside brackets.  upsd uses this to uniquely
# identify a UPS on this system.
#
# If you have a UPS called snoopy, your section header would be "[snoopy]".
# On a system called "doghouse", the line in your upsmon.conf to monitor
# it would look something like this:
#
#       MONITOR snoopy@doghouse 1 upsmonuser mypassword master
#It might look like this if monitoring in slave mode:
#
#       MONITOR snoopy@doghouse 1 upsmonuser mypassword slave
# Configuration directives
# ------------------------
#
# These directives are used by upsdrvctl only and should be specified outside
# of a driver definition:
#
#    maxretry: Optional.  Specify the number of attempts to start the driver(s),
#             in case of failure, before giving up. A delay of 'retrydelay' is
#             inserted between each attempt. Caution should be taken when using
#             this option, since it can impact the time taken by your system to
#             start.
#
#             The default is 1 attempt.
#
#  retrydelay: Optional.  Specify the delay between each restart attempt of the
#             driver(s), as specified by 'maxretry'. Caution should be taken
#             when using this option, since it can impact the time taken by your
#             system to start.
#
#             The default is 5 seconds.
#
# These directives are common to all drivers that support ups.conf:
#
#  driver: REQUIRED.  Specify the program to run to talk to this UPS.
#         apcsmart, bestups, and sec are some examples.
#
#    port: REQUIRED.  The serial port where your UPS is connected.
#         /dev/ttyS0 is usually the first port on Linux boxes, for example.
#
```

```
# #            upsd (Unix socket on Unix, Named pipe on Windows) without
#           waiting for these data to be actually consumed.  With
#           some HW, such as ePDUs, that can produce a lot of data,
#           asynchronous mode may cause some congestion, resulting in
#           the socket to be full, and the driver to appear as not
#           connected.  By enabling the 'synchronous' flag
#           (value = 'yes'), the driver will wait for data to be
#           consumed by upsd, prior to publishing more.  This can be
#           enabled either globally or per driver.
#
#           The default is 'no' (i.e. asynchronous mode) for backward
#           compatibility of the driver behavior.
#      < any other directives here >
```

```
#
# --- SECURITY NOTE ---
#
# If you use snmp-ups and set a community string in here, you
# will have to secure this file to keep other users from obtaining
# that string.  It needs to be readable by upsdrvctl and any drivers,
# and by upsd.
#
# ---
#
# This is where you configure all the UPSes that this system will be
# monitoring directly.  These are usually attached to serial ports, but
# USB devices and SNMP devices are also supported.
#
# This file is used by upsdrvctl to start and stop your driver(s), and
# is also used by upsd to determine which drivers to monitor.  The
# drivers themselves also read this file for configuration directives.
#
# Anything else is passed through to the hardware-specific part of
# the driver.
```

3. Make sure that nut properly detects the UPS- Start upsdrvctl

```
upsdrvctl start
```

```
Network UPS Tools - UPS driver controller 2.7.4

Network UPS Tools - Generic HID driver 0.41 (2.7.4)
USB communication driver 0.33
Using subdriver: CyberPower HID 0.4
```

4. Edit the /etc/nut/upsd.conf file

Tipp: If you need to monitor **multiple machines**, see the **man page for upsd.conf**

```
vi /etc/nut/upsd.conf

# Network UPS Tools: example upsd configuration file

#
# This file contains access control data, you should keep it secure.
#
# It should only be readable by the user that upsd becomes.  See the FAQ.
#
# Each entry below provides usage and default value.

#IP Address of Nut-Server
LISTEN 192.168.188.xxx
#If the Sever is the localhost
#LISTEN 127.0.0.1
ACCEPT servername
REJECT all

# =======================================================================
# MAXAGE <seconds>
# MAXAGE 15
#
# This defaults to 15 seconds.  After a UPS driver has stopped updating
# the data for this many seconds, upsd marks it stale and stops making
# that information available to clients.  After all, the only thing worse
# than no data is bad data.
#
# You should only use this if your driver has difficulties keeping
# the data fresh within the normal 15 second interval.  Watch the syslog
# for notifications from upsd about staleness.

# =======================================================================
# STATEPATH <path>
# STATEPATH /var/run/nut/usbhid-ups-cyberpower.pid

#
# Tell upsd to look for the driver state sockets in 'path' rather
# than the default that was compiled into the program.

# =======================================================================
#ACCEPT localhost
#REJECT all
# This defaults to the localhost listening addresses and port 3493.
# In case of IP v4 or v6 disabled kernel, only the available one will be used.
#
# You may specify each interface you want upsd to listen on for connections,
# optionally with a port number.
#
# You may need this if you have multiple interfaces on your machine and
# you don't want upsd to listen to all interfaces (for instance on a
# firewall, you may not want to listen to the external interface).
#
# This will only be read at startup of upsd.  If you make changes here,
# you'll need to restart upsd, reload will have no effect.

# =======================================================================
# MAXCONN <connections>
# MAXCONN 1024
#
# This defaults to maximum number allowed on your system.  Each UPS, each
# LISTEN address and each client count as one connection.  If the server
# runs out of connections, it will no longer accept new incoming client
# connections.  Only set this if you know exactly what you're doing.

# =======================================================================
# CERTFILE <certificate file>
# CERTFILE /usr/local/ups/etc/upsd.pem
#
# When compiled with SSL support with OpenSSL backend,
# you can enter the certificate file here.
# The certificates must be in PEM format and must be sorted starting with
# the subject's certificate (server certificate), followed by intermediate
# CA certificates (if applicable_ and the highest level (root) CA. It should
# end with the server key. See 'docs/security.txt' or the Security chapter of
# NUT user manual for more information on the SSL support in NUT.
#
# See 'docs/security.txt' or the Security chapter of NUT user manual
# for more information on the SSL support in NUT.

# =======================================================================
# CERTPATH <certificate file or directory>
# CERTPATH /usr/local/ups/etc/cert/upsd
#
# When compiled with SSL support with NSS backend,
# you can enter the certificate path here.
# Certificates are stored in a dedicated database (splitted in 3 files).
# Specify the path of the database directory.
#
# See 'docs/security.txt' or the Security chapter of NUT user manual
# for more information on the SSL support in NUT.

# =======================================================================
# CERTIDENT <certificate name> <database password>
# CERTIDENT "my nut server" "MyPasSw0rD"
#
# When compiled with SSL support with NSS backend,
# you can specify the certificate name to retrieve from database to
# authenticate itself and the password
# required to access certificate related private key.
#
# See 'docs/security.txt' or the Security chapter of NUT user manual
# for more information on the SSL support in NUT.

# =======================================================================
# CERTREQUEST <certificate request level>
# CERTREQUEST REQUIRE
#
# When compiled with SSL support with NSS backend and client certificate
# validation (disabled by default, see 'docs/security.txt'),
# you can specify if upsd requests or requires client's' certificates.
# Possible values are :
#  - 0 to not request to clients to provide any certificate
#  - 1 to require to all clients a certificate
#  - 2 to require to all clients a valid certificate
#
# See 'docs/security.txt' or the Security chapter of NUT user manual
# for more information on the SSL support in NUT.
```

5. Edit the /etc/nut/uspd.users file

Tipp: If you are monitoring from multiple machines **add multiple users.** see the **man page for upsd.users**

```
vi /etc/nut/upsd.users

Network UPS Tools: Example upsd.users

#
# This file sets the permissions for upsd - the UPS network daemon.
# Users are defined here, are given passwords, and their privileges are
# controlled here too.  Since this file will contain passwords, keep it
# secure, with only enough permissions for upsd to read it.

#--------------------------------------------------------------------------

# Each user gets a section.  To start a section, put the username in
# brackets on a line by itself.  To set something for that user, specify
# it under that section heading.  The username is case-sensitive, so
# admin and AdMiN are two different users.
#Example
            [user]
                upsmon master/slave
                password = userpassword
                actions = SET
                instcmds = ALL
#
#-----------------------------------------------------------------------
#
# "master" means this system will shutdown last, allowing the slaves
# time to shutdown first.
# "slave" means this system shuts down immediately when power goes critical.
#-----------------------------------------------------------------------

#
# password: The user's password.  This is case-sensitive.
#
#-----------------------------------------------------------------------
#
# actions: Let the user do certain things with upsd.
#
# Valid actions are:
#
# SET     - change the value of certain variables in the UPS
# FSD  - set the "forced shutdown" flag in the UPS
#
#-----------------------------------------------------------------------
#
# instcmds: Let the user initiate specific instant commands.  Use "ALL"
# to grant all commands automatically.  There are many possible
# commands, so use 'upscmd-l' to see what your hardware supports.  Here
# are a few examples:
#
# test.panel.start          - Start a front panel test
# test.battery.start        - Start battery test
# test.battery.stop         - Stop battery test
# calibrate.start - Start calibration
# calibrate.stop - Stop calibration
```

6. Edit the /etc/nut/upsmon.conf file

```
vi upsmon.conf
```

```
# Network UPS Tools: example upsmon configuration
# "master" means this system will shutdown last, allowing the slaves
# time to shutdown first.
# "slave" means this system shuts down immediately when power goes critical.
# This file contains passwords, so keep it secure.
MONITOR upsname@severname 1 user userpassword master/slave
SHUTDOWNCMD "/sbin/shutdown -h now"
POWERDOWNFLAG /etc/nut/killpower
#
# By default, upsmon splits into two processes.  One stays as root and
# waits to run the SHUTDOWNCMD.  The other one switches to another userid
# and does everything else.
#
# The default nonprivileged user is set at compile-time with
#       'configure --with-user=...'.
#
# You can override it with '-u <user>' when starting upsmon, or just
# define it here for convenience.
#
# Note: if you plan to use the reload feature, this file (upsmon.conf)
# must be readable by this user!  Since it contains passwords, DO NOT
# make it world-readable.  Also, do not make it writable by the upsmon
# user, since it creates an opportunity for an attack by changing the
# SHUTDOWNCMD to something malicious.
#
# For best results, you should create a new normal user like "nutmon",
# and make it a member of a "nut" group or similar.  Then specify it
# here and grant read access to the upsmon.conf for that group.
#
# This user should not have write access to upsmon.conf.
#
# RUN_AS_USER nut

# --------------------------------------------------------------------------
# MONITOR <system> <powervalue> <username> <password> ("master"|"slave")
#
# List systems you want to monitor.  Not all of these may supply power
# to the system running upsmon, but if you want to watch it, it has to
# be in this section.
#
# You must have at least one of these declared.
#
# <system> is a UPS identifier in the form <upsname>@<hostname>[:<port>]
# like ups@localhost, su700@mybox, etc.
```

7. Fix the permissions of the /etc/nut/ files

```
chown root:nut /etc/nut*

chmod 640 /etc/nut/*
```

8. Make sure upsd daemon and upsmon program start at system reboot

```
vi /etc/default/nut
```

```
START_UPSD=yes
START_UPSMON=yes
```

9. Start nut-server service

```
/etc/init.d/nut-server start

[ ok ] Starting nut-server (via systemctl): nut-server.service
```

10. Start nut-client service

```
/etc/init.d/nut-client start

[ ok ] Starting nut-client (via systemctl): nut-client.service.
```

B.3 The check_ups plugin

The check_ups plugin itself has the following options:

-H/--host	Addresss
-u /--ups	Name of UPS
-p / --port	Port
-v / --variable	Check_ups supports the following variables:
LINE :	Input voltage of the UPS
TEMP:	Temperature of the UPS
BATTPCT:	Remaining battery capacity in %
LOADPCT:	Load on the UPS in %
-w / --whole_ number	The warning limit as a whole number
-c/ --critical	Critical limit in connection with a variable
-T / --temperature	Temperature in degrees
-t / --timeout	Timeout in seconds, by default timeout is 10 sec. After these seconds the plugin stops the test and returns to CRITICAL state.

Note: If a variable is not used the plugin returns:

- CRITICAL if the UPS is in Status: OFF/On Battery/Low Battery/Replace Battery
- OK if the UPS is in Status: online/Calibrating/On Bypass/ Overload /Trimming /Boosting /Charging /Discharging

1. Test the plugin check_ups in /usr/local/nagios/libexec/

```
./check_ups -H localhost -u ups
```

Example

```
./check_ups -H localhost -u ups -T

UPS OK - Status=Online Utility=222.0V Batt=100.0% Load=5.0% |voltage=222000mV;;;0
battery=100%;;;0;100 load=5%;;;0;100
```

2. Create a command in the /usr/local/nagios/etc/objects/commands.cfg file

Transformed into a command object the above test for any host looks like this:

```
define command {

   command_name    check_local_ups
   command_line    $USER1$/check_ups -H $HOSTADDRESS$ -u $ARG1$ -T
}
```

3. Create a service for checking the ups occasionally in the usr/local/nagios/etc/objects/localhost.cfg file

For example hier is: Warning if > , critical if >10%, warning if >6%
# > 400 processes.

```
define service {

         use                      local-service
        host_name                      hp
        service_description       check _ups
        check_command           check_local_ups!ups!10%!6%
       check_interval                  5
        max_check_attempts     3
}
```

4. Verify the configuration

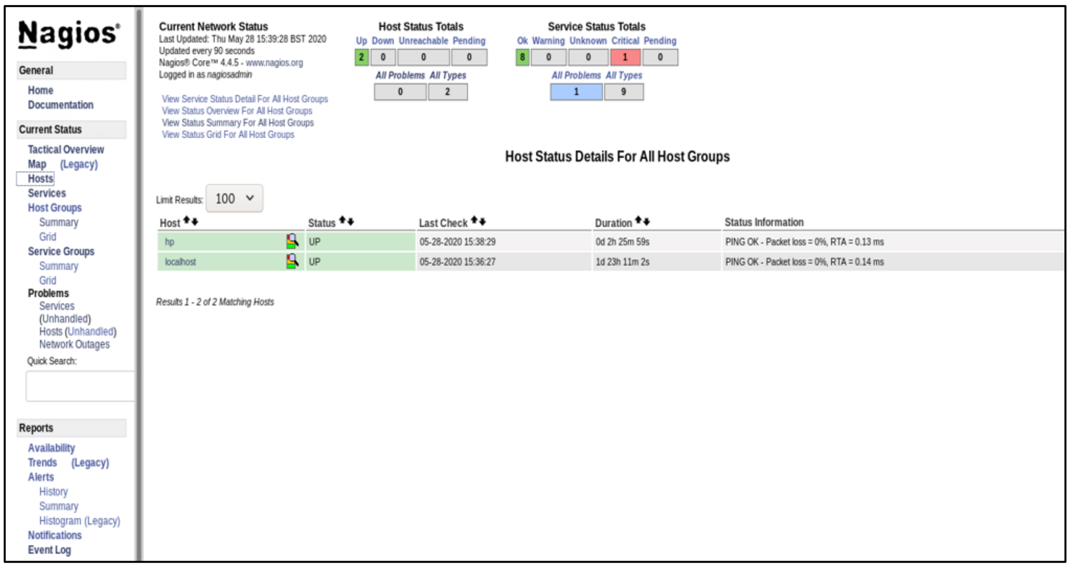
```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2019-08-20
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 9 services.
        Checked 2 hosts.
        Checked 1 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

5. Restart and check Nagios service:

```
service nagios restart
```

```
service nagios status
● nagios.service - Nagios Core 4.4.5
  Loaded: loaded (/lib/systemd/system/nagios.service; enabled; vendor preset: e
  Active: active (running) since Tue 2020-11-10 08:58:41 GMT; 36s ago
```

6. Check the results in your web browser:



Current Network Status
Last Updated: Thu May 28 15:39:28 BST 2020
Updated every 90 seconds
Nagios® Core™ 4.4.5 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	*All Types*
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
8	0	0	1	0

All Problems	*All Types*
1	9

Host Status Details For All Host Groups

Limit Results: 100

Host	Status	Last Check	Duration	Status Information
hp	UP	05-28-2020 15:38:29	0d 2h 25m 59s	PING OK - Packet loss = 0%, RTA = 0.13 ms
localhost	UP	05-28-2020 15:36:27	1d 23h 11m 2s	PING OK - Packet loss = 0%, RTA = 0.14 ms

Results 1 - 2 of 2 Matching Hosts