Installation Guide for Smart-Snmpd Nagios Plugins

Jens Rehsack

Contents

1 Introduction

1.1 Audience

This document targets people who intend to build and install smart-snmpd nagios plugins from source code. Usually only developers and package maintainers are doing this.

If you're neither of them, it's strongly recommended you find an appropriate binary package.

For those who still wants to build from source (or probably just have to), smart-snmpd nagios plugins follow the "./configure && make" philosophy introduced by open source packages for Unices and compatibles. Keep in mind that smart-snmpd nagios plugins have some dependencies, and some of the dependencies have either.

1.2 Dependency Overview

1.2.1 Tools

The smart-snmpd nagios plugins build infrastructure needs following tools to compile the check commands:

- libtool 2.2 or later
- pkg-config 0.20 or later
- a reasonable C++ Compiler supporting the C++-98 standard
- a POSIX.2001 compatible runtime environment

1.2.2 3rd-Party Libraries

- snmp++ 3.2.26 or later, probably not snmp++ 4.0
- log4cplus 1.1.0 or later, in case snmp++ is built with logging enabled
- boost 1.45 or later

1.2.3 Maintainer Tools

- autoconf 2.63 or later
- automake 1.11 or later
- LaTeX
- Doxygen
- Graphviz
- TeX4ht
- w3m

1.3 Dependencies in Detail

If not explicitely mentioned, MacOS X is rated as a normal Unix compatible operating system.

1.3.1 libtool 2.2 or later (REQUIRED)

Currently it's just a named requirement, because all functionality is built-in. In a later release it will put into a library bundled with smart-snmpd and I want to avoid surprises.

1.3.2 pkg-config 0.20 or later (REQUIRED)

Required to configure smart-snmpd nagios plugins.

You can avoid this requirement by setting the environment variables

- `snmp_CFLAGS, snmp_LIBS`
- `boost_CFLAGS, boost_LIBS`
- `log4cplus_CFLAGS, log4cplus_LIBS`

to appropriate values or simply let configure catch the error and try to find the dependent libraries on it own.

1.3.3 A Reasonable C++ Compiler Supporting the C++-98 Standard (REQUIRED)

The `smart-snmpd nagios plugins` uses ISO C++ 98 and `./configure` will fail if it cannot detect a reasonable C++ compiler.

Please understand *reasonable* as follows:

- full C++98 support
- support for `long long` data type
- support for variadic macros

1.3.4 snmp++ 3.2.26 or later, probably not snmp++ 4.0 (REQUIRED)

The library snmp++ in the version 3.2.26 built using it's autoconf toolchain is mandatory for the smart-snmpd nagios plugins.

You can always download the latest release of snmp++ from it's primary download site at `http://www.agentpp.com/`.

1.3.5 log4cplus 1.1.0 or later, in case snmp++ is built with logging enabled

log4cplus 1.1.0 has some fixes for threaded building and since snmp++ is usually built with threads it's mandatory to build the entire dependency tree with threads.

1.3.6 boost 1.45 or later

`Boost` is used for evaluating program options and doing a lot of conversion for user interaction (output). Further, some plugins use additional features from boost libraries (eg. regex).

1.3.7 autoconf 2.63 or later (REQUIRED)

Required to rebuild the autoconf toolchain (Developers only).

On Unix and compatible systems you can either use the autoconf provided by your manufacturer or distributor (if current enough) or install an own one following the INSTALL instructions in the downloaded autoconf package from `http://www.gnu.org/software/autoconf/`.

On Microsoft Windows you can try to use the "AutoConf for Windows" from `http://gnuwin32.sourceforge.net/packages/autoconf.htm`.

1.3.8 automake (OPTIONAL)

Required to rebuild the autoconf toolchain (Developers only). Latest versions can be obtained from `http://www.gnu.org/software/automake/`.

On Microsoft Windows you can try to use the "AutoMake for Windows" from `http://gnuwin32.sourceforge.net/packages/automake.htm`.

1.3.9 LaTeX(OPTIONAL)

Required to rebuild the technical documentation with Doxygen and the INSTALL.pdf and INSTALL files from INSTALL.tex.

1.3.10 Doxygen (OPTIONAL)

Required to rebuild source documentation.

1.3.11 Graphviz (OPTIONAL)

Required to rebuild source documentation.

1.3.12 TeX4ht (OPTIONAL)

Required to rebuild the INSTALL text file (Developers only).

1.3.13 w3m (OPTIONAL)

Required to rebuild the INSTALL text file (Developers only).

1.4 Documentation

I'm sorry to tell you that beside some accompanying common files and the rare source code documentation there is no documentation available. This might change by and by.

2 Installation

2.1 Building from a Tarball

Download the most recent (stable) distribution tarball from http://www.smart-snmpd.org/.

Unpack it, change the directory to smart-snmpd-nagios-plugins-$VERSION and use the standard GNU procedure to build:

```
  $ ./configure
  $ make
  $ sudo make install
```

Optionally you can specify following options to configure:

```
Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print `checking...' messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for `--cache-file=config.cache'
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or `..']

Installation directories:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [PREFIX]

By default, `make install' will install all the files in
`/usr/local/bin', `/usr/local/lib' etc.  You can specify
an installation prefix other than `/usr/local' using `--prefix',
for instance `--prefix=$HOME'.

For better control, use the options below.

Fine tuning of the installation directories:
  --bindir=DIR            user executables [EPREFIX/bin]
  --sbindir=DIR           system admin executables [EPREFIX/sbin]
  --libexecdir=DIR        program executables [EPREFIX/libexec]
  --sysconfdir=DIR        read-only single-machine data [PREFIX/etc]
  --sharedstatedir=DIR    modifiable architecture-independent data [PREFIX/com]
  --localstatedir=DIR     modifiable single-machine data [PREFIX/var]
  --libdir=DIR            object code libraries [EPREFIX/lib]
  --includedir=DIR        C header files [PREFIX/include]
  --oldincludedir=DIR     C header files for non-gcc [/usr/include]
  --datarootdir=DIR       read-only arch.-independent data root [PREFIX/share]
  --datadir=DIR           read-only architecture-independent data [DATAROOTDIR]
  --infodir=DIR          info documentation [DATAROOTDIR/info]
  --localedir=DIR        locale-dependent data [DATAROOTDIR/locale]
  --mandir=DIR           man documentation [DATAROOTDIR/man]
  --docdir=DIR           documentation root
                         [DATAROOTDIR/doc/smart-snmpd-nagios-plugins]
  --htmldir=DIR          html documentation [DOCDIR]
  --dvidir=DIR           dvi documentation [DOCDIR]
  --pdfdir=DIR           pdf documentation [DOCDIR]
  --psdir=DIR            ps documentation [DOCDIR]

Program names:
  --program-prefix=PREFIX            prepend PREFIX to installed program names
  --program-suffix=SUFFIX            append SUFFIX to installed program names
  --program-transform-name=PROGRAM   run sed PROGRAM on installed program names

System types:
  --build=BUILD     configure for building on BUILD [guessed]
  --host=HOST       cross-compile to build programs to run on HOST [BUILD]

Optional Features:
  --disable-option-checking  ignore unrecognized --enable/--with options
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-maintainer-mode  enable make rules and dependencies not useful
  (and sometimes confusing) to the casual installer
  --disable-dependency-tracking  speeds up one-time build
  --enable-dependency-tracking   do not reject slow dependency extractors
  --enable-shared[=PKGS]  build shared libraries [default=yes]
  --enable-static[=PKGS]  build static libraries [default=yes]
  --enable-fast-install[=PKGS]
                          optimize for fast installation [default=yes]
  --disable-libtool-lock  avoid locking (might break parallel builds)
  --disable-debug         disable support for debugging output
  --enable-namespace      enable using of namespace (default: on if supported)
  --disable-namespace     disable using of namespace
  --disable-rpath         do not hardcode runtime library paths
  --enable-tests          enable test building (default in maintainer-mode)
  --disable-tests         disable test building (default in all other cases)

Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-pic              try to use only PIC/non-PIC objects [default=use
                          both]
  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]

  --with-snmp                 will check for snmp++
  --without-snmp              will not check for snmp++
  --with-libsnmp-prefix[=DIR] search for snmp++ in DIR/include and DIR/lib
  --without-libsnmp-prefix    search for snmp++ in DIR/include and DIR/lib
  --with-log4cplus                  will check for log4cplus
  --without-log4cplus               will not check for log4cplus
  --with-liblog4cplus-prefix[=DIR]  search for log4cplus in DIR/include and DIR/lib
  --without-liblog4cplus-prefix     search for log4cplus in DIR/include and DIR/lib

  --with-boost                  will check for boost
  --without-boost               will not check for boost
  --with-libboost-prefix[=DIR]  search for boost in DIR/include and DIR/lib
  --without-libboost-prefix     search for boost in DIR/include and DIR/lib
  --with-gnu-ld           assume the C compiler uses GNU ld default=no
  --with-perl5[=/path/to/perl5]
  --without-perl5
  --with-remote-cfg=file  use given file to expand additional placeholders
                          in remote_test.cfg
  --with-smooth-cfg=file  use given file to expand additional placeholders
                          in smooth_test.cfg
  --with-snmpd-cfg=file   use given file to expand additional placeholders
                          in snmpd_test.cfg

Some influential environment variables:
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  LIBS        libraries to pass to the linker, e.g. -l<library>
  CPPFLAGS    (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if
              you have headers in a nonstandard directory <include dir>
  CXX         C++ compiler command
  CXXFLAGS    C++ compiler flags
  CPP         C preprocessor
  CXXCPP      C++ preprocessor
  PKG_CONFIG  path to pkg-config utility
  snmp_CFLAGS C compiler flags for snmp, overriding pkg-config
  snmp_LIBS   linker flags for snmp, overriding pkg-config
  log4cplus_CFLAGS
              C compiler flags for log4cplus, overriding pkg-config
  log4cplus_LIBS
              linker flags for log4cplus, overriding pkg-config

Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.
```

2.2 Building the Lastest Source

Assuming you've already checked out a copy from the smart-snmpd-nagios-plugins repository trunk or any branch and starting with the current working directory is the source directory, you need at first create or update your GNU configure toolchain:

```
$ autoreconf --install
```

All further steps are similar to "Building from a Tarball" (see Section 2.1).

If you want to re-start clean, you can do a

```
$ make distclean
```

and start over to the begin of this section.

3 Setting up Nagios using the Smart-Snmpd Nagios Plugins

Build and install the smart-snmpd nagios plugins as any other nagios-plugin. Setup you nagios/etc/.../commands.cfg to use the newly installed plugins.

4 Command Line Interaction

The available command line arguments are displayed when any check of the `smart-snmpd nagios plugins` is called using the "-h" switch.

Following command line arguments are taken by all plugins:

4.1 General options

--help produce an individual help message for the plugin, including predefined default values

--version output the version number and exits

--debug-level increase verbosity level of logging (when logging was enabled in dependency snmp++)

--alarm-timeout sets an alarm timeout to end the plugin after a defined period of time. **Note**: a value of 0 disables the alarm timeout.

--show-performance-data boolean value whether performance data shall be printed out or not

4.2 SNMP options

4.2.1 General SNMP Options

--host host name or ip-address of the machine/service to query

--port port number on which the snmpd is listening

--snmp-version snmp protocol version to use

--timeout time in seconds waiting for responses from snmp daemon

--retries amount of retries to send before giving up

4.2.2 SNMP V1/V2 options

--community snmp community name

4.2.3 SNMP V3 options

--auth-protocol authentication protocol to use

--auth-password authentication password to identify

--priv-protocol privacy protocol to use

--priv-password privacy password to identify

--security-name security name

--security-level security level

--context-name context name

4.3 Check options

--snmpd-type name of the snmpd type to query, if auto-detection isn't suitable

Additional required check options are usually defined by each plugin. See the help output and man-pages for details.

5 Running Tests

The smart-snmpd nagios plugins are coming with four tests:

local test against a locally running smart-snmpd

remote test against remote running smart-snmpd or net-snmpd

smooth run default nagios plugins for net-snmpd and their smart-snmpd nagios plugins counterparts and compare the results

snmpd run the smart-snmpd nagios plugins against a net-snmpd and a smart-snmpd running on the same machine on different ports and compare the results

Note that all tests except the local test may need special firewall rules to allow queries from the machine running the plugin-tests. Further, the tests need locally adapted configuration for remote addresses, communities etc. Because the tests are written in Perl5, any Perl5 accessible database or pseudo-database will do fine (the authors are using CSV files via DBD::CSV).

6 Platform Specific Issues

6.1 Windows NT

6.2 VMS

6.3 MacOS X

7 Language Bindings

Future Task