Installation Guide for Smart-Snmpd

Jens Rehsack

Contents

1 Introduction

1.1 Audience

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

If you are neither of them, it is strongly recommended you find an appropriate binary package.

For those who still want to build from source (or probably just have to), smart-snmpd follows the "./configure && make" philosophy introduced by open source packages for Unices and compatibles. Keep in mind that smart-snmpd has a lot of dependencies, and some of the dependencies have further dependencies.

1.2 Dependency Overview

1.2.1 Tools

The smart-snmpd build infrastructure needs the following tools to compile the daemon:

- 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
- agent++ 3.5.32 or later, probably not agent++ 4.0
- libstatgrab 0.18 or later
- libConfuse 2.8 or later
- log4cplus 1.0.5 or later
- libjson 0.10 or later

1.2.3 Maintainer Tools

- autoconf 2.63 or later
- automake 1.11
- LaTeX

- Doxygen
- Graphviz
- TEX4ht
- w3m

1.3 Dependencies in Detail

1.3.1 libtool 2.2 or later (REQUIRED)

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

1.3.2 pkg-config 0.20 or later (RECOMMENDED)

Recommended to configure smart-snmpd.

You can avoid this prerequisite and achieve similar results by setting the environment variables

- `snmp_CFLAGS, snmp_LIBS`
- `agent_CFLAGS, agent_LIBS`
- `confuse_CFLAGS, confuse_LIBS`
- `log4cplus_CFLAGS, log4cplus_LIBS`
- `json_CFLAGS, json_LIBS`
- `statgrab_CFLAGS, statgrab_LIBS`

to appropriate values. If *pkg-config* can't be found, the `./configuration` script of *smart-snmpd* searches in typical installation directories (in addition to the specified prefix and search path) for runtime prerequisites.

1.3.3 A Reasonable C++ Compiler (REQUIRED)

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

1.3.4 a POSIX.2001 compatible runtime environment (REQUIRED)

`smart-snmpd` uses features from the X/Open CAE Specification, Issue 5 (XPG5/UNIX 98/SUSv2) (`_XOPEN_SOURCE >= 500`). Your environment may not satisfy the specific requirements and currently there are no distinct checks whether the standard is fully supported. If your environment does not feature POSIX.2001 (look at the warnings `./configure` emits), run all the tests and report any problems found.

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

The library *snmp++* version 3.2.26 is mandatory for smart-snmpd's SNMPv3 features.

snmp++ must have been built using its autoconf toolchain with SNMPv3 support enabled. It is strongly recommended that logging and namespace support is enabled too.

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

1.3.6 agent++ 3.5.32 or later, probably not agent++ 4.0 (REQUIRED)

The library *agent++* version 3.5.32 is mandatory for smart-snmpd's SNMPv3 features and MIB management.

agent++ must have been built using its autoconf toolchain with thread support enabled. It is strongly recommended that the thread-pool feature is enabled too.

You can download the latest release of agent++ from its primary download site at `http://www.agentpp.com/`.

1.3.7 libstatgrab 0.18 or later (REQUIRED)

The library libstatgrab 0.18 is required, because it is the first release which supports the reentrant calls to grab system stats.

Consequently it is required that the provided libstatgrab has been built with support for multi-threading applications.

You can download the latest release of libstatgrab from its primary download site at `http://www.i-scream.org/`.

1.3.8 libConfuse 2.8 or later (REQUIRED)

The libConfuse library is required at least at version 2.8, because this version contains some C++ compatibility patches.

You can download the latest release of libConfuse from its project page at `http://savannah.nongnu.org/projects/confuse/`.

1.3.9 log4cplus 1.0.5 or later (OPTIONAL)

The *log4cplus* library enhances the logging capabilities dramatically. While the current 1.1.0 snapshots also support a C-API to allow the also required *libstatgrab*, *Smart-SNMPd* can be satisfied with the version 1.0.5 of *log4cplus*.

You can download the latest release of log4cplus from its project page at `http://log4cplus.sourceforge.net/`.

1.3.10 libjson 0.9 or later (OPTIONAL)

The libjson library needs to be at least version 0.9 because this version contains some C++ compatibility patches. You can use the bundled version (use the `--with-bundled-libjson` option when configuring).

Building without libjson disables MIBs from external commands.

You can download the latest release of libjson from the page of its developer at `http://projects.snarc.org/` or from his github repository.

1.3.11 autoconf 2.63 or later (MAINTAINER)

On Unix and compatible systems you can either use the autoconf provided by your manufacturer or distributor (if current enough) or install your 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`.

This dependency is required to rebuild the configure and build toolchain (Developers and Packagers only).

1.3.12 automake 1.11 or later (MAINTAINER)

This dependency is required to rebuild the configure and build toolchain (Developers and Packagers only).

1.3.13 LaTeX(MAINTAINER, OPTIONAL)

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

1.3.14 Doxygen (OPTIONAL)

Required to rebuild source documentation.

1.3.15 Graphviz (MAINTAINER, OPTIONAL)

Required to rebuild source documentation.

1.3.16 TeX4ht (MAINTAINER, OPTIONAL)

Required to rebuild the INSTALL and OPERATION text files (Developers only).

1.3.17 w3m (MAINTAINER, OPTIONAL)

Required to rebuild the INSTALL and OPERATION text files (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 may change in the future.

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-$VERSION and use the standard GNU procedure to build:

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

Optionally you can specify the 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-snmp]
  --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
  --disable-threads       disable thread support
```

```
  --disable-debug         disable support for debugging output
  --enable-docbuild       enable build of documentation
  --disable-rpath         do not hardcode runtime library paths
  --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)

Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=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-agent                 will check for agent++
  --without-agent              will not check for agent++
  --with-libagent-prefix[=DIR] search for agent++ in DIR/include and DIR/lib
  --without-libagent-prefix    search for agent++ in DIR/include and DIR/lib

  --with-confuse                 will check for confuse
  --without-confuse              will not check for confuse
  --with-libconfuse-prefix[=DIR] search for confuse in DIR/include and DIR/lib
  --without-libconfuse-prefix    search for confuse in DIR/include and DIR/lib

  --with-json                will check for json
  --without-json             will not check for json
  --with-libjson-prefix[=DIR] search for json in DIR/include and DIR/lib
  --without-libjson-prefix    search for json in DIR/include and DIR/lib
  --with-bundled-libjson  uses bundled libjson instead of external library

  --with-statgrab                 will check for statgrab
  --without-statgrab              will not check for statgrab
  --with-libstatgrab-prefix[=DIR] search for statgrab in DIR/include and DIR/lib
  --without-libstatgrab-prefix    search for statgrab 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-su-cmd[=ARG]     use su-cmd (default: search for sudo and su)
  --without-su-cmd        do not use an su-cmd to switch user for external commands
  --with-su-args[=ARG]    use su-args (default: choose reasonable flags depending
                          on su-cmd)
  --with-v1-community[=ARG]
                          use SNMP v1 community ARG (default: public)
  --without-v1-community  do not use SNMP v1 community ARG (default: public)
  --with-v2-community[=ARG]
                          use SNMP v2 community ARG (default: public)
  --without-v2-community  do not use SNMP v2c community ARG (default: public)
  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]
  --with-gnu-ld           assume the C compiler uses GNU ld default=no
  --with-pic              try to use only PIC/non-PIC objects [default=use
                          both]
```

```
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
  agent_CFLAGS
              C compiler flags for agent, overriding pkg-config
  agent_LIBS  linker flags for agent, overriding pkg-config
  statgrab_CFLAGS
              C compiler flags for statgrab, overriding pkg-config
  statgrab_LIBS
              linker flags for statgrab, overriding pkg-config
  confuse_CFLAGS
              C compiler flags for confuse, overriding pkg-config
  confuse_LIBS
              linker flags for confuse, overriding pkg-config
  log4cplus_CFLAGS
              C compiler flags for log4cplus, overriding pkg-config
  log4cplus_LIBS
              linker flags for log4cplus, overriding pkg-config
  json_CFLAGS C compiler flags for json, overriding pkg-config
  json_LIBS   linker flags for json, 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

This assumes you already have a copy of smart-snmpd, either from the trunk of the smart-snmpd repository or some interesting branch. Also, your current working directory should be the root directory of the checked out copy. The very first step is to create the GNU configure toolchain with:

```
$ 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 go back to the beginning of this section.

If you want to run a smart-snmpd within the build environment, you should carefully read "Setting up a Smart-Snmpd in the Build Environment" (Section **??**).

3 Platform Specific Issues

3.1 Windows NT

Unsupported.

3.2 VMS

Unsupported.

3.3 MacOS X

Partially working.

4 Language Bindings

Future Task