

Software Collections

(packaging draft)

Jindřich Nový, jnovy@redhat.com

February 23, 2012

Copyright © 2012 Jindřich Nový

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Agenda

1 Software Collections

- Filesystem hierarchy layout
- How to use SCL
- SCL meta package
- SCL scriptlets

2 SCL packaging

- SCL packaging macros
- Conversion How-to
- Why a SCL meta-package is needed?
- SCL build in koji/brew
- Special cases when packaging a SCL

Section 1

Software Collections

Software Collections in Fedora/EPEL

- consist of two basic packages

Runtime utility for running Software Collection applications

```
# yum install scl-utils
```

Build macros to build Software Collections

```
# yum install scl-utils-build
```

- present in Fedora 15, 16, rawhide, EPEL5, EPEL6

SCL highlights and features

- no need to do any RPM code modification on a host system
- minimal spec file modifications to convert a package to SCL
- allows to build an unmodified spec as a normal package
- allows to build an unmodified spec into different SCL
- uses SCL namespacing
 - unique name for any package in collection
- solves concurrent SCL update problems
 - there no longer exist update conflicts due to SCL package namespacing
- inter-SCL dependencies
 - allows to implement multiple levels of SCLs

SCL filesystem hierarchy

Filesystem hierarchy layout

```
/opt/rh/           - configurable via %_scl_prefix
  Collection1/
    <arch>/
      root/
      enable
      <Collection1 scriptlets>
  Collection2/
    ...
```

How to enable a SCL?

- `scl` tool is used to do it for us

Tool synopsis

```
$ scl <action> [<SCL1>, <SCL2> ...] <command>
```

Example of `scl` tool invocation

```
$ scl enable example_scl 'perl --version'
```

- it is possible to run shell with SCL enable, after *Ctrl-D* we are back in untouched system environment
- one can use a wrapper script to simplify execution of a SCL application

SCL packaging layout

- SCL meta package
 - **scl_name** - main SCL package shipping base package set
 - **scl_name-runtime** - package shipping scriptlets and owns SCL filesystem
 - **scl_name-build** - package shipping SCL build configuration (not mandatory)
- SCL packages
 - **scl_name_pkgname** - SCL namespaced and relocated packages

What is SCL scriptlet?

- a simple shell script that changes current environment to prefer SCL package set over a system package set
- currently only `enable` scriptlet is required
- **scl** tool is an interface to use these scriptlets

Section 2

SCL packaging

How a system and SCL package build differ?

Normal system package local build

```
$ rpmbuild -bb package.spec
```

SCL package local build

```
$ rpmbuild -bb package.spec --define 'scl <name>'
```

What SCL packaging macro set does?

- relocates file hierarchy to SCL-exclusive filesystem
- defines convenience macros for packagers
- defines file ownerships for the main meta package

Which macros to use in SCL environment?

- SCL specific macros usage need to be prefixed with `%{?scl: ... }`
- **%scl_name** - name of the SCL, e.g. `my_collection`
- **%pkg_name** - original package name, e.g. `ruby`
- **%_scl_prefix** - SCL prefix, e.g. `/opt/rh`
 - can be redefined
- **%_scl_scripts** - where SCL scriptlets are, e.g. `/opt/rh/my_collection`
- **%_scl_root** - package root for a SCL, e.g. `/opt/rh/my_collection/root`

Which macros to use in SCL environment?

- all path macros which are not pointing to SCL filesystem are prefixed with `_root`:
 - `%_root_prefix` ⇒ `/usr`
 - `%_root_bindir` ⇒ `/usr/bin`
 - `%_root_datadir` ⇒ `/usr/share`
 - `%_root_sysconfdir` ⇒ `/etc`
 - `%_root_includedir` ⇒ `/usr/include`
 - ...

How do I convert ordinary spec to SCL?

```
+{%?scl:Zscl_package less}
+
|Summary: A text file browser similar to more, but better
-Name: less
+Name: {%?scl_prefix}less
|Version: 443
|Release: 1{%?dist}
|License: GPLv3+
|Group: Applications/Text
-Source: http://www.greenwoodsoftware.com/less/{name}-{version}.tar.gz
+Source: http://www.greenwoodsoftware.com/less/less-{version}.tar.gz
|Source1: lesspipe.sh
|Source2: less.sh
|Source3: less.csh
@@ -19,6 +21,7 @@
|BuildRequires: ncurses-devel
|BuildRequires: pcre-devel
|BuildRequires: autoconf automake libtool
+{%?scl:Requires:Zscl_runtime}
|
|Zdescription
|The less utility is a text file browser that resembles more, but has
@@ -31,7 +34,7 @@
|files, and you'll use it frequently.
|
|Zprep
-|Zsetup -q
+|Zsetup -q {%?scl:-n %?pkg_name%}-{version}
|Zpatch1 -p1 -b .foption
|Zpatch4 -p1 -b .time
|Zpatch5 -p1 -b .fsync
@@ -50,17 +53,18 @@
|Zfiles
|Zdefattr(-,root,root,-)
|Zdoc LICENSE
-|Zetc/profile.d/*
+|Z_sysconfdir{/profile.d/*}
|Z{bindir}/*
|Z{mandir}/man1/*
```

How do I convert ordinary spec to SCL?

- `scl` macro definition needs to be added before package preamble:
 - `%{?scl:%scl_package package_name}`
- Name tag needs to be modified to
 - Name: `%{?scl_prefix}package_name`
- all essential SCL packages should be dependent on main meta package:
 - `%{?scl:Requires: %scl_runtime}`
- `%setup` macro needs to deal with different package name in SCL environment:
 - `%setup -q %{?scl:-n %{pkg_name}-%{version}}`

How should I install a SCL?

- SCL is installed via the main meta package named **scl_name** which contains dependencies to basic SCL package set (i.e. no optional packages)
 - **yum install scl_name**
- Every package in SCL depends on **scl_name-runtime** which contains:
 - base filesystem structure
 - SCL scriptlets
 - optional SCL configuration files

SCL build in koji/brew

- simple approach - defining the default SCL name in spec
 - i.e. `%{?scl:%global scl scl_name}` is the first line in spec
 - SCL packages can be built directly in koji/brew if they contain default SCL name definition in every spec
- complex approach
 - SCL can be built via **chain-build**, building SCL meta package first
 - SCL packages can be built as soon as meta-package **scl_name-build** can is into the buildroot
 - build requirement from a SCL package to **scl_name-runtime** assures all build requirements are met

How SCL meta-package looks like?

```

%{?scl:%global scl example}
%{?scl_package: %scl}

Summary: Package that installs %scl
Name: %scl_name
Version: 1
Release: 1%{?dist}
BuildArch: noarch
License: GPLv2+
Group: Applications/File
Buildroot: %toppath/%{name}-%{version}-%{release}-root-%{?id,u} -n
Requires: %scl_prefix!less

%description
This is the main package for %scl Software Collection.

%package runtime
Summary: Package that handles %scl Software Collection.
Group: Applications/File
Requires: scl-utils

%description runtime
Package shipping essential scripts to work with %scl Software Collection.

%package build
Summary: Package shipping basic build configuration
Group: Applications/File

%description build
Package shipping essential configuration macros to build %scl Software Collection.

%install
rm -rf %buildroot
mkdir -p %buildroot/%scl_scripts/root
cat >> %buildroot/%scl_scripts/enable << EOF
export PATH=%bindir:%PATH
EOF
%{?scl_install}

%files
%files runtime
%{?scl_files}

%files build
%{?root_sysconfdir}/rpm/macros.%scl-config

%changelog
* Thu Jan 07 2011 Jindrich Novy <jnovy@redhat.com> 1-1
- initial packaging

```

Special cases when packaging a SCL

- libraries
 - `%{_root_sysconfdir}/ld.so.conf.d/%{scl_prefix}lib.conf`
- initscripts
 - `%{_root_sysconfdir}/rc.d/%{stack_prefix}service_name`
- manpath
 - put MANPATH enablement script to
`%{_root_sysconfdir}/profile.d/%{scl_prefix}manpages.sh`
- cronjobs
- logrotate
- locks
- kernel modules

References



SCL macros and utilities:

<http://jnovy.fedorapeople.org/scl-utils/>



This presentation:

[http:](http://jnovy.fedorapeople.org/scl-utils/scl.pp.pdf)

[//jnovy.fedorapeople.org/scl-utils/scl.pp.pdf](http://jnovy.fedorapeople.org/scl-utils/scl.pp.pdf)

Questions.

Thanks for listening.