



Centreon Plugins Documentation

Release

Merethis

November 24, 2015

Centreon Plugins is a common monitoring library and plugins written in Perl. It is licensed under the terms of the [GNU General Public License Version 2](#) as published by the Free Software Foundation.

Contents:

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 latest 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/~wolsage/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:
  processcount
  time
  list-storages
  disk-usage
  diskio
  uptime
  swap
  cpu-detailed
  load
  traffic
  cpu
  inodes
  list-diskspath
  list-interfaces
  packet-errors
  memory
  tcpcon
  storage
```

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	TELNET	WSMAN	Protocol		HTTP		WMI	Experimental	Comments
	SSH				SNMP						
	Active Directory								•		Use 'dcdiag' command. Must be installed on Windows. Need Apache 'mod_ssl' module.
Application	Apache					•					
	Apc					•					
	Apcupsd	•							•		Use 'apcu' command.
	Bluemind					•					Use 'fluxd' API.
	Checkmyws					•					

Continued

Table 1.1 – continued from previous page

Category	Check SSH	Transport	TELNET	WSMAN	Protocol SNMP		HTTP		WMI	Experimental	Comments
	Elasticsearch					•					
	Exchange								•		Use powershell script. Must be installed on Windows. Use 'github' API.
	Github					•					
	Hddtemp								•		Open a TTY. Custom communication. Must be installed on Windows. Open MIM.
	IIS						•				
Jenkins			•			•			•		JSON LWP: URI, HTTP XML.
Kayako					•					Use 'kayako' API.	Di-gest:: LWP: URI, HTTP
Lmsensors				•							
Msmq								•	•	Must be installed on Windows. Not developed yet.	

Continued

Table 1.1 – continued from previous page

Category	Check SSH	Transport	TELNET	WSMAN	Protocol SNMP		HTTP		WMI	Experiment	Command
Nginx					•					Need 'Http-Stub-Status-Module' module. Use 'crm_mon' command.	LWP: URI, HTTP
Pacemaker	•							•			
Pfsense				•							
Selenium								•		Connect to a selenium server to play a scenario. Need tomcat web-manager. Use varnish commands. Need 'centreon_esxd' connector from Merethis.	XML: WVV
Tomcat					•						XML: LWP: URI, HTTP
Varnish	•							•			
VMWare								•			
Pfsense				•							
Bgp					•						
Dhcp									•		
Dns									•		
Protocols									•		

Continued

Table 1.1 – continued from previous page

Category	Check SSH	Transport	TELNET	WSMAN	Protocol SNMP		HTTP		WMI	Experimental	Com JMX
Database	Ftp								.		
	Http					.					
	Ftp								.		
	Imap								.		
	Jmx							.			
	Ldap								.		
	Ntp								.		
	Radius								.		
	Smtpt								.		
	Tcp								.		
	Udp								.		
	x509								.		
	Informix								.		
	MS SQL								.		
	MySQL								.		

Continued

Table 1.1 – continued from previous page

Category	Check SSH	Transport	TELNET	WSMAN	Protocol SNMP		HTTP		WMI	Experimental	Command JMX
Hardware	Oracle								•		
	Postgres								•		
	ATS Apc				•					•	
	PDU Apc				•					•	
	PDU Eaton				•					•	
	Standard Printers				•						
	Sensorip				•						
	Sensormetrix Em01					•					
	Serverscheck				•						
	Cisco UCS				•						
	Dell CMC				•						
	Dell iDrac				•						
	Dell Open- manage				•						
Continued											

Table 1.1 – continued from previous page

Category	Check SSH	Transport	TELNET	WSMAN	Protocol SNMP		HTTP		WMI	Experimental	Can JMX
	HP Pro- liant				•						Need 'HP Insight agent on oper- ating system
	HP Blade Chassis				•						
	IBM Blade- Center				•						
	IBM HMC	•							•	•	
	IBM IMM				•						
	Sun hard- ware	•	•		•				•		Can monit many sun hard- ware.
	UPS Mge				•						
	UPS Stan- dard				•						
	UPS Power- ware				•						
	Alcatel Om- niswitch				•						
	Arkoon				•						
	Aruba				•						
	Bluecoat				•						
Network	Brocade				•						

	Checkpoint				.						
	Cisco				.						Many cisco (2800 Nexu Wlc, Iron-port,,
	Citrix Netscaler				.						
	Dell Power-connect Dlink				.						
	Extreme				.						
	F5 Big-IP				.						
	Fortinet Forti-gate Fritzbox				.						
	H3C				.						
	Hirschmann				.						

AIX

-

Use AIX commands.

-

Freebsd

-

Need 'bsnmpd' agent.

Linux

-

Use Linux commands.

-

Solaris

-

-

Use Solaris commands.

-

Windows

-

-

-

-

Storage Dell EqualLogic

-

Dell MD3000

-

Need 'SMcli' command.
Dell TL2000

-

EMC Celerra

-

-

Use appliance commands.
EMC Clariion

-

Need 'navisphere' command.
EMC DataDomain

-

EMC Recoverypoint

-

-

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
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'=0
```

There are too many perfdatas and i want to keep 'total' perfddata only. I use the option `--filter-perfddata='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-perfddata` option. So, i can exclude perfddata 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=snmp_standard::plugin --mode=numeric-value --oid='.1.3.6.1.2.1.1
```

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:feb5:6a
```

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=traffic --hostname=127.0.0.1 --snmp
OK: All traffic are ok | 'traffic_in_lo'=197.40b/s;;;0;10000000 'traffic_out_lo'=197.40b/s;;;0;10000000
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 use a mode with a plugin. It is 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::poller
OK: All poller delay for last update are ok | 'delay_Central'=2s;0:300;0:600;0; 'delay_Poller-Engine'=2s;0:300;0:600;0;
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 --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 --snmp
UNKNOWN: Not good version for plugin mode. Excepted at least: 2.x. Get: 1.0
```

1.5 Troubleshooting

1.5.1 SNMP

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

The following error can happen 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 'traff
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 authentication of an application (POST request)

An example for authentication 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;
```

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.

The latest version is available on following git repository: <http://git.centreon.com/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}} = (
    'model1'    => '<plugin_path>::mode::model1',
    'model2'    => '<plugin_path>::mode::model2',
    ...
);
```

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 model.pm
```

Then, edit model.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::model;
```

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->{perfddata}->threshold_validate(label => 'warning', value => $self->{option_results}->{warning_value})) {
    $self->{output}->add_option_msg(short_msg => "Wrong warning threshold '" . $self->{option_results}->{warning_value} . "'");
    $self->{output}->option_exit();
}
if (($self->{perfddata}->threshold_validate(label => 'critical', value => $self->{option_results}->{critical_value})) {
    $self->{output}->add_option_msg(short_msg => "Wrong critical threshold '" . $self->{option_results}->{critical_value} . "'");
    $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->{perfddata}->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 => 'warn
                           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 <code>-verbose</code> 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 Perfddata

This library allows you to manage performance data.

get_perfddata_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    => 300000000,
                             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->{perfddata}->threshold_validate(label => 'warning', value => 80);
$self->{perfddata}->threshold_validate(label => 'critical', value => 90);
my $prct_used = 85;

my $exit = $self->{perfddata}->threshold_check(value => $prct_used, threshold => [ { label => 'critical' } ]);

$self->{output}->output_add(severity => $exit,
                           short_msg => sprintf("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->{perfddata}->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 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 occurred.
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 => 1);

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, $oid_memoryDeviceFailureModes],
                  instances => [keys %$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 occurred.
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_rcDeviceErrWatchdogReset);

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 occurred.
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 occurred.

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 methods 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 <code>--credentials</code> is used).
username	String	Username (if <code>--credentials</code> is used).
password	String	User password (if <code>--credentials</code> 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 occurred.

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.

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 2005-2015 MERETHIS
# Centreon is developped by : Julien Mathis and Romain Le Merlus under
# GPL Licence 2.0.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation ; either version 2 of the License.
#
# This program is distributed in the hope that it will be useful, but WITHOUT ANY
# WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
# PARTICULAR PURPOSE. See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, see <http://www.gnu.org/licenses>.
#
# Linking this program statically or dynamically with other modules is making a
# combined work based on this program. Thus, the terms and conditions of the GNU
# General Public License cover the whole combination.
#
# As a special exception, the copyright holders of this program give MERETHIS
# permission to link this program with independent modules to produce an executable,
# regardless of the license terms of these independent modules, and to copy and
# distribute the resulting executable under terms of MERETHIS choice, provided that
# MERETHIS also meet, for each linked independent module, the terms and conditions
# of the license of that module. An independent module is a module which is not
# derived from this program. If you modify this program, you may extend this
# exception to your version of the program, but you are not obliged to do so. If you
# do not wish to do so, delete this exception statement from your version.
#
# For more information : contact@centreon.com
# Authors : your name <your@mail>
#
#####

# 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 2005-2015 MERETHIS
# Centreon is developped by : Julien Mathis and Romain Le Merlus under
# GPL Licence 2.0.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation ; either version 2 of the License.
#
# This program is distributed in the hope that it will be useful, but WITHOUT ANY
# WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
# PARTICULAR PURPOSE. See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, see <http://www.gnu.org/licenses>.
#
# Linking this program statically or dynamically with other modules is making a
# combined work based on this program. Thus, the terms and conditions of the GNU
# General Public License cover the whole combination.

```



```

#
# As a special exception, the copyright holders of this program give MERETHIS
# permission to link this program with independent modules to produce an executable,
# regardless of the license terms of these independent modules, and to copy and
# distribute the resulting executable under terms of MERETHIS choice, provided that
# MERETHIS also meet, for each linked independent module, the terms and conditions
# of the license of that module. An independent module is a module which is not
# derived from this program. If you modify this program, you may extend this
# exception to your version of the program, but you are not obliged to do so. If you
# do not wish to do so, delete this exception statement from your version.
#
# For more information : contact@centreon.com
# Authors : your name <your@mail>
#
#####

# 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}->{warning})) {
        $self->{output}->add_option_msg(short_msg => "Wrong warning threshold '" . $self->{option_results}->{warning} . "'" );
        $self->{output}->option_exit();
    }
    if (($self->{perfdata}->threshold_validate(label => 'critical', value => $self->{option_results}->{critical})) {
        $self->{output}->add_option_msg(short_msg => "Wrong critical threshold '" . $self->{option_results}->{critical} . "'" );
        $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 previously 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_port});
    # 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->{perfddata}->threshold_check(value => $memDropPacketsPerSec,
                                                         threshold => [ { label => 'critical', 'exit_lit
    }

    # Add a performance data
    $self->{output}->perfddata_add(label => 'dropped_packets_Per_Sec',
                                   value => sprintf("%.2f", $memDropPacketsPerSec),
                                   warning => $self->{perfddata}->get_perfddata_for_output(label => 'warn
                                   critical => $self->{perfddata}->get_perfddata_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 --host
```

Output may display:

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