Centreon Plugins Documentation

Release

Merethis

January 11, 2019

centreon 46/52, rue Albert – 75013 Paris – FRANCE
tél +33 (0)1 49 69 97 12 fax +33 (0)1 78 12 00 28 mail contact@centreon.com
SAS au capital de 500 000 € – 483 494 589 RCS Créteil – TVA FR 02 483 494 589
www.centreon.com

Centreon Plugins is a common monitoring library and plugins written in Perl. It is licensed under the terms of the Apache License Version 2 as published by the Free Software Foundation.

Contents:

CHAPTER 1

User guide

1.1 Description

"centreon-plugins" is a free and open source project to monitor systems. The project can be used with Centreon, Icinga and all monitoring softwares compatible Nagios plugins.

The lastest version is available on following git repository: https://github.com/centreon/centreon-plugins.git

1.2 Installation

1.2.1 Debian Wheezy

Get the last version of "centreon-plugins" from the repository:

```
# aptitude install git
# git clone https://github.com/centreon/centreon-plugins.git
```

To monitor SNMP systems, you need to install the following packages:

```
# aptitude install perl libsnmp-perl
```

You can install other packages to use more plugins:

```
# aptitude install libxml-libxml-perl libjson-perl libwww-perl libxml-xpath-perl libnet-telnet-perl
```

To use 'memcached' functionality, you need to install the following CPAN module (no debian package): http://search.cpan.org/~wolfsage/Memcached-libmemcached-1.001702/libmemcached.pm

1.2.2 Centos/Rhel 6

Get the last version of "centreon-plugins" from the repository:

```
# yum install git
# git clone https://github.com/centreon/centreon-plugins.git
```

To monitor SNMP systems, you need to install the following packages:

```
# yum install perl net-snmp-perl
```

You can install other packages to use more plugins:

```
# yum install perl-XML-LibXML perl-JSON perl-libwww-perl perl-XML-XPath perl-Net-Telnet perl-Net-DNS
```

To use 'memcached' functionality, you need to install the following CPAN module (package available in 'rpmforge'):
http://search.cpan.org/~wolfsage/Memcached-libmemcached-1.001702/libmemcached.pm

1.3 Basic Usage

We'll use a basic example to show you how to monitor a system. I have finished the install section and i want to monitor a Linux in SNMP. First, i need to find the plugin to use in the list:

```
$ perl centreon_plugins.pl --list-plugin | grep -i linux | grep 'PLUGIN'
PLUGIN: os::linux::local::plugin
PLUGIN: os::linux::snmp::plugin
```

It seems that 'os::linux::snmp::plugin' is the good one. So i verify with the option `--help` to be sure:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --help
...
Plugin Description:
  Check Linux operating systems in SNMP.
```

It's exactly what i need. Now i'll the option `--list-mode` to know what can i do with it:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --list-mode
...
Modes Available:
 cpu
 cpu-detailed
 disk-usage
 diskio
 inodes
 interfaces
 list-diskspath
 list-interfaces
 list-storages
 load
 memory
 processcount
 storage
 swap
 tcpcon
 time
 uptime
```

I would like to test the 'load' mode:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=load
UNKNOWN: Missing parameter --hostname.
```

It's not working because some options are missing. I can have a description of the mode and options with the option `--help`:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=load --help
```

Eventually, i have to configure some SNMP options:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=load --hostname=127.0.0.1 --snmp-v
OK: Load average: 0.00, 0.00, 0.00 | 'load1'=0.00;;;0; 'load5'=0.00;;;0; 'load15'=0.00;;;0;
```

I can set threshold with options `--warning` and `--critical`:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=load --hostname=127.0.0.1 --snmp-v
OK: Load average: 0.00, 0.00, 0.00 | 'load1'=0.00;0:1;0:2;0; 'load5'=0.00;0:2;0:3;0; 'load15'=0.00;0:
```

1.4 FAQ

1.4.1 What can i monitor ?

The option `--list-plugin` can be used to get the list of plugins and a short description.

Headers of the table mean:

- Transport: The check has internal options for the transport.
- Protocol: what is used to get the monitoring datas.
- Experimental: The check is still in development.

Category	Check	Transport			Protocol					Experimental	Comm
		SSH	TELNET	WSMAN	SNMP		HTTP		WMI		JMX
Application	Active Directory								•		Use 'dcdia com- mand Must instal on W dows.
	Apache					•					Need Apacl 'mod modu
	Apc					•					
	Apcupsd	•							•		Use 'apcu com- mand
	Bluemind					•					Use ' fluxdl API.
	Checkmyws					•					

Continued

Table 1.1 – continued from previous page

Category	Check	Transport			Protocol					Experimental	Comm
		SSH	TELNET	WSMAN	SNMP		HTTP		WMI		JMX
	Elasticsearch					•					
	Exchange								•		Use pow-ershel script Must instal on W dows.
	Github					•					Use ‘githu API.
	Hddtemp								•		Open a T custo comm nicati
	IIS						•				Must instal on W dows. openv
			•			•			•		MIM
Jenkins					•						JSON LWP: URI, HTTI
Kayako					•					Use ‘kayako’ API.	XML Di-gest:: LWP: URI, HTTI
Lmsensors				•							
Msmq								•	•	Must be installed on Win-dows. Not de-veloped yet.	

Continued

Table 1.1 – continued from previous page

Category	Check	Transport			Protocol					Experimental	Comm
	SSH		TELNET	WSMAN	SNMP		HTTP		WMI		JMX
Nginx					•					Need ‘Http-Stub-Status-Module’ module.	LWP: URI, HTTP
Pacemaker	•							•		Use ‘crm_mon’ command.	
Pfsense				•							
Selenium								•		Connect to a selenium server to play a scenario.	XML WWW
Tomcat					•					Need tomcat web-manager.	XML LWP: URI, HTTP
Varnish	•							•		Use varnish commands.	
VMWare								•		Need ‘centreon-vmware’ connector from Centreon.	
Pfsense				•							
Protocols	Bgp				•						
	Dhcp								•		
	Dns								•		

Continued

Table 1.1 – continued from previous page

Category	Check	Transport			Protocol					Experimental	Comm
		SSH	TELNET	WSMAN	SNMP		HTTP		WMI		JMX
Database	Ftp								•		
	Http					•					
	Ftp								•		
	Imap								•		
	Jmx							•			
	Ldap								•		
	Ntp								•		
	Radius								•		
	Smtp								•		
	Tcp								•		
	Udp								•		
	x509								•		
	Informix								•		
	Firebird								•		
	MS SQL								•		

Continued

Table 1.1 – continued from previous page

Category	Check	Transport			Protocol					Experimental	Comm
		SSH	TELNET	WSMAN	SNMP		HTTP		WMI		JMX
Hardware	MySQL								•		
	Oracle								•		
	Postgres								•		
	ATS Apc				•					•	
	PDU Apc				•						
	PDU Eaton				•					•	
	PDU Raritan				•						
	Standard Printers				•						
	Hwgste				•						
	Sensorip				•						
	Sensormetrix Em01					•					
	Serverscheck				•						
	Cisco UCS				•						
	Dell CMC				•						
	Dell iDrac				•						

Continued

Table 1.1 – continued from previous page

Category	Check	Transport			Protocol					Experimental	Comm
		SSH	TELNET	WSMAN	SNMP		HTTP		WMI		JMX
	Dell Open-manage				•						Need 'open mana agent on oper-ating syste
	HP Pro-liant				•						Need 'HP Insigh agent on oper-ating syste
	HP Blade Chassis				•						
	IBM Blade-Center				•						
	IBM HMC	•							•	•	
	IBM IMM				•						
	Sun hard-ware	•	•		•				•		Can monit many sun hard-ware.
	UPS APC				•						
	UPS Mge				•						
	UPS Stan-dard				•						
	UPS Power-ware				•						
	3com				•						

	Alcatel Omniswitch Arkoon Aruba Bluecoat Brocade Checkpoint Cisco Citrix Netscaler Dell Powerconnect Dlink Extreme				• • • • • • • • • • • •						Many cisco (2800 Nexu Wlc, Iron-port,..

AIX

-
-

Use AIX commands.

-

Freebsd

-

Need ‘bsnmpd’ agent.

Linux

-
-

Use Linux commands.

-

Solaris
-
-

Use Solaris commands.

-

Windows
-
-
-
-

Dell EqualLogic Storage
-

Dell MD3000

-

Need ‘SMcli’ command.
Dell TL2000

-

Dell ML6000

-

EMC Celerra

-
-

Use appliance commands.
EMC Clariion

-

Need ‘navisphere’ command.
EMC DataDomain

-

EMC Recoverypoint

-
-

Use appliance commands.
EMC Vplex

-

Use the JSON API. JSON, LWP::UserAgent, URI, HTTP::Cookies
EMC Xtremio

-

Use the JSON API. JSON, LWP::UserAgent, URI, HTTP::Cookies
Fujitsu Eternus DX

-
-

Use appliance commands.
HP 3par

-
-

Use appliance commands.
HP Lefthand

-

HP MSA2000

-

HP p2000

-

Use the XML API. XML::XPath, LWP::UserAgent, URI, HTTP::Cookies
IBM DS3000

-

Use ‘SMcli’ command.
IBM DS4000

-

Use ‘SMcli’ command.
IBM DS5000

-

Use ‘SMcli’ command.
IBM TS3100

-

IBM TS3200

-

Netapp

-

DateTime
Nimble

-

Panzura

-

Qnap

-

Synology

-

Violin 3000

-

1.4.2 How can i remove perfdatas ?

For example, i check TCP connections from a linux in SNMP with following command:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=tcpcon --hostname=127.0.0.1 --snmp
OK: Total connections: 1 | 'total'=1;;;0; 'con_closed'=0;;;0; 'con_closeWait'=0;;;0; '  con_synSent'=
```

There are too many perfdatas and i want to keep 'total' perfdata only. I use the option `--filter-perfdata='total'`:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=tcpcon --hostname=127.0.0.1 --snmp
OK: Total connections: 1 | 'total'=1;;;0;
```

I can use regexp in `--filter-perfdata` option. So, i can exclude perfdata beginning by 'total':

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=tcpcon --hostname=127.0.0.1 --snmp
OK: Total connections: 1 | 'con_closed'=0;;;0; 'con_closeWait'=0;;;0; 'con_synSent'=0;;;0; 'con_estab
```

1.4.3 How can i set threshold: critical if value < X ?

"centreon-plugins" can manage Nagios threshold ranges: https://nagios-plugins.org/doc/guidelines.html#THRESHOLDFORMAT

For example, i want to check that 'crond' is running (if there is less than 1 process, critical). I have two ways:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=processcount --hostname=127.0.0.1
CRITICAL: Number of current processes running: 0 | 'nbproc'=0;;1:;0;
```

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=processcount --hostname=127.0.0.1
CRITICAL: Number of current processes running: 0 | 'nbproc'=0;;@0:0;0;
```

1.4.4 How can i check a generic SNMP OID value ?

There is a generic SNMP plugin to check it. An example to get 'SysUptime' SNMP OID:

```
$ perl centreon_plugins.pl --plugin=apps::protocols::snmp::plugin --mode=numeric-value --oid='.1.3.6.
```

1.4.5 How can i check ipv6 equipment in SNMP ?

To check ipv6 equipment, use the following syntax (`udp6:[xxxx]`):

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --hostname='udp6:[fe80::250:56ff:feb5:6ae
```

1.4.6 How to use memcached server for retention datas ?

Some plugins need to store datas. Two ways to store it:

- File on a disk (by default).
- Memcached server.

To use 'memcached', you must have a memcached server and the CPAN 'Memcached::libmemcached' module installed. You can set the memcached server with the option `--memcached`:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=interfaces --hostname=127.0.0.1 --
OK: All traffic are ok | 'traffic_in_lo'=197.40b/s;;;0;10000000 'traffic_out_lo'=197.40b/s;;;0;10000
Interface 'lo' Traffic In : 197.40b/s (0.00 %), Out : 197.40b/s (0.00 %)
Interface 'eth0' Traffic In : 14.54Kb/s (0.00 %), Out : 399.59b/s (0.00 %)
Interface 'eth1' Traffic In : 13.88Kb/s (0.00 %), Out : 1.69Kb/s (0.00 %)
```

Tip: Local file is used if the memcached server is not responding.

1.4.7 What does `--dyn-mode` option do ?

With the option, you can used a mode with a plugin. It commonly used for database checks. For example, i have an application which stores some monitoring information on a database. The developer can use another plugin to create the check (no need to do the SQL connections,... It saves time):

```
$ perl centreon_plugins.pl --plugin=database::mysql::plugin --dyn-mode=apps::centreon::mysql::mode::p
OK: All poller delay for last update are ok | 'delay_Central'=2s;0:300;0:600;0; 'delay_Poller-Engine'
Delay for last update of Central is 2 seconds
Delay for last update of Poller-Engine is 2 seconds
```

Warning: A mode using the following system must notice it (in the help description). So you should open the file with an editor and read at the end the description.

1.4.8 How can i check the plugin version ?

You can check the version of plugins and modes with option `--version`:

```
$ perl centreon_plugins.pl --version
Global Version: 20160524
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --version
Plugin Version: 0.1
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=storage --version
Mode Version: 1.0
```

You can also use the option `--mode-version` to execute the mode only if there is the good version. For example, we want to execute the mode only if the version >= 2.x:

```
$  perl  centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=storage --hostname=127.0.0.1 --s
UNKNOWN: Not good version for plugin mode. Excepted at least: 2.x. Get: 1.0
```

1.4.9 Can i have one standalone Perl file ?

We have done some tests and it will cost around 4% more of execution time. We are going to create a standalone Linux SNMP plugin.

Download the Perl module `App::FatPacker` on metacpan:

```
# tar zxvf App-FatPacker-0.010005.tar.gz
# cd App-FatPacker-0.010005
# perl Makefile.PL && make && make install
```

Create a directory to build it:

```
# mkdir -p build/plugin
# cd build
```

Clone `centreon-plugins`:

```
# git clone https://github.com/centreon/centreon-plugins.git
```

`fatpack` includes `pm` files under the directory `lib`:

```
# mkdir plugin/lib && cd centreon-plugins
```

Copy the common files for all plugins:

```
# find . -name "*.pm" -exec sed -i ' /__END__/d' \{\} \;
# cp -R --parent centreon/plugins/{misc,mode,options,output,perfdata,script,statefile,values}.pm cent
# cp centreon_plugins.pl ../plugin
# sed -i 's/alternative_fatpacker = 0/alternative_fatpacker = 1/' ../plugin/lib/centreon/plugins/scri
```

Copy files for Linux SNMP plugin:

```
# cp -R --parent centreon/plugins/{script_snmp,snmp}.pm os/linux/snmp/ snmp_standard/mode/{cpu,cpudet
```

Build the standalone Perl file:

```
# cd ../plugin
# fatpack file centreon_plugins.pl > centreon_linux_snmp.pl
```

The plugin works in the same way:

```
# perl centreon_linux_snmp.pl --plugin=os::linux::snmp::plugin --mode=processcount --snmp-community p
```

1.4.10 Howto build a standalone Windows executable ?

This is only useful if you want to compile your own `centreon_plugins.exe`. You won't need to install Perl on your windows server.

- Install on Windows Strawberry Perl 5.24.x (Download on http://strawberryperl.com/)
- Trunk of centreon-plugins repository (Download on https://github.com/centreon/centreon-plugins/archive/master.zip)

Once everything is installed, install CPAN Module `PAR::Packer` (replace `<PERL_INSTALL_DIR>`):

```
cmd> <PERL_INSTALL_DIR>\perl\bin\cpan.bat
cpan> install PAR::Packer
```

It can take several minutes to install the CPAN Module.

In the parent directory containing the directory `centreon-plugins`, create a `build.bat` file (replace `<PERL_INSTALL_DIR>`). We exclude the module `IO::Socket::INET6` (Perl 5.14 has the full set of IPv6 functions as part of its core Socket module).

```
set PERL_INSTALL_DIR=<PERL_INSTALL_DIR>

chdir /d %~dp0
set PAR_VERBATIM=1

cmd /C %PERL_INSTALL_DIR%\perl\site\bin\pp --lib=centreon-plugins\ -o centreon_plugins.exe centreon-p
--unicode ^
-X IO::Socket::INET6 ^
--link=%PERL_INSTALL_DIR%\c\bin\libxml2-2__.dll ^
--link=%PERL_INSTALL_DIR%\c\bin\libiconv-2__.dll ^
--link=%PERL_INSTALL_DIR%\c\bin\liblzma-5__.dll ^
--link=%PERL_INSTALL_DIR%\c\bin\zlib1__.dll ^
-M Win32::Job ^
-M centreon::plugins::script ^
-M centreon::plugins::alternative::Getopt ^
-M apps::backup::netbackup::local::plugin ^
-M apps::backup::netbackup::local::mode::dedupstatus ^
-M apps::backup::netbackup::local::mode::drivecleaning ^
-M apps::backup::netbackup::local::mode::drivestatus ^
-M apps::backup::netbackup::local::mode::jobstatus ^
-M apps::backup::netbackup::local::mode::listpolicies ^
-M apps::backup::netbackup::local::mode::tapeusage ^
-M apps::activedirectory::local::plugin ^
-M apps::activedirectory::local::mode::dcdiag ^
-M apps::activedirectory::local::mode::netdom ^
-M apps::citrix::local::plugin ^
-M apps::citrix::local::mode::license ^
-M apps::citrix::local::mode::session ^
-M apps::citrix::local::mode::zone ^
-M apps::citrix::local::mode::folder ^
-M apps::iis::local::plugin ^
-M apps::iis::local::mode::listapplicationpools ^
-M apps::iis::local::mode::applicationpoolstate ^
-M apps::iis::local::mode::listsites ^
-M apps::iis::local::mode::webservicestatistics ^
-M apps::exchange::2010::local::plugin ^
-M apps::exchange::2010::local::mode::activesyncmailbox ^
-M apps::exchange::2010::local::mode::databases ^
-M apps::exchange::2010::local::mode::listdatabases ^
-M apps::exchange::2010::local::mode::imapmailbox ^
-M apps::exchange::2010::local::mode::mapimailbox ^
-M apps::exchange::2010::local::mode::outlookwebservices ^
-M apps::exchange::2010::local::mode::owamailbox ^
-M apps::exchange::2010::local::mode::queues ^
-M apps::exchange::2010::local::mode::replicationhealth ^
-M apps::exchange::2010::local::mode::services ^
-M centreon::common::powershell::exchange::2010::powershell ^
```

```
-M apps::cluster::mscs::local::plugin ^
-M apps::cluster::mscs::local::mode::listnodes ^
-M apps::cluster::mscs::local::mode::listresources ^
-M apps::cluster::mscs::local::mode::networkstatus ^
-M apps::cluster::mscs::local::mode::nodestatus ^
-M apps::cluster::mscs::local::mode::resourcestatus ^
-M apps::cluster::mscs::local::mode::resourcegroupstatus ^
-M os::windows::local::plugin ^
-M os::windows::local::mode::ntp ^
-M os::windows::local::mode::rdpsessions ^
-M storage::dell::compellent::local::plugin ^
-M storage::dell::compellent::local::mode::hbausage ^
-M storage::dell::compellent::local::mode::volumeusage ^
--verbose

pause
```

Add plugins and modes you need in `centreon_plugins.exe` (the example add some plugins). Eventually, execute `build.bat` file to create executable `centreon_plugins.exe`.

If you want to change the executable version and ico file, add following code after `PERL_INSTALL_DIR` first line:

```
set ICO_FILE=centreon.ico
set RC_FILE=centreon.rc

chdir /d %~dp0

for /f "tokens=4 delims= " %%i in ('type centreon-plugins\centreon\plugins\script.pm ^| findstr globa
set VERSION_PLUGIN=%VERSION_PLUGIN:~0,8%

(
echo #define PP_MANIFEST_FILEFLAGS 0
echo #include ^<windows.h^>
echo.
echo CREATEPROCESS_MANIFEST_RESOURCE_ID RT_MANIFEST "winres\\pp.manifest"
echo.
echo VS_VERSION_INFO VERSIONINFO
echo    FILEVERSION        0,0,0,0
echo    PRODUCTVERSION     0,0,0,0
echo    FILEFLAGSMASK      VS_FFI_FILEFLAGSMASK
echo    FILEFLAGS          PP_MANIFEST_FILEFLAGS
echo    FILEOS             VOS_NT_WINDOWS32
echo    FILETYPE           VFT_APP
echo    FILESUBTYPE        VFT2_UNKNOWN
echo BEGIN
echo    BLOCK "StringFileInfo"
echo    BEGIN
echo        BLOCK "000004B0"
echo        BEGIN
echo            VALUE "CompanyName", "Centreon\0"
echo            VALUE "FileDescription", " \0"
echo            VALUE "FileVersion", "1.0.0.0\0"
echo            VALUE "InternalName", " \0"
echo            VALUE "LegalCopyright", " \0"
echo            VALUE "LegalTrademarks", " \0"
echo            VALUE "OriginalFilename", " \0"
echo            VALUE "ProductName", "centreon-plugins\0"
echo            VALUE "ProductVersion", "%VERSION_PLUGIN%.0\0"
```

```
echo        END
echo    END
echo    BLOCK "VarFileInfo"
echo    BEGIN
echo        VALUE "Translation", 0x00, 0x04B0
echo    END
echo END
echo.
echo WINEXE ICON winres\\pp.ico
)> %RC_FILE%

for /f "delims=" %%i in ('dir /ad /B %PERL_INSTALL_DIR%\cpan\build\PAR-Packer-*') do set "PAR_PACKER_
SET PAR_PACKER_SRC=%PERL_INSTALL_DIR%\cpan\build\%PAR_PACKER_DIRNAME%

copy /Y %ICO_FILE% %PAR_PACKER_SRC%\myldr\winres\pp.ico
copy /Y centreon.rc %PAR_PACKER_SRC%\myldr\winres\pp.rc
del %PAR_PACKER_SRC%\myldr\ppresource.coff
cd /D %PAR_PACKER_SRC%\myldr\ && perl Makefile.PL
cd /D %PAR_PACKER_SRC%\myldr\ && dmake boot.exe
cd /D %PAR_PACKER_SRC%\myldr\ && dmake Static.pm
attrib -R %PERL_INSTALL_DIR%\perl\site\lib\PAR\StrippedPARL\Static.pm
copy /Y %PAR_PACKER_SRC%\myldr\Static.pm %PERL_INSTALL_DIR%\perl\site\lib\PAR\StrippedPARL\Static.pm
```

You can build a 32 bits binary from a Windows 64 bits:

- Install Strawberry Perl 5.24.x 32 bits on Windows (Download on http://strawberryperl.com/)
- Install cpan module `PAR::Packer`
- Add following line in the build script : PATH=%PERL_INSTALL_DIR%cbin;%PERL_INSTALL_DIR%perlbin;C:WindowsSyst

1.5 Troubleshooting

1.5.1 SNMP

I get the SNMP error: 'UNKNOWN:.* (tooBig).*'

The following error can happened with some equipments. You can resolve it if you set following options:

- `--subsetleef=20 --maxrepetitions=20`

I get the SNMP error: 'UNKNOWN:.*Timeout'

The following error means:

- Don't have network access to the target SNMP Server (a firewall can block UDP 161).
- Wrong SNMP community name or SNMP version set.

I get the SNMP error: 'UNKNOWN:.*Cant get a single value'

The following error means: SNMP access is working but you can't retrieve SNMP values. Very possible reasons:

- SNMP value is not set yet (can be happened when a SNMP server is just started).
- SNMP value is not implemented by the constructor.

- SNMP value is set on a specific firmware or OS.

Seems that process check is not working well for some arguments filter

In SNMP, there is a limit in argument length of 128 characters. So, if you try to filter with an argument after 128 characters, it won't work. It can happen with Java arguments. To solve the problem, you should prefer a NRPE check.

Can't access in SNMP v3

First, you need to validate SNMP v3 connection with snmpwalk. When it's working, you set SNMP v3 options in command line. The mapping between 'snmpwalk' options and "centreon-plugins" options:

- -a => `--authprotocol`
- -A => `--authpassphrase`
- -u => `--snmp-username`
- -x => `--privprotocol`
- -X => `--privpassphrase`
- -l => not needed (automatic)
- -e => `--securityengineid`
- -E => `--contextengineid`

1.5.2 Miscellaneous

I get the error: "UNKNOWN: Need to specify '–custommode'."

Some plugins need to set the option `--custommode`. You can know the value to set with the option `--list-custommode`. An example:

```
$ perl centreon_plugins.pl --plugin=storage::ibm::DS3000::cli::plugin --list-custommode
...
Custom Modes Available:
 smcli

$ perl centreon_plugins.pl --plugin=storage::ibm::DS3000::cli::plugin --custommode=smcli --list-mode
```

I get the error: "UNKNOWN: Cannot write statefile .*"

You must create the directory (with write permissions) to let the plugin stores some datas on disk.

I get the error: "UNKNOWN: Cannot load module 'xxx'."

The problem can be:

- A prerequisite CPAN module is missing. You need to install it.
- The CPAN module cannot be loaded because of its path. Perl modules must be installed on some specific paths.

I can't see help messages

"centreon-plugins" files must Unix format (no Windows carriage returns). You can change it with the following command:

```
$ find . -name "*.p[ml]" -type f -exec dos2unix \{\} \;
```

Warning: Execute the command in "centreon-plugins" directory.

1.6 Command Samples

1.6.1 Windows

Check all disks in SNMP

Warning if space used > 80% and critical if space used > 90%:

```
$ perl centreon_plugins.pl --plugin=os::windows::snmp::plugin --mode=storage --hostname=xxx.xxx.xxx.x
OK: All storages are ok. | used_C:'=38623698944B;0:108796887040;0:122396497920;0;135996108800 used_D
Storage 'C:' Total: 126.66 GB Used: 35.97 GB (28.40%) Free: 90.69 GB (71.60%)
Storage 'D:' Total: 126.66 GB Used: 35.97 GB (28.40%) Free: 90.69 GB (71.60%)
```

Warning if space free < 5G and critical if space free < 2G:

```
$ perl centreon_plugins.pl --plugin=os::windows::snmp::plugin --mode=storage --hostname=xxx.xxx.xxx.x
OK: All storages are ok. | 'free_C:'=97372344320B;0:5497558138880;0:2199023255552;0;135996108800 'fre
Storage 'C:' Total: 126.66 GB Used: 35.97 GB (28.40%) Free: 90.69 GB (71.60%)
Storage 'D:' Total: 126.66 GB Used: 35.97 GB (28.40%) Free: 90.69 GB (71.60%)
```

1.6.2 Linux

Check all interface traffics in SNMP

Warning if traffic in/out used > 80% and critical if traffic in/out used > 90%:

```
$ perl centreon_plugins.pl --plugin=os::linux::snmp::plugin --mode=interfaces --hostname=127.0.0.1 --
OK: All traffic are ok | 'traffic_in_lo'=126.58b/s;0.00:8000000.00;0.00:9000000.00;0;10000000 'traffi
Interface 'lo' Traffic In : 126.58b/s (0.00 %), Out : 126.58b/s (0.00 %)
Interface 'eth0' Traffic In : 1.87Kb/s (0.00 %), Out : 266.32b/s (0.00 %)
Interface 'eth1' Traffic In : 976.65b/s (0.00 %), Out : 1.02Kb/s (0.00 %)
```

1.6.3 HTTP Protocol

Check authentification of an application (POST request)

An example for authentification form of `demo.centreon.com`:

```
$ perl centreon_plugins.pl --plugin=apps::protocols::http::plugin --mode=expected-content --hostname=
OK: 'color_UNREACHABLE' is present in content. | 'time'=0.575s;;;0; 'size'=20708B;;;0;
```

1.6.4 Modbus Protocol

Check 3 holding registers

The content of the `modbus.json` file can be set directly in `--config` option (eg. : `--config='{ "selection": { "metric1":{...'`). The `type` attribute can have following values:

- holding (default)
- coils
- discrete
- input

```
{
    "selection":{
        "metric1":{
            "address": 1,
            "quantity": 1,
            "type": "holding",
            "display": true
        },
        "metric2":{
            "address": 2,
            "quantity": 1,
            "type": "holding",
            "display": true
        },
        "metric3":{
            "address": 3,
            "quantity": 1,
            "type": "holding",
            "display": true
        }
    }
}
```

The command result:

```
$ perl centreon_plugins.pl --plugin=apps/protocols/modbus/plugin.pm --mode=numeric-value --tcp-host=1
OK: All metrics are OK | 'metric1'=0;;;; 'metric2'=41291;;;; 'metric3'=42655;;;;
Metric 'metric1' value is '0'
Metric 'metric2' value is '41291'
Metric 'metric3' value is '42655'
```

How to change the formatting output ?

There is a command section to change formatting output globally and also to override for each metrics:

```
{
    "selection":{
        "metric1":{
            "address": 1,
            "quantity": 1,
            "type": "holding",
            "display": true
        },
        "metric2":{
            "address": 2,
            "quantity": 1,
            "type": "holding",
            "display": true
        },
        "metric3":{
            "address": 3,
            "quantity": 1,
            "type": "holding",
            "display": true,
            "formatting": {
                "printf_msg": "Override '%s' value is %.2f",
                "printf_var": "$self->{result_values}->{instance}, $self->{result_values}->{value}"
            }
        }
    },
    "formatting": {
        "printf_msg": "My metric '%s' value is %.2f",
        "printf_var": "$self->{result_values}->{instance}, $self->{result_values}->{value}"
    }
}
```

The command result:

```
$ perl centreon_plugins.pl --plugin=apps/protocols/modbus/plugin.pm --mode=numeric-value --tcp-host=1
OK: All metrics are OK | 'metric1'=0;;;; 'metric2'=41291;;;; 'metric3'=42655;;;;
My Metric 'metric1' value is 0.00
My Metric 'metric2' value is 41291.00
Override 'metric3' value is 42655.00
```

How to average 4 registers ?

We create following average values: `[x1 + x2 / 2 = y] [x3 + x4 / 2 = z]`. With the pattern, you can select the metrics. In our case, we get 4 values in one selection. Selected metric names are: `metrics.0`, `metrics.1`, `metrics.2`, `metrics.3` (order is preserved).

The `aggregation` attribute can have following values:

- avg: returns the average of all the elements.
- sum: returns the numerical sum of all the elements.
- min: returns the entry in elements with the lowest numerical value.
- max: returns the entry in elements with the highest numerical value.

```
{
    "selection":{
        "metrics":{
            "address": 1,
            "quantity": 4,
            "type": "holding",
            "display": false
        }
    },
    "virtualcurve":{
        "avg1":{
            "pattern": "metrics\\.[01]$",
            "aggregation": "avg",
            "unit": "con"
        },
        "avg2":{
            "pattern": "metrics\\.[23]$",
            "aggregation": "avg",
            "unit": "con"
        }
    }
}
```

The command result:

```
$ perl centreon_plugins.pl --plugin=apps/protocols/modbus/plugin.pm --mode=numeric-value --tcp-host=1
OK: Global metrics are OK | 'avg1'=42192con;;;; 'avg2'=40574con;;;;
Metric 'avg1' value is '42192'
Metric 'avg2' value is '40574'
```

Apply a custom calculation

There is a `custom` attribute to applied some change on the value:

```
{
    "selection":{
        "metrics":{
            "address": 1,
            "quantity": 4,
            "type": "holding",
            "display": false
        }
    },
    "virtualcurve":{
        "avg":{
            "aggregation": "avg",
            "custom": " / 10",
            "unit": "con"
        }
    }
}
```

The command result:

```
$ perl centreon_plugins.pl --plugin=apps/protocols/modbus/plugin.pm --mode=numeric-value --tcp-host=1
OK: Metric 'avg' value is '3072.3' | 'avg'=3072.3con;;;;
Metric 'avg' value is '3072.3'
```

1.6.5 Multi-service plugin

This mode allow you to concatenate several result of host and/or service check into one check. Can be used to make some aggregation into logical group or gather information from one Centreon to display in another without checking resources twice.

Design of configuration file

```
{
    "mode":"sqlmatching",
    "selection":{
        "ESX":{
            "host_name_filter":"%clus-esx-n%",
            "service_name_filter":"Esx-Status"
        },
        "XIVO":{
            "host_name_filter":"%xivo%",
            "service_name_filter":"Ping"
        }
    },
    "counters":{
        "totalservices":true,
        "totalhosts":true,
        "groups":true
    },
    "formatting":{
        "groups_global_msg":"Nothing special on groups",
        "host_service_separator":"/",
        "display_details":true
    }
}
```

- mode (mandatory) : can be 'sqlmatching' or 'exactmatch'. Linked to the layout of "selection" bloc
- selection (mandatory) : if we use sqlmatching, we define filters like above, if we use exact match, need to pass a key/value pair corresponding to host/service (example below with two "groups" esx-status/load)

```
"selection":{
    "esx-status":{
        "esx-n1":"Esx-Status",
        "esx-n2":"Esx-Status",
        "esx-n3":"Esx-Status"
    },
    "esx-load":{
        "esx-n1":"Esx-Memory",
        "esx-n2":"Esx-Memory",
        "esx-n3":"Esx-Memory",
        "esx-n1":"Esx-Cpu",
        "esx-n2":"Esx-Cpu",
```

```
        "esx-n3":"Esx-Cpu"
    }
},
```

- counters (optionnal) : Contains three keys to choose which counters we should use and consider (totalservices, totalhosts, groups)
- formatting (optionnal) : Contains three keys, 'groups_global_msg' to define a global OK status message, 'host_service_separator' to define separator used between host and service name in output, 'display_details' to config if plugin should display details of host/service name in the verbose output.

Command line, output, threshold ...

Sample command:

```
/usr/lib/nagios/plugins/centreon_plugins.pl --plugin database::mysql::plugin --dyn-mode apps::centreo
```

Sample output:

```
OK: Hosts state summary [up:4][down:2][unreachable:0] - Services state summary [ok:4][warning:0][crit
```

Perfdatas:

```
'total_host_up'=4;;;0; 'total_host_down'=2;;;0; 'total_host_unreachable'=0;;;0; 'total_host_ok'=4;;;0
```

Verbose mode (with display details set as true):

```
Group 'ESX': HOSTS: [up: 4 (clus-esx-n1.com - clus-esx-n2.com - clus-esx-n3.com - clus-esx-n4.com)][d
Group 'XIVO': HOSTS: [up: 0][down: 2 (srvi-xivo-n1 - srvi-xivo-n2)][unreachable: 0] - SERVICES: [ok:
```

Concerning the threshold, you can use some example below:

```
--critical-total '%{total_down} > 4' --critical-groups '%{instance} eq 'ESX' && %{unknown} > 5'
```

1.6.6 NSClient

You can monitor Windows/Linux system with the Rest API of NSClient. Commands and arguments are the same than NRPE (look the NSClient documentation for more informations):

```
$ perl centreon_plugins.pl --plugin=apps::nsclient::restapi::plugin --mode=query --hostname="10.30.2
OK All 2 drive(s) are ok | '\\?\Volume{7cd2d555-9868-11e7-8199-806e6f6e6963}\ used'=289468416.000B;29
```

CHAPTER 2

Developer guide

2.1 Description

This document introduces the best practices in the development of "centreon-plugins".

As all plugins are written in Perl, "there is more than one way to do it". But to avoid reinventing the wheel, you should first take a look at the "example" directory, you will get an overview of how to build your own plugin and associated modes.

There are 3 chapters:

- *Quick Start*: Howto create file structure.
- *Libraries Reference*: API description.
- *Code Style Guidelines*: Follow it.
- *Model Classes Usage*: description of classes you should use for your plugin.

The lastest version is available on following git repository: https://github.com/centreon/centreon-plugins.git

2.2 Quick Start

2.2.1 Directory creation

First of all, you need to create a directory on the git to store the new plugin.

Root directories are organized by section:

- Application : apps
- Database : database
- Hardware : hardware
- network equipment : network
- Operating System : os
- Storage equipment : storage

According to the monitored object, it exists an organization which can use:

- Type
- Constructor

- Model
- Monitoring Protocol

For example, if you want to add a plugin to monitor Linux by SNMP, you need to create this directory:

```
$ mkdir -p os/linux/snmp
```

You also need to create a "mode" directory for futures modes:

```
$ mkdir os/linux/snmp/mode
```

2.2.2 Plugin creation

Once the directory is created, create the plugin file inside it:

```
$ touch plugin.pm
```

Then, edit plugin.pm to add **license terms** by copying it from an other plugin. Don't forget to put your name at the end of it:

```
# ...
# Authors : <your name> <<your email>>
```

Next, describe your **package** name : it matches your plugin directory.

```
package path::to::plugin;
```

Declare used libraries (**strict** and **warnings** are mandatory). Centreon libraries are described later:

```
use strict;
use warnings;
use base qw(**centreon_library**);
```

The plugin need a **new** constructor to instantiate the object:

```
sub new {
      my ($class, %options) = @_;
      my $self = $class->SUPER::new(package => __PACKAGE__, %options);
      bless $self, $class;

      ...

      return $self;
}
```

Plugin version must be declared in the **new** constructor:

```
$self->{version} = '0.1';
```

Several modes can be declared in the **new** constructor:

```
%{$self->{modes}} = (
                        'mode1'     => '<plugin_path>::mode::mode1',
                        'mode2'     => '<plugin_path>::mode::mode2',
                        ...
                        );
```

Then, declare the module:

```
1;
```

A description of the plugin is needed to generate the documentation:

```
__END__

=head1 PLUGIN DESCRIPTION

<Add a plugin description here>.

=cut
```

Tip: You can copy-paste an other plugin.pm and adapt some lines (package, arguments...).

Tip: The plugin has ".pm" extension because it's a Perl module. So don't forget to add **1;** at the end of the file.

2.2.3 Mode creation

Once **plugin.pm** is created and modes are declared in it, create modes in the **mode** directory:

```
cd mode
touch mode1.pm
```

Then, edit mode1.pm to add **license terms** by copying it from an other mode. Don't forget to put your name at the end of it:

```
# ...
# Authors : <your name> <<your email>>
```

Next, describe your **package** name: it matches your mode directory.

```
package path::to::plugin::mode::mode1;
```

Declare used libraries (always the same):

```
use strict;
use warnings;
use base qw(centreon::plugins::mode);
```

The mode needs a **new** constructor to instantiate the object:

```
sub new {
      my ($class, %options) = @_;
      my $self = $class->SUPER::new(package => __PACKAGE__, %options);
      bless $self, $class;

      ...

      return $self;
}
```

Mode version must be declared in the **new** constructor:

```
$self->{version} = '1.0';
```

Several options can be declared in the **new** constructor:

```
$options{options}->add_options(arguments =>
                               {
                                 "option1:s" => { name => 'option1' },
                                 "option2:s" => { name => 'option2', default => 'value1' },
                                 "option3"   => { name => 'option3' },
                               });
```

Here is the description of arguments used in this example:

- option1 : String value
- option2 : String value with default value "value1"
- option3 : Boolean value

Tip: You can have more informations about options format here: http://perldoc.perl.org/Getopt/Long.html

---

The mode need a **check_options** method to validate options:

```
sub check_options {
  my ($self, %options) = @_;
  $self->SUPER::init(%options);
  ...
}
```

For example, Warning and Critical thresholds must be validate in **check_options** method:

```
if (($self->{perfdata}->threshold_validate(label => 'warning', value => $self->{option_results}->{wa
     $self->{output}->add_option_msg(short_msg => "Wrong warning threshold '" . $self->{option_result
     $self->{output}->option_exit();
}
if (($self->{perfdata}->threshold_validate(label => 'critical', value => $self->{option_results}->{c
     $self->{output}->add_option_msg(short_msg => "Wrong critical threshold '" . $self->{option_resul
     $self->{output}->option_exit();
}
```

In this example, help is printed if thresholds do not have a correct format.

Then comes the **run** method, where you perform measurement, check thresholds, display output and format performance datas. This is an example to check a SNMP value:

```
sub run {
  my ($self, %options) = @_;
  $self->{snmp} = $options{snmp};
  $self->{hostname} = $self->{snmp}->get_hostname();

  my $result = $self->{snmp}->get_leef(oids => [$self->{option_results}->{oid}], nothing_quit => 1);
  my $value = $result->{$self->{option_results}->{oid}};

  my $exit = $self->{perfdata}->threshold_check(value => $value,
                               threshold => [ { label => 'critical', 'exit_litteral' => 'critical' }, 
  $self->{output}->output_add(severity => $exit,
                              short_msg => sprintf("SNMP Value is %s.", $value));

  $self->{output}->perfdata_add(label => 'value', unit => undef,
                                value => $value,
                                warning => $self->{perfdata}->get_perfdata_for_output(label => 'warni
                                critical => $self->{perfdata}->get_perfdata_for_output(label => 'crit
                                min => undef, max => undef);

  $self->{output}->display();
  $self->{output}->exit();
}
```

In this example, we check a SNMP OID that we compare to warning and critical thresholds. There are the methods which we use:

- get_leef : get a SNMP value from an OID
- threshold_check : compare SNMP value to warning and critical thresholds
- output_add : add output
- perfdata_add : add perfdata to output
- display : display output
- exit : exit

Then, declare the module:

```
1;
```

A description of the mode and its arguments is needed to generate the documentation:

```
__END__

=head1 PLUGIN DESCRIPTION

<Add a plugin description here>.

=cut
```

2.2.4 Commit and push

Before committing the plugin, you need to create an **enhancement ticket** on the centreon-plugins forge : http://forge.centreon.com/projects/centreon-plugins

Once plugin and modes are developed, you can commit (commit messages in english) and push your work:

```
git add path/to/plugin
git commit -m "Add new plugin for XXXX refs #<ticked_id>"
git push
```

2.3 Libraries reference

This chapter describes Centreon libraries which you can use in your development.

2.3.1 Output

This library allows you to build output of your plugin.

output_add

Description

Add string to output (print it with **display** method). If status is different than 'ok', output associated with 'ok' status is not printed.

Parameters

Parameter	Type	Default	Description
severity	String	OK	Status of the output.
separator	String	-	Separator between status and output string.
short_msg	String		Short output (first line).
long_msg	String		Long output (used with –verbose option).

Example

This is an example of how to manage output:

```
$self->{output}->output_add(severity  => 'OK',
                            short_msg => 'All is ok');
$self->{output}->output_add(severity  => 'Critical',
                            short_msg => 'There is a critical problem');
$self->{output}->output_add(long_msg  => 'Port 1 is disconnected');

$self->{output}->display();
```

Output displays :

```
CRITICAL - There is a critical problem
Port 1 is disconnected
```

perfdata_add

Description

Add performance data to output (print it with **display** method). Performance data are displayed after 'l'.

Parameters

Parameter	Type	Default	Description
label	String		Label of the performance data.
value	Int		Value of the performance data.
unit	String		Unit of the performance data.
warning	String		Warning threshold.
critical	String		Critical threshold.
min	Int		Minimum value of the performance data.
max	Int		Maximum value of the performance data.

Example

This is an example of how to add performance data:

```
$self->{output}->output_add(severity  => 'OK',
                            short_msg => 'Memory is ok');
$self->{output}->perfdata_add(label    => 'memory_used',
                              value    => 30000000,
                              unit     => 'B',
                              warning  => '80000000',
                              critical => '90000000',
                              min      => 0,
                              max      => 100000000);

$self->{output}->display();
```

Output displays:

```
OK - Memory is ok | 'memory_used'=30000000B;80000000;90000000;0;100000000
```

2.3.2 Perfdata

This library allows you to manage performance data.

get_perfdata_for_output

Description

Manage thresholds of performance data for output.

Parameters

Parameter	Type	Default	Description
label	String		Threshold label.
total	Int		Percent threshold to transform in global.
cast_int	Int (0 or 1)		Cast absolute to int.
op	String		Operator to apply to start/end value (uses with 'value').
value	Int		Value to apply with 'op' option.

Example

This is an example of how to manage performance data for output:

```
my $format_warning_perfdata  = $self->{perfdata}->get_perfdata_for_output(label => 'warning', total =
my $format_critical_perfdata = $self->{perfdata}->get_perfdata_for_output(label => 'critical', total

$self->{output}->perfdata_add(label    => 'memory_used',
                              value    => 30000000,
                              unit     => 'B',
                              warning  => $format_warning_perfdata,
                              critical => $format_critical_perfdata,
                              min      => 0,
                              max      => 1000000000);
```

Tip: In this example, instead of print warning and critical thresholds in 'percent', the function calculates and prints these in 'bytes'.

---

threshold_validate

Description

Validate and affect threshold to a label.

Parameters

Parameter	Type	Default	Description
label	String		Threshold label.
value	String		Threshold value.

Example

This example checks if warning threshold is correct:

```
if (($self->{perfdata}->threshold_validate(label => 'warning', value => $self->{option_results}->{wa
  $self->{output}->add_option_msg(short_msg => "Wrong warning threshold '" . $self->{option_results}-
  $self->{output}->option_exit();
}
```

Tip: You can see the correct threshold format here: https://nagios-plugins.org/doc/guidelines.html#THRESHOLDFORMAT

threshold_check

Description

Check performance data value with threshold to determine status.

Parameters

Parameter	Type	Default	Description
value	Int		Performance data value to compare.
threshold	String array		Threshold label to compare and exit status if reached.

Example

This example checks if performance data reached thresholds:

```
$self->{perfdata}->threshold_validate(label => 'warning', value => 80);
$self->{perfdata}->threshold_validate(label => 'critical', value => 90);
my $prct_used = 85;

my $exit = $self->{perfdata}->threshold_check(value => $prct_used, threshold => [ { label => 'critica

$self->{output}->output_add(severity  => $exit,
                            short_msg => sprint("Used memory is %i%%", $prct_used));
$self->{output}->display();
```

Output displays:

```
WARNING - Used memory is 85% |
```

change_bytes

Description

Convert bytes to human readable unit. Return value and unit.

Parameters

Parameter	Type	Default	Description
value	Int		Performance data value to convert.
network		1024	Unit to divide (1000 if defined).

Example

This example change bytes to human readable unit:

```
my ($value, $unit) = $self->{perfdata}->change_bytes(value => 100000);

print $value.' '.$unit."\n";
```

Output displays:

```
100 KB
```

2.3.3 Snmp

This library allows you to use SNMP protocol in your plugin. To use it, add the following line at the beginning of your **plugin.pm**:

```
use base qw(centreon::plugins::script_snmp);
```

get_leef

Description

Return hash table table of SNMP values for multiple OIDs (do not work with SNMP table).

Parameters

Parameter	Type	Default	Description
oids	String array		Array of OIDs to check (Can be set by 'load' method).
dont_quit	Int (0 or 1)	0	Don't quit even if an snmp error occured.
nothing_quit	Int (0 or 1)	0	Quit if no value is returned.

Example

This is an example of how to get 2 SNMP values:

```
my $oid_hrSystemUptime = '.1.3.6.1.2.1.25.1.1.0';
my $oid_sysUpTime = '.1.3.6.1.2.1.1.3.0';

my $result = $self->{snmp}->get_leef(oids => [ $oid_hrSystemUptime, $oid_sysUpTime ], nothing_quit =>

print $result->{$oid_hrSystemUptime}."\n";
print $result->{$oid_sysUpTime}."\n";
```

load

Description

Load a range of OIDs to use with **get_leef** method.

Parameters

Parameter	Type	Default	Description
oids	String array		Array of OIDs to check.
instances	Int array		Array of OID instances to check.
instance_regexp	String		Regular expression to get instances from **instances** option.
begin	Int		Instance to begin
end	Int		Instance to end

Example

This is an example of how to get 4 instances of a SNMP table by using **load** method:

```
my $oid_dskPath = '.1.3.6.1.4.1.2021.9.1.2';

$self->{snmp}->load(oids => [$oid_dskPercentNode], instances => [1,2,3,4]);

my $result = $self->{snmp}->get_leef(nothing_quit => 1);

use Data::Dumper;
print Dumper($result);
```

This is an example of how to get multiple instances dynamically (memory modules of Dell hardware) by using **load** method:

```
my $oid_memoryDeviceStatus = '.1.3.6.1.4.1.674.10892.1.1100.50.1.5';
my $oid_memoryDeviceLocationName = '.1.3.6.1.4.1.674.10892.1.1100.50.1.8';
my $oid_memoryDeviceSize = '.1.3.6.1.4.1.674.10892.1.1100.50.1.14';
my $oid_memoryDeviceFailureModes = '.1.3.6.1.4.1.674.10892.1.1100.50.1.20';

my $result = $self->{snmp}->get_table(oid => $oid_memoryDeviceStatus);
$self->{snmp}->load(oids => [$oid_memoryDeviceLocationName, $oid_memoryDeviceSize, $oid_memoryDeviceF
                    instances => [keys %$result],
                    instance_regexp => '(\d+\.\d+)$');

my $result2 = $self->{snmp}->get_leef();

use Data::Dumper;
print Dumper($result2);
```

get_table

Description

Return hash table of SNMP values for SNMP table.

Parameters

Parameter	Type	Default	Description
oid	String		OID of the snmp table to check.
start	Int		First OID to check.
end	Int		Last OID to check.
dont_quit	Int (0 or 1)	0	Don't quit even if an SNMP error occured.
nothing_quit	Int (0 or 1)	0	Quit if no value is returned.
return_type	Int (0 or 1)	0	Return a hash table with one level instead of multiple.

Example

This is an example of how to get a SNMP table:

```
my $oid_rcDeviceError            = '.1.3.6.1.4.1.15004.4.2.1';
my $oid_rcDeviceErrWatchdogReset = '.1.3.6.1.4.1.15004.4.2.1.2.0';

my $results = $self->{snmp}->get_table(oid => $oid_rcDeviceError, start => $oid_rcDeviceErrWatchdogRe

use Data::Dumper;
print Dumper($results);
```

get_multiple_table

Description

Return hash table of SNMP values for multiple SNMP tables.

Parameters

Parameter	Type	Default	Description
oids	Hash table		Hash table of OIDs to check (Can be set by 'load' method). Keys can be: "oid", "start", "end".
dont_quit	Int (0 or 1)	0	Don't quit even if an SNMP error occured.
nothing_quit	Int (0 or 1)	0	Quit if no value is returned.
return_type	Int (0 or 1)	0	Return a hash table with one level instead of multiple.

Example

This is an example of how to get 2 SNMP tables:

```
my $oid_sysDescr        = ".1.3.6.1.2.1.1.1";
my $aix_swap_pool       = ".1.3.6.1.4.1.2.6.191.2.4.2.1";

my $results = $self->{snmp}->get_multiple_table(oids => [
```

```
                                                    { oid => $aix_swap_pool, start => 1 },
                                                    { oid => $oid_sysDescr },
                                                ]);

use Data::Dumper;
print Dumper($results);
```

get_hostname

Description

Get hostname parameter (useful to get hostname in mode).

Parameters

None.

Example

This is an example of how to get hostname parameter:

```
my $hostname = $self->{snmp}->get_hostname();
```

get_port

Description

Get port parameter (useful to get port in mode).

Parameters

None.

Example

This is an example of how to get port parameter:

```
my $port = $self->{snmp}->get_port();
```

oid_lex_sort

Description

Return sorted OIDs.

Parameters

Parameter	Type	Default	Description
-	String array		Array of OIDs to sort.

Example

This example prints sorted OIDs:

```
foreach my $oid ($self->{snmp}->oid_lex_sort(keys %{$self->{results}->{$my_oid}})) {
  print $oid;
}
```

2.3.4 Misc

This library provides a set of miscellaneous methods. To use it, you can directly use the path of the method:

```
centreon::plugins::misc::<my_method>;
```

trim

Description

Strip whitespace from the beginning and end of a string.

Parameters

Parameter	Type	Default	Description
-	String		String to strip.

Example

This is an example of how to use **trim** method:

```
my $word = '  Hello world !  ';
my $trim_word =  centreon::plugins::misc::trim($word);

print $word."\n";
print $trim_word."\n";
```

Output displays :

```
Hello world !
```

change_seconds

Description

Convert seconds to human readable text.

Parameters

Parameter	Type	Default	Description
-	Int		Number of seconds to convert.

Example

This is an example of how to use **change_seconds** method:

```
my $seconds = 3750;
my $human_readable_time =  centreon::plugins::misc::change_seconds($seconds);

print 'Human readable time : '.$human_readable_time."\n";
```

Output displays:

```
Human readable time : 1h 2m 30s
```

backtick

Description

Execute system command.

Parameters

Parameter	Type	Default	Description
command	String		Command to execute.
arguments	String array		Command arguments.
timeout	Int	30	Command timeout.
wait_exit	Int (0 or 1)	0	Command process ignore SIGCHLD signals.
redirect_stderr	Int (0 or 1)	0	Print errors in output.

Example

This is an example of how to use **backtick** method:

```
my ($error, $stdout, $exit_code) = centreon::plugins::misc::backtick(
                                    command => 'ls /home',
                                    timeout => 5,
                                    wait_exit => 1
                                    );
```

```
print $stdout."\n";
```

Output displays files in '/home' directory.

execute

Description

Execute command remotely.

Parameters

Parameter	Type	Default	Description
output	Object		Plugin output ($self->{output}).
options	Object		Plugin options ($self->{option_results}) to get remote options.
sudo	String		Use sudo command.
command	String		Command to execute.
command_path	String		Command path.
command_options	String		Command arguments.

Example

This is an example of how to use **execute** method. We suppose `--remote` option is enabled:

```
my $stdout = centreon::plugins::misc::execute(output => $self->{output},
                                               options => $self->{option_results},
                                               sudo => 1,
                                               command => 'ls /home',
                                               command_path => '/bin/',
                                               command_options => '-l');
```

Output displays files in /home using ssh on a remote host.

windows_execute

Description

Execute command on Windows.

Parameters

Parameter	Type	Default	Description
output	Object		Plugin output ($self->{output}).
command	String		Command to execute.
command_path	String		Command path.
command_options	String		Command arguments.
timeout	Int		Command timeout.
no_quit	Int		Don't quit even if an error occured.

Example

This is an example of how to use **windows_execute** method.

```
my $stdout = centreon::plugins::misc::windows_execute(output => $self->{output},
                                                      timeout => 10,
                                                      command => 'ipconfig',
                                                      command_path => '',
                                                      command_options => '/all');
```

Output displays IP configuration on a Windows host.

2.3.5 Statefile

This library provides a set of methods to use a cache file. To use it, add the following line at the beginning of your **mode**:

```
use centreon::plugins::statefile;
```

read

Description

Read cache file.

Parameters

Parameter	Type	Default	Description
statefile	String		Name of the cache file.
statefile_dir	String		Directory of the cache file.
memcached	String		Memcached server to use.

Example

This is an example of how to use **read** method:

```
$self->{statefile_value} = centreon::plugins::statefile->new(%options);
$self->{statefile_value}->check_options(%options);
$self->{statefile_value}->read(statefile => 'my_cache_file',
                               statefile_dir => '/var/lib/centreon/centplugins'
                              );

use Data::Dumper;
print Dumper($self->{statefile_value});
```

Output displays cache file and its parameters.

get

Description

Get data from cache file.

Parameters

Parameter	Type	Default	Description
name	String		Get a value from cache file.

Example

This is an example of how to use **get** method:

```
$self->{statefile_value} = centreon::plugins::statefile->new(%options);
$self->{statefile_value}->check_options(%options);
$self->{statefile_value}->read(statefile => 'my_cache_file',
                               statefile_dir => '/var/lib/centreon/centplugins'
                              );

my $value = $self->{statefile_value}->get(name => 'property1');
print $value."\n";
```

Output displays value for 'property1' of the cache file.

write

Description

Write data to cache file.

Parameters

Parameter	Type	Default	Description
data	String		Data to write in cache file.

Example

This is an example of how to use **write** method:

```
$self->{statefile_value} = centreon::plugins::statefile->new(%options);
$self->{statefile_value}->check_options(%options);
$self->{statefile_value}->read(statefile => 'my_cache_file',
                               statefile_dir => '/var/lib/centreon/centplugins'
                              );

my $new_datas = {};
```

```
$new_datas->{last_timestamp} = time();
$self->{statefile_value}->write(data => $new_datas);
```

Then, you can read the result in '/var/lib/centreon/centplugins/my_cache_file', timestamp is written in it.

2.3.6 HTTP

This library provides a set of methodss to use HTTP protocol. To use it, add the following line at the beginning of your **mode**:

```
use centreon::plugins::http;
```

Some options must be set in **plugin.pm**:

Option	Type	Description
hostname	String	IP Addr/FQDN of the webserver host.
port	String	HTTP port.
proto	String	Used protocol ('http' or 'https').
credentials		Use credentials.
ntlm		Use NTLM authentication (if `--credentials` is used).
username	String	Username (if `--credentials` is used).
password	String	User password (if `--credentials` is used).
proxyurl	String	Proxy to use.
url_path	String	URL to connect (start to '/').

connect

Description

Test a connection to an HTTP url. Return content of the webpage.

Parameters

This method use plugin options previously defined.

Example

This is an example of how to use **connect** method. We suppose these options are defined : * –hostname = 'google.com' * –urlpath = '/' * –proto = 'http' * –port = 80

```
$self->{http} = centreon::plugins::http->new(output => $self->{output});
$self->{http}->set_options(%{$self->{option_results}});
my $webcontent = $self->{http}->request();
print $webcontent;
```

Output displays content of the webpage 'http://google.com/'.

2.3.7 DBI

This library allows you to connect to databases. To use it, add the following line at the beginning of your **plugin.pm**:

```
use base qw(centreon::plugins::script_sql);
```

connect

Description

Connect to databases.

Parameters

Parameter	Type	Default	Description
dontquit	Int (0 or 1)	0	Don't quit even if errors occured.

Example

This is an example of how to use **connect** method. The format of the connection string can have the following forms:

```
DriverName:database_name
DriverName:database_name@hostname:port
DriverName:database=database_name;host=hostname;port=port
```

In plugin.pm:

```
$self->{sqldefault}->{dbi} = ();
$self->{sqldefault}->{dbi} = { data_source => 'mysql:host=127.0.0.1;port=3306' };
```

In your mode:

```
$self->{sql} = $options{sql};
my ($exit, $msg_error) = $self->{sql}->connect(dontquit => 1);
```

Then, you are connected to the MySQL database.

query

Description

Send query to database.

Parameters

Parameter	Type	Default	Description
query	String		SQL query to send.

Example

This is an example of how to use **query** method:

```
$self->{sql}->query(query => q{SHOW /*!50000 global */ STATUS LIKE 'Slow_queries'});
my ($name, $result) = $self->{sql}->fetchrow_array();

print 'Name : '.$name."\n";
print 'Value : '.$value."\n";
```

Output displays count of MySQL slow queries.

fetchrow_array

Description

Return Array from sql query.

Parameters

None.

Example

This is an example of how to use **fetchrow_array** method:

```
$self->{sql}->query(query => q{SHOW /*!50000 global */ STATUS LIKE 'Uptime'});
my ($dummy, $result) = $self->{sql}->fetchrow_array();

print 'Uptime : '.$result."\n";
```

Output displays MySQL uptime.

fetchall_arrayref

Description

Return Array from SQL query.

Parameters

None.

Example

This is an example of how to use **fetchrow_array** method:

```
$self->{sql}->query(query => q{
      SELECT SUM(DECODE(name, 'physical reads', value, 0)),
          SUM(DECODE(name, 'physical reads direct', value, 0)),
          SUM(DECODE(name, 'physical reads direct (lob)', value, 0)),
          SUM(DECODE(name, 'session logical reads', value, 0))
      FROM sys.v_$sysstat
});
my $result = $self->{sql}->fetchall_arrayref();

my $physical_reads = @$result[0]->[0];
my $physical_reads_direct = @$result[0]->[1];
my $physical_reads_direct_lob = @$result[0]->[2];
my $session_logical_reads = @$result[0]->[3];

print $physical_reads."\n";
```

Output displays physical reads on Oracle database.

fetchrow_hashref

Description

Return Hash table from SQL query.

Parameters

None.

Example

This is an example of how to use **fetchrow_hashref** method:

```
$self->{sql}->query(query => q{
  SELECT datname FROM pg_database
});

while ((my $row = $self->{sql}->fetchrow_hashref())) {
  print $row->{datname}."\n";
}
```

Output displays Postgres databases.

Output displays Postgres databases.

2.4 Complete examples

2.4.1 Simple SNMP request

Description

This example explains how to check a single SNMP value on a PfSense firewall (memory dropped packets).
We use cache file because it's a SNMP counter. So we need to get the value between 2 checks.
We get the value and compare it to warning and critical thresholds.

Plugin file

First, create the plugin directory and the plugin file:

```
$ mkdir -p apps/pfsense/snmp
$ touch apps/pfsense/snmp/plugin.pm
```

Tip: PfSense is a firewall application and we check it using SNMP protocol

Then, edit **plugin.pm** and add the following lines:

```
#
# Copyright 2018 Centreon (http://www.centreon.com/)
#
# Centreon is a full-fledged industry-strength solution that meets
# the needs in IT infrastructure and application monitoring for
# service performance.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Path to the plugin
package apps::pfsense::snmp::plugin;

# Needed libraries
use strict;
use warnings;
# Use this library to check using SNMP protocol
use base qw(centreon::plugins::script_snmp);
```

Tip: Don't forget to edit 'Authors' line.

Add **new** method to instantiate the plugin:

```
sub new {
  my ($class, %options) = @_;
  my $self = $class->SUPER::new(package => __PACKAGE__, %options);
  bless $self, $class;
  # $options->{options} = options object

  # Plugin version
  $self->{version} = '0.1';

  # Modes association
  %{$self->{modes}} = (
                        # Mode name => path to the mode
                        'memory-dropped-packets'    => 'apps::pfsense::snmp::mode::memorydroppedpackets
                        );

  return $self;
}
```

Declare this plugin as a perl module:

```
1;
```

Add a description to the plugin:

```
__END__

=head1 PLUGIN DESCRIPTION

Check pfSense in SNMP.

=cut
```

Tip: This description is printed with '–help' option.

Mode file

Then, create the mode directory and the mode file:

```
$ mkdir apps/pfsense/snmp/mode
$ touch apps/pfsense/snmp/mode/memorydroppedpackets.pm
```

Edit **memorydroppedpackets.pm** and add the following lines:

```
#
# Copyright 2018 Centreon (http://www.centreon.com/)
#
# Centreon is a full-fledged industry-strength solution that meets
# the needs in IT infrastructure and application monitoring for
# service performance.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
```

```
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Path to the plugin
package apps::pfsense::snmp::mode::memorydroppedpackets;

# Needed library for modes
use base qw(centreon::plugins::mode);

# Needed libraries
use strict;
use warnings;

# Custom library
use POSIX;

# Needed library to use cache file
use centreon::plugins::statefile;
```

Add **new** method to instantiate the mode:

```
sub new {
  my ($class, %options) = @_;
  my $self = $class->SUPER::new(package => __PACKAGE__, %options);
  bless $self, $class;

  # Mode version
  $self->{version} = '1.0';

  # Declare options
  $options{options}->add_options(arguments =>
                                {
                                  # option name         => variable name
                                  "warning:s"           => { name => 'warning', },
                                  "critical:s"          => { name => 'critical', },
                                });

  # Instantiate cache file
  $self->{statefile_value} = centreon::plugins::statefile->new(%options);
  return $self;
}
```

Tip: A default value can be added to options. Example : "warning:s" => { name => 'warning', default => '80' },

Add **check_options** method to validate options:

```
sub check_options {
  my ($self, %options) = @_;
```

```
  $self->SUPER::init(%options);

  # Validate threshold options with threshold_validate method
  if (($self->{perfdata}->threshold_validate(label => 'warning', value => $self->{option_results}->{w
     $self->{output}->add_option_msg(short_msg => "Wrong warning threshold '" . $self->{option_result
     $self->{output}->option_exit();
  }
  if (($self->{perfdata}->threshold_validate(label => 'critical', value => $self->{option_results}->{
     $self->{output}->add_option_msg(short_msg => "Wrong critical threshold '" . $self->{option_resul
     $self->{output}->option_exit();
  }

  # Validate cache file options using check_options method of statefile library
  $self->{statefile_value}->check_options(%options);
}
```

Add **run** method to execute mode:

```
sub run {
  my ($self, %options) = @_;
  # $options{snmp} = snmp object

  # Get SNMP options
  $self->{snmp} = $options{snmp};
  $self->{hostname} = $self->{snmp}->get_hostname();
  $self->{snmp_port} = $self->{snmp}->get_port();

  # SNMP oid to request
  my $oid_pfsenseMemDropPackets = '.1.3.6.1.4.1.12325.1.200.1.2.6.0';
  my ($result, $value);

  # Get SNMP value for oid previsouly defined
  $result = $self->{snmp}->get_leef(oids => [ $oid_pfsenseMemDropPackets ], nothing_quit => 1);
  # $result is a hash table where keys are oids
  $value = $result->{$oid_pfsenseMemDropPackets};

  # Read the cache file
  $self->{statefile_value}->read(statefile => 'pfsense_' . $self->{hostname}  . '_' . $self->{snmp_po
  # Get cache file values
  my $old_timestamp = $self->{statefile_value}->get(name => 'last_timestamp');
  my $old_memDropPackets = $self->{statefile_value}->get(name => 'memDropPackets');

  # Create a hash table with new values that will be write to cache file
  my $new_datas = {};
  $new_datas->{last_timestamp} = time();
  $new_datas->{memDropPackets} = $value;

  # Write new values to cache file
  $self->{statefile_value}->write(data => $new_datas);

  # If cache file didn't have any values, create it and wait another check to calculate value
  if (!defined($old_timestamp) || !defined($old_memDropPackets)) {
      $self->{output}->output_add(severity => 'OK',
                                  short_msg => "Buffer creation...");
      $self->{output}->display();
      $self->{output}->exit();
  }
```

```
  # Fix when PfSense reboot (snmp counters initialize to 0)
  $old_memDropPackets = 0 if ($old_memDropPackets > $new_datas->{memDropPackets});

  # Calculate time between 2 checks
  my $delta_time = $new_datas->{last_timestamp} - $old_timestamp;
  $delta_time = 1 if ($delta_time == 0);

  # Calculate value per second
  my $memDropPacketsPerSec = ($new_datas->{memDropPackets} - $old_memDropPackets) / $delta_time;

  # Calculate exit code by comparing value to thresholds
  # Exit code can be : 'OK', 'WARNING', 'CRITICAL', 'UNKNOWN'
  my $exit_code = $self->{perfdata}->threshold_check(value => $memDropPacketsPerSec,
                                                     threshold => [ { label => 'critical', 'exit_litt

  # Add a performance data
  $self->{output}->perfdata_add(label => 'dropped_packets_Per_Sec',
                                value => sprintf("%.2f", $memDropPacketsPerSec),
                                warning => $self->{perfdata}->get_perfdata_for_output(label => 'warni
                                critical => $self->{perfdata}->get_perfdata_for_output(label => 'crit
                                min => 0);

  # Add output
  $self->{output}->output_add(severity => $exit_code,
                              short_msg => sprintf("Dropped packets due to memory limitations : %.2f
                                  $memDropPacketsPerSec));

  # Display output
  $self->{output}->display();
  $self->{output}->exit();
}
```

Declare this plugin as a perl module:

```
1;
```

Add a description of the mode options:

```
__END__

=head1 MODE

Check number of packets per second dropped due to memory limitations.

=over 8

=item B<--warning>

Threshold warning for dropped packets in packets per second.

=item B<--critical>

Threshold critical for dropped packets in packets per second.

=back

=cut
```

Command line

This is an example of command line:

```
$ perl centreon_plugins.pl --plugin apps::pfsense::snmp::plugin --mode memory-dropped-packets --hostr
```

Output may display:

```
OK: Dropped packets due to memory limitations : 0.00 /s | dropped_packets_Per_Sec=0.00;0;;1;2
```

2.5 Code Style Guidelines

2.5.1 Introduction

Perl code from Pull-request must conform to the following style guidelines. If you find any code which doesn't conform, please fix it.

2.5.2 Indentation

Space should be used to indent all code blocks. Tabs should never be used to indent code blocks. Mixing tabs and spaces results in misaligned code blocks for other developers who prefer different indentation settings. Please use 4 for indentation space width.

```
if ($1 > 1) {
....return 1;
} else {
    if ($i == -1) {
    ....return 0;
    }
    return -1
}
```

2.5.3 Comments

There should always be at least 1 space between the # character and the beginning of the comment. This makes it a little easier to read multi-line comments:

```
# Good comment
#Wrong comment
```

2.5.4 Subroutine & Variable Names

Whenever possible, use underscore to seperator words and don't use uppercase characters:

```
sub get_logs {}
my $start_time;
```

Keys of hash table should be used alphanumeric and underscore characters only (and no quote!):

```
$dogs->{meapolitan_mastiff} = 10;
```

2.5.5 Curly Brackets, Parenthesis

There should be a space between every control/loop keyword and the opening parenthesis:

```
if ($i == 1) {
    ...
}
while ($i == 2) {
    ...
}
```

2.5.6 If/Else Statements

'else', 'elsif' should be on the same line after the previous closing curly brace:

```
if ($i == 1) {
    ...
} else {
    ...
}
```

You can use single line if conditional:

```
next if ($i == 1);
```

2.6 Model Classes Usage

2.6.1 Introduction

With the experience of plugin development, we have created two classes:

- centreon::plugins::templates::counter
- centreon::plugins::templates::hardware

It was developed to have a more consistent code and less redundant code. According to context, you should use one of two classes for modes. Following classes can be used for whatever plugin type (SNMP, Custom, DBI,...).

2.6.2 Class counter

When to use it ?

If you have some counters (CPU Usage, Memory, Session...), you should use that class. If you have only one global counter to check, it's maybe not useful to use it (but only for these case).

Class methods

List of methods:

- **new**: class constructor. Overload if you need to add some specific options or to use a statefile.
- **check_options**: overload if you need to check your specific options.
- **manage_selection**: overload if *mandatory*. Method to get informations for the equipment.
- **set_counters**: overload if **mandatory**. Method to configure counters.

Examples

Example 1

We want to develop the following SNMP plugin:

- measure the current sessions and current SSL sessions usages.

```
package my::module::name;

use base qw(centreon::plugins::templates::counter);

use strict;
use warnings;

sub set_counters {
  my ($self, %options) = @_;

  $self->{maps_counters_type} = [
      { name => 'global', type => 0, message_separator => ' - ' },
  ];
  $self->{maps_counters}->{global} = [
      { label => 'sessions', set => {
              key_values => [ { name => 'sessions' } ],
              output_template => 'Current sessions : %s',
              perfdatas => [
                  { label => 'sessions', value => 'sessions_absolute', template => '%s',
                    min => 0 },
              ],
          }
      },
      { label => 'sessions-ssl', set => {
              key_values => [ { name => 'sessions_ssl' } ],
              output_template => 'Current ssl sessions : %s',
              perfdatas => [
                  { label => 'sessions_ssl', value => 'sessions_ssl_absolute', template => '%s',
                    min => 0 },
              ],
          }
      },
  ];
}

sub manage_selection {
  my ($self, %options) = @_;
```

```
  # OIDs are fake. Only for the example.
  my ($oid_sessions, $oid_sessions_ssl) = ('.1.2.3.4.0', '.1.2.3.5.0');

  my $result = $options{snmp}->get_leef(oids => [ $oid_sessions, $oid_sessions_ssl ],
                                        nothing_quit => 1);
  $self->{global} = { sessions => $result->{$oid_sessions},
                      sessions_ssl => $result->{$oid_sessions_ssl}
                    };
}
```

Output may display:

```
OK: Current sessions : 24 - Current ssl sessions : 150 | sessions=24;;;0; sessions_ssl=150;;;0;
```

As you can see, we create two arrays of hash tables in **set_counters** method. We use arrays to order the output.

- **maps_counters_type**: global configuration. Attributes list:
  - *name*: the name is really important. It will be used in hash **map_counters** and also in **manage_selection** as you can see.
  - *type*: 0 or 1. With 0 value, the output will be written in the short output. With the value 1, it depends if we have one or multiple instances.
  - *message_multiple*: only useful with *type* 1 value. The message will be displayed in short ouput if we have multiple instances selected.
  - *message_separator*: the string displayed between counters (Default: ', ').
  - *cb_prefix_output*, *cb_suffix_output*: name of a method (in a string) to callback. Methods will return a string to be displayed before or after **all** counters.
  - *cb_init*: name of a method (in a string) to callback. Method will return 0 or 1. With 1 value, counters are not checked.
- **maps_counters**: complex structure to configure counters. Attributes list:
  - *label*: name used for threshold options.
  - *threshold*: if we set the value to 0. There is no threshold check options (can be used if you want to set and check option yourself).
  - *set*: hash table:
    - *keys_values*: array of hashes. Set values used for the counter. Order is important (by default, the first value is used to check).
      - *name*: attribute name. Need to match with attributes in **manage_selection** method!
      - *diff*: if we set the value to 1, we'll have the difference between two checks (need a statefile!).
    - *output_template*: string to display. '%s' will be replaced by the first value of *keys_values*.
    - *output_use*: which value to be used in *output_template* (If not set, we use the first value of *keys_values*).
    - *per_second*: if we set the value to 1, the *diff* values will be calculated per seconds.
    - *output_change_bytes*: if we set the value to 1 or 2, we can use a second '%s' in *output_template* to display the unit. 1 = divide by 1024 (Bytes), 2 = divide by 1000 (bits).
    - *perfdata*: array of hashes. To configure perfdatas
      - *label*: name displayed.

- *value*: value to used. It's the name from *keys_values* with a **suffix**: '_absolute' or '_per_second' (depends of other options).
- *template*: value format (could be for example: '%.3f').
- *unit*: unit displayed.
- *min*, *max*: min and max displayed. You can use a value from *keys_values*.
- *label_extra_instance*: if we set the value to 1, perhaps we'll have a suffix concat with *label*.
- *instance_use*: which value from *keys_values* to be used. To be used if *label_extra_instance* is 1.

Example 2

We want to add the current number of sessions by virtual servers.

```
package my::module::name;

use base qw(centreon::plugins::templates::counter);

use strict;
use warnings;

sub set_counters {
  my ($self, %options) = @_;

  $self->{maps_counters_type} = [
      { name => 'global', type => 0, cb_prefix_output => 'prefix_global_output' },
      { name => 'vs', type => 1, cb_prefix_output => 'prefix_vs_output', message_multiple => 'All Vi
  ];
  $self->{maps_counters}->{global} = [
      { label => 'total-sessions', set => {
              key_values => [ { name => 'sessions' } ],
              output_template => 'current sessions : %s',
              perfdatas => [
                  { label => 'total_sessions', value => 'sessions_absolute', template => '%s',
                    min => 0 },
              ],
          }
      },
      { label => 'total-sessions-ssl', set => {
              key_values => [ { name => 'sessions_ssl' } ],
              output_template => 'current ssl sessions : %s',
              perfdatas => [
                  { label => 'total_sessions_ssl', value => 'sessions_ssl_absolute', template => '%s
                    min => 0 },
              ],
          }
      },
  ];

  $self->{maps_counters}->{vs} = [
      { label => 'sessions', set => {
              key_values => [ { name => 'sessions' }, { name => 'display' } ],
              output_template => 'current sessions : %s',
              perfdatas => [
                  { label => 'sessions', value => 'sessions_absolute', template => '%s',
```

```
                    min => 0, label_extra_instance => 1, instance_use => 'display_absolute' },
                ],
            }
        },
        { label => 'sessions-ssl', set => {
                key_values => [ { name => 'sessions_ssl' }, { name => 'display' } ],
                output_template => 'current ssl sessions : %s',
                perfdatas => [
                    { label => 'sessions_ssl', value => 'sessions_ssl_absolute', template => '%s',
                      min => 0, label_extra_instance => 1, instance_use => 'display_absolute' },
                ],
            }
        },
    ];
}

sub prefix_vs_output {
    my ($self, %options) = @_;

    return "Virtual server '" . $options{instance_value}->{display} . "' ";
}

sub prefix_global_output {
    my ($self, %options) = @_;

    return "Total ";
}

sub manage_selection {
    my ($self, %options) = @_;

    # OIDs are fake. Only for the example.
    my ($oid_sessions, $oid_sessions_ssl) = ('.1.2.3.4.0', '.1.2.3.5.0');

    my $result = $options{snmp}->get_leef(oids => [ $oid_sessions, $oid_sessions_ssl ],
                                          nothing_quit => 1);
    $self->{global} = { sessions => $result->{$oid_sessions},
                        sessions_ssl => $result->{$oid_sessions_ssl}
                      };
    my $oid_table_vs = '.1.2.3.10';
    my $mapping = {
        vsName        => { oid => '.1.2.3.10.1' },
        vsSessions    => { oid => '.1.2.3.10.2' },
        vsSessionsSsl => { oid => '.1.2.3.10.3' },
    };

    $self->{vs} = {};
    $result = $options{snmp}->get_table(oid => $oid_table_vs,
                                        nothing_quit => 1);
    foreach my $oid (keys %{$result->{ $oid_table_vs }}) {
        next if ($oid !~ /^$mapping->{vsName}->{oid}\.(.*)$/;
        my $instance = $1;
        my $data = $options{snmp}->map_instance(mapping => $mapping, results => $result->{$oid_table_vs

        $self->{vs}->{$instance} = { display => $data->{vsName},
                                     sessions => $data->{vsSessions}, sessions_ssl => $data->{vsSessio
    }
}
```

If we have at least 2 virtual servers:

```
OK: Total current sessions : 24, current ssl sessions : 150 - All Virtual servers are ok | total_sess
Virtual server 'foo1' current sessions : 11, current ssl sessions : 70
Virtual server 'foo2' current sessions : 13, current ssl sessions : 80
```

Example 3

The model can also be used to check strings (not only counters). So we want to check the status of a virtualserver.

```
package my::module::name;

use base qw(centreon::plugins::templates::counter);

use strict;
use warnings;

my $instance_mode;

sub set_counters {
  my ($self, %options) = @_;

  $self->{maps_counters_type} = [
      { name => 'vs', type => 1, cb_prefix_output => 'prefix_vs_output', message_multiple => 'All Vi
  ];
  $self->{maps_counters}->{vs} = [
      { label => 'status', threshold => 0, set => {
              key_values => [ { name => 'status' }, { name => 'display' } ],
              closure_custom_calc => $self->can('custom_status_calc'),
              closure_custom_output => $self->can('custom_status_output'),
              closure_custom_perfdata => sub { return 0; },
              closure_custom_threshold_check => $self->can('custom_threshold_output'),
          }
      },
  ];
}

sub custom_threshold_output {
  my ($self, %options) = @_;
  my $status = 'ok';

  if ($self->{result_values}->{status} =~ /problem/) {
      $status = 'critical';
  }
  return $status;
}

sub custom_status_output {
  my ($self, %options) = @_;

  my $msg = sprintf("status is '%s'", $self->{result_values}->{status});
  return $msg;
}

sub custom_status_calc {
  my ($self, %options) = @_;
```

```
  $self->{result_values}->{status} = $options{new_datas}->{$self->{instance} . '_status'};
  $self->{result_values}->{display} = $options{new_datas}->{$self->{instance} . '_display'};
  return 0;
}

sub prefix_vs_output {
  my ($self, %options) = @_;

  return "Virtual server '" . $options{instance_value}->{display} . "' ";
}

sub check_options {
  my ($self, %options) = @_;
  $self->SUPER::check_options(%options);

  # Sometimes, you'll need to have access of the current object in the callback
  $instance_mode = $self;
}

sub manage_selection {
  my ($self, %options) = @_;

  my $oid_table_vs = '.1.2.3.10';
  my $mapping = {
      vsName        => { oid => '.1.2.3.10.1' },
      vsStatus      => { oid => '.1.2.3.10.4' },
  };

  $self->{vs} = {};
  my $result = $options{snmp}->get_table(oid => $oid_table_vs,
                                         nothing_quit => 1);
  foreach my $oid (keys %{$result->{ $oid_table_vs }}) {
      next if ($oid !~ /^$mapping->{vsName}->{oid}\.(.*)$/;
      my $instance = $1;
      my $data = $options{snmp}->map_instance(mapping => $mapping, results => $result->{$oid_table_vs

      $self->{vs}->{$instance} = { display => $data->{vsName},
                                   status => $data->{vsStatus} };
  }
}
```

The following example show 4 new attributes:

- *closure_custom_calc*: should be used to have a simple name (without '_absolute' or '_per_second'). Or to do some more complex calculation.
- *closure_custom_output*: should be used to have a more complex output (An example: want to display the total, free and used value at the same time).
- *closure_custom_perfdata*: should be used to manage yourself the perfdata.
- *closure_custom_threshold_check*: should be used to manage yourself the threshold check.

2.6.3 Class hardware

TODO