The Monitoring Plugins Interface

Abstract

This document aims to document the Monitoring Plugin Interface, a standard more or less strictly implemented by different network monitoring solutions. Implementers and Users of network monitoring solutions, monitoring plugins and libraries can use this as a reference point as to how these programs interface with each other.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-kaestle-monitoring-plugins-interface/.

Source for this draft and an issue tracker can be found at https://github.com/RincewindsHat/rfc-monitoring-plugins-interface.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 March 2024.
1. Introduction

With the emergence of NetSaint/Nagios at the latest, these system and their successors/forks have relied on a loose group of programs called "Monitoring Plugins" to do the lower level task of actually determining the state of a particular entity or conduct measurements of certain values.

This document shall help users and especially developers of those programs as a basis on how they should be implemented, how they should work and how they should behave. It encourages the standardization of libraries, Monitoring Plugins and Monitoring Systems, to reduce the cognitive load on users, administrators and developers, if they work with different implementations.

This document aims to be as general as possible and not to assume a special implementation detail, e.g. the programming language, the
install mechanism or the monitoring system which executes the Monitoring Plugin.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Range expressions

In many cases thresholds for metrics mark a certain range of values where the values is considered to be good or bad if it is inside or outside. While for significant number of metrics a upper (e.g. load on unixoid systems) or lower (e.g. effective throughput, free space in memory or storage) border might suffice, for some it does not, for example a temperature value from a temperature sensor should be within certain range (e.g. between 10℃ and 45℃).

Regarding input parameters this might be handled with options like --critical-upper-temperature and --critical-lower-temperature, but this creates a problem in the performance data output, if only scalar values could be used. To resolve this situation the Range expression format was introduced, with the following definition:

[@][start:][end]

where:

1. At least start or end MUST be provided.
2. start <= end
3. If start == 0, then start can be omitted.
4. If end is omitted, it has the "value" of positive infinity.
5. Negative infinity can be specified with the tilde character ~.
6. If the prefix @ IS given, the value exceeds the threshold if it is INSIDE the range between start and end (including the endpoints).
7. If the prefix @ is NOT given, the value exceeds the threshold if it is OUTSIDE of the range between start and end (including the endpoints).
2.1.1. Examples

Range definition Exceeds threshold if x...
10 < 0 or > 10, (outside the range of {0 .. 10})
10: < 10, (outside {10 .. ∞})
~:10 > 10, (outside the range of {-∞ .. 10})
10:20 < 10 or > 20, (outside the range of {10 .. 20})
@10:20 ≥ 10 and ≤ 20, (inside the range of {10 .. 20})

Table 1

3. The basic Monitoring Plugin usage

A Monitoring System executes a Monitoring Plugin. The Monitoring Plugin MAY accept parameters in the form of command line arguments, environment variables or a configuration file (the location of which MAY in turn be given on the command line or via environment variable). The Monitoring Plugin then proceeds to execute its duty and returns the result to the Monitoring System. Part of the process of returning the result is the termination of the execution of the Monitoring Plugin itself.

4. Input Parameters for a Monitoring Plugin

A Monitoring Plugin MUST expect input parameters as arguments during execution, if any are needed/expected at all. It MAY accept these parameters given as environment variables and it MAY accept them in a configuration file (with a default path or a path given via arguments or environment variables).

In general positional arguments are strongly discouraged.

Some arguments MUST have this predetermined meaning, if they are used:

<table>
<thead>
<tr>
<th>Argument (long)</th>
<th>Argument (short version, optional)</th>
<th>Meaning</th>
<th>can be given multiple times</th>
</tr>
</thead>
<tbody>
<tr>
<td>--help -h</td>
<td>None</td>
<td>Triggers the help functionality of the Monitoring Plugin, showing the individual parameters and their meaning,</td>
<td>-- (makes no difference)</td>
</tr>
<tr>
<td>Argument (long)</td>
<td>Argument (short version, optional)</td>
<td>Meaning</td>
<td>optional</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>--version</td>
<td>-V</td>
<td>None</td>
<td>no</td>
</tr>
<tr>
<td>--timeout</td>
<td>-t</td>
<td>Integer</td>
<td>no</td>
</tr>
</tbody>
</table>

Examples for usage of the Monitoring Plugin and general remarks about the how and why of the Monitoring Plugin. **SHOULD overwrite all other options, meaning, they are ignored if --help is given.** The Monitoring Plugin **SHOULD exit with state UNKNOWN (3).** Shows the version of the Monitoring Plugin to allow users to report errors better and therefore help them and the developers. The Monitoring Plugin **SHOULD exit with state UNKNOWN (3).** Sets a limit for the time which a Monitoring...
<table>
<thead>
<tr>
<th>Argument (long)</th>
<th>Argument (short version, optional)</th>
<th>Meaning</th>
<th>optional can be given multiple times</th>
</tr>
</thead>
<tbody>
<tr>
<td>time duration</td>
<td>string</td>
<td>Plugin is given to execute. This is there to enforce the abortion of the test and improve the reaction time of the Monitoring System (e.g. in bad network conditions it might be helpful to abort the test prematurely and inform the user about that, than trying forever to do something which won't succeed. Or if soft real time constraints are present, a result might be worthless altogether after some time). A sane default is probably 30 seconds, although this depends heavily on the scenario</td>
<td></td>
</tr>
<tr>
<td>Argument (long)</td>
<td>Argument (short version, optional)</td>
<td>Meaning</td>
<td>optional</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>--hostname -H</td>
<td>hostname</td>
<td>String, meaning either a DNS name or an IP address of the targeted system</td>
<td>and should be given a thought during development. If the execution is terminated by this timeout, it should exit with state UNKNOWN (3) and (if possible) give some helpful output in which stage of the execution the timeout occurred. If the Monitoring Plugin targets exactly one other system on the network, this option should be used to tell it which one. If the Monitoring Plugin does its test just locally or the logic does not apply to it, this option is, of course, optional.</td>
</tr>
<tr>
<td>Argument (long)</td>
<td>Argument (short version, optional)</td>
<td>Meaning</td>
<td>optional can be given multiple times</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------</td>
<td>---------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>--verbose -v</td>
<td>None</td>
<td>Increases the verbosity of the output, thereby breaking the suggested rules about a short and concise output. The intention is to provide more information to a user.</td>
<td>yes     yes</td>
</tr>
</tbody>
</table>

The Monitoring Plugin exits unconditionally with OK (0). Mostly useful for the purpose of packaging and testing plugins, but might be used to always ignore errors (e.g. to just collect data). | yes | no |

Table 2

4.1. Examples

For the execution with --help:

```
$ my_check_plugin --help
```

the output might look like this:
This imaginary Monitoring Plugin tries to be really helpful here, displays the version, the license and the upstream repository with the help (although not necessary), has a short description about the purpose, lists the options in an easily readable way and even gives some examples.

Usage:
my_check_plugin --name NAME [--greeting GREETING]

Options:
--help
  this help
--version
  Shows the version of the plugin
--name NAME
  if given, uses NAME as a name to greet.
--greeting GREETING
  if given, uses GREETING instead of Hello.

Examples:
$ my_check_plugin --name Jane
Hello Jane

$ my_check_plugin --greeting Ciao --name Alice
Ciao Alice

This imaginary Monitoring Plugin tries to be really helpful here, displays the version, the license and the upstream repository with the help (although not necessary), has a short description about the purpose, lists the options in an easily readable way and even gives some examples.

For the execution with --version

$ my_check_plugin --version

  the output might be a bit shorter:

my_check_plugin version 3.1.4

  or even:

3.1.4

  where both show the necessary information.

5. Output of a Monitoring Plugin

The output of a Monitoring Plugin consists of two parts on the first level, the Exit Code and output in textual form on stdout.
5.1. Exit Code

The Monitoring Plugin **MUST** make use of the *Exit Code* as a method to communicate a result to the Monitoring System. Since the *Exit Code* is more or less standardized over different systems as an integer number with a width of or greater than 8bit, the following mapping is used:

<table>
<thead>
<tr>
<th>Exit Code (numerical)</th>
<th>Meaning (short)</th>
<th>Meaning (extended)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
<td>The execution of the Monitoring Plugin proceeded as planned and the tests appeared to function properly and the measured values are within their respective thresholds.</td>
</tr>
<tr>
<td>1</td>
<td>WARNING</td>
<td>The execution of the Monitoring Plugin proceeded as planned and the tests appeared to not function properly or the measured values are not with their respective thresholds. The problem(s) do(es) not seem exceptionally grave though and do(es) not require immediate attention.</td>
</tr>
<tr>
<td>2</td>
<td>CRITICAL</td>
<td>The execution of the Monitoring Plugin proceeded as planned and the tests appeared to not function properly or the measured values are not with their respective thresholds. The problem(s) do(es) seem exceptionally grave though and do(es) require immediate attention.</td>
</tr>
<tr>
<td>3</td>
<td>UNKNOWN</td>
<td>The execution of the Monitoring Plugin did not proceed as planned. The reasons might be manifold, e.g. missing permissions, missing libraries, no available network connection to the destination, etc.. In summary: The Monitoring Plugin could not determine the state of whatever it should have been checking and can therefore make no reliable statement about it.</td>
</tr>
<tr>
<td>4-125</td>
<td>reserved for future use</td>
<td></td>
</tr>
</tbody>
</table>

Table 3

5.2. Textual Output

The original purpose of the output on *stdout* was to provide human readable information for the user of the Monitoring System, a way for the Monitoring Plugin to communicate further details on what
happened. This purpose still exists, but was expanded with the, so called, performance data to allow the machine readable communication of measured values for further processing in the Monitoring System, e.g. for the creation of diagrams.

Therefore the further explanation is split into human readable output and performance data.

5.2.1. Human readable output

This part of the output should give an user information about the state of the test and, in the case of problems, ideally hint what the origin of the problem might be or what the symptoms are. If the test relies on numeric values, this might be displayed to give an user more information about the specific problem. It might consist of one or more lines of printable symbols.

Although no strict guidelines for creating this part of the output can really be given, a developer should keep a potential user in mind. It might, for example, be OK to put the output in a single line if there are only one or two items of a similar type (think: multiple file systems, multiple sensors, etc.) are present, but not if there 10 or 100, although this might present a valid use case. If there are several different items exists in the output of the Monitoring Plugin they probably SHOULD be given their own line in the output.

5.2.1.1. Examples

Remaining space on filesystem "/" is OK

Sensor temperature is within thresholds

Available Memory is too low

Sensor temperature exceeds thresholds

are OK, but

Remaining space on filesystem "/" is OK ( 62GiB / 128GiB )

Sensor temperature is within thresholds ( 42°C )

Available Memory is too low ( 126MiB / 32GiB )

Sensor temperature exceeds thresholds ( 78°C > 70°C )

are better.
5.2.2. Performance data

In addition to the human readable part the output can contain machine readable measurement values. These data points are separated from the human readable part by the "|" symbol which is in effect until the end of the output. The performance data then MUST consist of space (ASCII 0x20) separated single values, these MUST have the following format:

[']label[']=value[UOM][;warn[;crit[;min[;max]]]]

with the following definitions:

1. label MUST consist of at least one non-space character, but can otherwise contain any printable characters except for the equals sign (=) or single quotes ('). If it contains spaces, it must be surrounded by single quotes

2. value is a numerical value, might be either an integer or a floating point number. Using floating point numbers if the value is really discreet SHOULD be avoided. The representation of a floating point number SHOULD NOT use the "scientific notation" (e.g. 6.02e23 or -3e-45), since some systems might not be able to parse them correctly. Values with a base other then 10 SHOULD be avoided (see below for more information on Byte values).

3. UOM is the Unit of measurement (e.g. "B" for Bytes, "s" for seconds) which gives more context to the Monitoring System.

*The following constraints MUST be applied:

1. An UOM of % MUST be used for percentage values

2. An UOM of c MUST be used for continuous counters (commonly used for the sum of bytes transmitted on an interface)

*The following recommendations SHOULD be applied:

1. The UOM for Byte values is B and although many systems do understand units like KB,KiB, MB, GB, TB they SHOULD be avoided, at the least to avoid the ugly hassle about people misinterpreting the base10 values as base2 values and the other way round. This is also a prime example where floating point number SHOULD NOT be used, since there are obviously only integer numbers included.
2. The UOM for time is s, meaning seconds, SI- Prefixes (e.g. ms for milli seconds) are allowed if necessary or useful.

3. In general, SI units and SI prefixes \textit{MAY} be used as UOM if applicable, but the Monitoring System may not understand them correctly (mostly in uncommon cases), in that cases appropriate workarounds \textit{MAY} be applied on the side of the Monitoring Plugin. Since the values are not intended to be human readable normalized units are recommended (e.g. overall\_power=14000000000W instead of overall\_power=14GW)

4. \texttt{warn} and \texttt{crit} are the threshold values for this measurement, which may have been given by the user as input, may be hardcoded in the Monitoring Plugin or may be retrieved from a file or a device or somewhere else during the execution of the Monitoring Plugin. The unit used \textbf{MUST} be the same as for \texttt{value}. These values are not simple numbers, but range expressions (Section 2.1).

5. \texttt{min} and \texttt{max} are the minimal respectively the maximal value the value could possibly be. The unit \textbf{MUST} be the same as for \texttt{value}. These values can be omitted, if the value is a percentage value, since \texttt{min} and \texttt{max} are always 0 and 100 in this case.

6. \textbf{Implementation Status}

The interface mentioned here is implemented by several network monitoring systems. A non-exhaustive list of these systems includes:

* Icinga 2
* Naemon
* Nagios

The other side of the interface is implemented by several different projects, again in a non-exhaustive list:

* The Monitoring Plugins Project
* The Nagios Plugins Project
* The Linuxfabrik Monitoring Plugins
* Madrisan Nagios Plugins
7. Security Considerations

Special security considerations are hard to define regarding this topic. Regarding the implementation of this interface, the usual programming security considerations should apply (e.g. sanitize inputs), but the risks and problems regarding security are dependent on the specific implementation and usage.

8. IANA Considerations

This document has no IANA actions.

9. Normative References


Acknowledgments

Thanks for previous have to be said to the original inventors of this interface, although it is not easy to determine who these persons are, so they are mentioned here in general.

Thanks are going also to the many different implementors on either side of this interface for their hard work which allows the use of different components and systems with each other in the best spirit of free software.

Author's Address

Lorenz Kästle
The Monitoring Plugins Project

Email: lorenz@vulgrim.de