
Table of Contents

[Preface](#)

1. [Development platform requirements](#) 开发平台要求
2. [Plugin Output for Nagios](#) Nagios 插件输出
 - 2.1. [Print only one line of text](#) 只输出一行文本
 - 2.2. [Verbose output](#) 具体输出
 - 2.3. [Screen Output](#) 屏幕输出
 - 2.4. [Plugin Return Codes](#) plugin 返回代码
 - 2.5. [Threshold and ranges](#) 阈值和区间
 - 2.6. [Performance data](#) 性能数据
 - 2.7. [Translations](#) 翻译
3. [System Commands and Auxiliary Files](#) 系统命令和附录文件
 - 3.1. [Don't execute system commands without specifying their full path](#) 不指定完整路径的状况下请不要执行系统命令
 - 3.2. [Use spopen if external commands must be executed](#) 使用 spopen 执行外部命令
 - 3.3. [Don't make temp files unless absolutely required](#) 假设不是确定需要不要创立临时文件
 - 3.4. [Don't be tricked into following symlinks](#) 不要被下面的符号连接哄骗
 - 3.5. [Validate all input](#) 验证全部输入

4. [Perl Plugins](#) Perl 插件

5. [Runtime Timeouts](#) 运行时超时

5.1. [Use DEFAULT_SOCKET_TIMEOUT](#) 使用DEFAULT_SOCKET_TIMEOUT

5.2. [Add alarms to network plugins](#) 添加报警到网络插件

6. [Plugin Options](#) 插件选项

6.1. [Option Processing](#) 选项处理

6.2. [Plugins with more than one type of threshold, or with threshold ranges](#) 有多个临界值

或有临界值序列的插件

7. [Test cases](#) 测试用例

7.1. [Test cases for plugins](#) 插件测试用例

7.2. [Testing the C library functions](#) 测试 C 库函数

8. [Coding guidelines](#) 编码向导

8.1. [C coding](#) C 编码

8.2. [Crediting sources](#) 认证源码

8.3. [CVS comments](#) CVS 提交

8.4. [Translations for developers](#) 开发者翻译

8.5. [Translations for translators](#) 翻译人员翻译

9. [Submission of new plugins and patches](#) 提交插件或补丁

9.1. [Patches](#) 补丁

9.2. [Contributed plugins](#) 分发插件

9.3. [New plugins](#) 插件

List of Tables

1. [Verbose output levels](#) 具体输出级别

2. [Plugin Return Codes](#) 插件返回代码
 3. [Example ranges](#) 系列实例
 4. [Command line examples](#) 命令行实例
-

Preface

The purpose of this guidelines is to provide a reference for the plug-in developers and encourage the standarization of the different kind of plug-ins: C, shell, perl, python, etc.这个向导的目的是为插件开发者供给一个手册，并为不同类型插件〔如： C, shell, perl, python 等〕供给一个标准。

Nagios Plug-in Development Guidelines Copyright (C) 2023-2023 (Nagios Plugins Team)

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

The plugins themselves are copyrighted by their respective authors.插件的版权归开发者个人。

Table of Contents

1. [Development platform requirements](#)
2. [Plugin Output for Nagios](#)
 - 2.1. [Print only one line of text](#)
 - 2.2. [Verbose output](#)
 - 2.3. [Screen Output](#)
 - 2.4. [Plugin Return Codes](#)
 - 2.5. [Threshold and ranges](#)
 - 2.6. [Performance data](#)
 - 2.7. [Translations](#)
3. [System Commands and Auxiliary Files](#)
 - 3.1. [Don't execute system commands without specifying their full path](#)
 - 3.2. [Use spopen if external commands must be executed](#)
 - 3.3. [Don't make temp files unless absolutely required](#)
 - 3.4. [Don't be tricked into following symlinks](#)
 - 3.5. [Validate all input](#)

- 4. [Perl Plugins](#)
- 5. [Runtime Timeouts](#)
 - 5.1. [Use DEFAULT_SOCKET_TIMEOUT](#)
 - 5.2. [Add alarms to network plugins](#)
- 6. [Plugin Options](#)
 - 6.1. [Option Processing](#)
 - 6.2. [Plugins with more than one type of threshold, or with threshold ranges](#)
- 7. [Test cases](#)
 - 7.1. [Test cases for plugins](#)
 - 7.2. [Testing the C library functions](#)
- 8. [Coding guidelines](#)
 - 8.1. [C coding](#)
 - 8.2. [Crediting sources](#)
 - 8.3. [CVS comments](#)
 - 8.4. [Translations for developers](#)
 - 8.5. [Translations for translators](#)
- 9. [Submission of new plugins and patches](#)
 - 9.1. [Patches](#)
 - 9.2. [Contributed plugins](#)
 - 9.3. [New plugins](#)

1. Development platform requirements

Nagios plugins are developed to the GNU standard, so any OS which is supported by GNU should run the plugins. While the requirements for compiling the Nagios plugins release are very basic, developing from the Git repository requires additional software to be installed.

These are the minimum levels of software required: 下面是最根本软件环境要求:

GNU make 3.79
GNU automake 1.9.2
GNU autoconf 2.59
GNU m4 1.4.2
GNU libtool 1.5

To compile from Git, after you have cloned the repository, run: 从 Git 编译, 从库中复制了代码后, 运行以下命令:

```
tools/setup  
./configure  
make  
make install
```

1. Plugin Output for Nagios

You should always print something to STDOUT that tells if the service is working or why it is failing. Try to keep the output short - probably less than 80 characters. Remember that you ideally would like the entire output to appear in a message, which will get chopped off after a certain length. 你总是应当输出一些东西到 STDOUT 来说明效劳正在工作或者它为什么会失败了。试着保持输出的简短-最好少于 80 个字符。记住最抱负的是在一个页面中输出全部的消息，否则它们到达肯定的长度会被裁掉。

As Nagios does not capture stderr output, you should only output to STDOUT and not print to STDERR. Nagios 不捕获STDERR 输出，你应当只输出到STDOUT 而不要打印到STDERR 中。

1.1. Print only one line of text 输出一行文本

Nagios will only grab the first line of text from STDOUT when it notifies contacts about potential problems. If you print multiple lines, you're out of luck (though this will be a feature of Nagios 3). Remember, keep your output short and to the point. Nagios 从 STDOUT 中只抓取第一行文本，当它通知联系人可能的问题时。假设你输出多行，你就不走运了（尽管这可能会是Nagios3 的一个特征）。记住，保持你的输出简短和切题。

Output should be in the format: 输出的格式如下：

SERVICE STATUS: Information text

However, note that this is not a requirement of the API, so you cannot depend on this being an accurate reflection of the status of the service - the status should always be determined by

the return code. 留意这不是API 的要求，所以你不必依靠于它，来准确地反映效劳的状态——状态总是由返回的代码打算的。

1.1. Verbose output 具体输出

Use the -v flag for verbose output. You should allow multiple -v options for additional verbosity, up to a maximum of 3. The standard type of output should be: 使用 -v 标记详细输出。你应当为更具体的内容允很多个-v 选项。最多可以到达 3 个。标准的输出类型可以是：

Table 1. Verbose output levels

Verbosity level	Type of output
0	Single line, minimal output. Summary 单行，最小输出，简要
1	Single line, additional information (eg list processes that fail) 单行，附加信息（比方失败的过程处理）
2	Multi line, configuration debug output (eg ps command used) 多行，配置高度输出（如进程命令使用）
3	Lots of detail for plugin problem diagnosis 很多的插件具体问题诊断等

1.2. Screen Output 屏幕输出

The plug-in should print the diagnostic and just the usage part of the help message. A well written plugin would then have --help as a way to get the verbose help. 插件应当打印诊断信息，它只是有用的帮助信息的一局部。一个编写良好的插件应当有一个 -- help 作为一个获得具体帮助的一种。

Code and output should try to respect the 80x25 size of a crt (remember when fixing stuff in the server room!) 代码和输出应当试着遵守 80x25 尺寸的显示器

1.1. Plugin Return Codes 插件返回代码

The return codes below are based on the POSIX spec of returning a positive value. Netsaint prior to v0.0.7 supported non-POSIX compliant return code of “-1“ for unknown. Nagios supports POSIX return codes by default. 下面的返回代码是基于POSIX 细则返回一个正值。之前的v0.0.7 版本支持non-POSIX 兼容，返回-1 代码来表示一个UNKNOWN 状态。Nagios 默认支持POSIX 返回代码。

Note: Some plugins will on occasion print on STDOUT that an error occurred and error code is 138 or 255 or some such number. These are usually caused by plugins using system commands and having not enough checks to catch unexpected output. Developers should include a default catch-all for system command output that returns an UNKNOWN return code. 留意：一些插件间或在向STDOUT 输出，当一个错误发生时，错误代码是 138 或 255 或者是一个些其它数字。这些通常是由于使用系统命令并且没有足够的检查来捕获特别输出。开发着应当为系统命令输出包含一个默认的捕获全部来返回一个 UNKNOWN 代码。

Table 2. Plugin Return Codes 插件返回代码

Numeric Value	Service Status	Status Description
0	OK	The plugin was able to check the service and it appeared to be functioning properly 插件可以检查效劳并且可以正确呈现功能
1	Warning	The plugin was able to check the service, but it appeared to be above some “warning “ threshold or did not appear to be working properly 插件可以检查效劳，但是它呈现出一些临界警告信息，或者不能正确工作
2	Critical	The plugin detected that either the service was not running or it was above some “critical “ threshold 插件检测到效劳可能没有运行或者它到达某些临界值
3	Unknown	Invalid command line arguments were supplied to the plugin or low-level failures internal to the plugin (such as unable to

Numeric Value	Service Status	Status Description
		fork, or open a tcp socket) that prevent it from performing the specified operation. Higher-level errors (such as name resolution errors, socket timeouts, etc) are outside of the control of plugins and should generally NOT be reported as UNKNOWN states. 向插件供给了不详的命令行参数或者插件低级内部失败（如不能创立子进程或翻开 tcp 连接）阻挡了它处理指定的操作。高级错误（比方一些误差，socket 超时等）是插件掌握之外的并且通常不能被报告的即为 UNKOWN 状态。

1.1. Threshold and ranges 临界和区间

A range is defined as a start and end point (inclusive) on a numeric scale (possibly negative or positive infinity). 区间被定义来作为一个开头和完毕点（包含）数值范围（可能是负数或下无穷大）

A threshold is a range with an alert level (either warning or critical). Use the `set_thresholds(thresholds *, char *, char *)` function to set the thresholds. 一个阈值是一个报警级别的区间（warning 或 critical）。使用函数 `set_thresholds(thresholds *, char *, char *)`来设置阈值。

The theory is that the plugin will do some sort of check which returns back a numerical value, or metric, which is then compared to the warning and critical thresholds. Use the `get_status(double, thresholds *)` function to compare the value against the thresholds. 理论上，插件会做一些简短的检查，这些检查会返回一个数据值，或者度量标准，这些值会同 warning 和 critical 阈值进展比较。使用函数 `get_status(double, thresholds *)`来与阈值进展比较。

This is the generalised format for ranges: 下面是区间的通常格式：

`[@]start:end`