Red Hat Software Collections 1.x 1.2 Release Notes

Release Notes for Red Hat Software Collections 1.2

Lenka Špačková Jaromír Hradílek Eliška Slobodová

Red Hat Software Collections 1.x 1.2 Release Notes

Release Notes for Red Hat Software Collections 1.2

Lenka Špačková
Red Hat Customer Content Services
lspackova@redhat.com

Jaromír Hradílek
Red Hat Customer Content Services
jhradilek@redhat.com

Eliška Slobodová
Red Hat Customer Content Services

Legal Notice

Abstract

The Red Hat Software Collections 1.2 Release Notes document the major features and contain important information about known problems in Red Hat Software Collections 1.2. The Red Hat Developer Toolset collection is documented in the Red Hat Developer Toolset Release Notes and the Red Hat Developer Toolset User Guide.

Table of Contents

Chapter 1. Red Hat Software Collections 1.2

This chapter serves as an overview of the Red Hat Software Collections 1.2 content set. It sums up its main features, provides a list of components and their descriptions, compatibility information, and a list of known issues.

1.1. About Red Hat Software Collections

For certain applications, more recent versions of some software components are often needed in order to use their latest new features. **Red Hat Software Collections** is a Red Hat offering that provides a set of dynamic programming languages, database servers, and various related packages that are either more recent than their equivalent versions included in the base Red Hat Enterprise Linux system, or are available for this system for the first time. For a complete list of components that are distributed as part of Red Hat Software Collections and a brief summary of their features, see Section 1.2, “Main Features”.

Red Hat Software Collections does not replace the default system tools provided with Red Hat Enterprise Linux 6 or Red Hat Enterprise Linux 7. Instead, a parallel set of tools is installed in the `/opt/` directory and can be optionally enabled per application by the user using the supplied `scl` utility. The default versions of Perl or PostgreSQL, for example, remain those provided by the base Red Hat Enterprise Linux system.

With the notable exception of **Node.js**, all Red Hat Software Collections components are fully supported under Red Hat Enterprise Linux Subscription Level Agreements, are functionally complete, and are intended for production use. Important bug fix and security errata are issued to Red Hat Software Collections subscribers in a similar manner to Red Hat Enterprise Linux for at least three years from the release of each major version. A new major version of Red Hat Software Collections is released approximately every 18 months, and in each major release stream, each version of a selected component remains backward compatible.

Red Hat Developer Toolset is now part of Red Hat Software Collections, included as a separate Software Collection. For more information about Red Hat Developer Toolset, refer to the Red Hat Developer Toolset Release Notes and the Red Hat Developer Toolset User Guide.

1.2. Main Features

Red Hat Software Collections 1.2 provides recent stable versions of the tools listed in Table 1.1, “Red Hat Software Collections 1.2 Components”.

Table 1.1. Red Hat Software Collections 1.2 Components

Component	Software Collection	Description
Red Hat Developer Toolset 3.0	*devtoolset-3*	Red Hat Developer Toolset is designed for developers working on the Red Hat Enterprise Linux platform. It provides current versions of the **GNU Compiler Collection**, **GNU Debugger**, **Eclipse** development platform, and other development, debugging, and performance monitoring tools. For a complete list of components, see the Red Hat Developer Toolset Components table in the *Red Hat Developer Toolset User Guide*.

Component	Software Collection	Description
Perl 5.16.3	*perl516*	A release of Perl with a number of additional utilities, scripts, and *database connectors for MySQL and PostgreSQL*. This version provides a large number of new features and enhancements, including new debugging options, improved Unicode support, and better performance. Also, it adds **perl-DateTime** and **mod_perl**, which is supported only with the *httpd24* Software Collection package.
PHP 5.4.16	*php54*	A release of PHP with **PEAR 1.9.4** and a number of additional extensions. PHP 5.4 provides a number of *language and interface improvements*. The **APC**, **memcache**, and **Zend OPcache** extensions are also included.
PHP 5.5.6	*php55*	A release of PHP with *enhanced language features* including better exception handling, generators, and **Zend OPcache**. The **memcache** and **mongodb** extensions are also included.
Python 2.7.5	*python27*	A release of Python 2.7 with a number of additional utilities. This Python version provides various new features and enhancements, including a new ordered dictionary type, faster I/O operations, and improved forward compatibility with Python 3. The *python27* Software Collections contains the *Python 2.7.5 interpreter*, a set of extension libraries useful for programming web applications and **mod_wsgi** (only supported with the *httpd24* Software Collection), MySQL and PostgreSQL database connectors, and **numpy** and **scipy**.
Python 3.3.2	*python33*	A release of Python 3 with a number of additional utilities. This Software Collection gives developers on Red Hat Enterprise Linux access to Python 3 and allows them to benefit from various advantages and new features of this version. The *python33* Software Collection contains *Python 3.3.2 interpreter*, a set of extension libraries useful for programming web applications and **mod_wsgi** (only supported with the *httpd24* Software Collection), PostgreSQL database connector, and **numpy** and **scipy**.
Ruby 1.9.3 [a]	*ruby193*	A release of Ruby 1.9.3 and **Ruby on Rails 3.2.8** with *a large collection of Ruby gems*. This Software Collection gives developers on Red Hat Enterprise Linux access to Ruby 1.9, which provides a number of new features and enhancements, including improved Unicode support, enhanced threading, faster load times, and **mod_passenger**, which is supported only with the *httpd24* Software Collection package.
Ruby 2.0.0	*ruby200*	A release of Ruby 2.0.0. This version provides substantial performance and reliability improvements and includes a number of new features and improved debugging capabilities, while maintaining source level backward compatibility with Ruby 1.9.3.

Component	Software Collection	Description
Ruby on Rails 4.0.2[a]	*ror40*	A release of **Ruby on Rails 4.0**, a web application development framework written in the Ruby language. This version provides a number of new features and improvements and adds live streaming for persistent connections. This Software Collection is supported together with the *ruby200* collection.
MariaDB 5.5.37	*mariadb55*	A release of MariaDB, *an alternative to MySQL* for users of Red Hat Enterprise Linux. MySQL is binary compatible with MariaDB and can be replaced with it without any data conversions. This version adds the PAM authentication plugin to MariaDB.
MongoDB 2.4.9 [b]	*mongodb24*	A release of MongoDB, a cross-platform *document-oriented database system classified as a NoSQL database*. This Software Collection includes the *mongo-java-driver* package.
MySQL 5.5.37	*mysql55*	A release of MySQL, which provides a number of new features and enhancements, including improved performance.
PostgreSQL 9.2.8	*postgresql92*	A release of PostgreSQL, which provides a number of new features and enhancements, including *cascading replication, native JSON support,* improved scalability, and better performance.
Node.js 0.10[b] [c]	*nodejs010*	A release of Node.js with **npm 1.3.24**. This Software Collection gives users of Red Hat Enterprise Linux access to this programming platform.
nginx 1.6.1	*nginx16*	A release of nginx, a web and proxy server with a focus on high concurrency, performance and low memory usage. This version introduces a number of new features, including various *SSL improvements*, support for *SPDY 3.1 (limited to Red Hat Enterprise Linux 7), cache revalidation with conditional requests,* and *authentication request module*.
Apache httpd 2.4.6	*httpd24*	A release of the Apache HTTP Server (httpd), including a high performance *event-based processing model, enhanced SSL module and FastCGI support*. The **mod_auth_kerb** module is also included.
Thermostat 1.0.4	*thermostat1*	A release of Thermostat, a monitoring and instrumentation tool for the *OpenJDK HotSpot JVM*, with support for monitoring *multiple JVM instances*. This Software Collection depends on the *mongodb24* component.
Git 1.9.4	*git19*	A release of Git, a *distributed revision control* system with a decentralized architecture. As opposed to centralized version control systems with a client-server model, Git ensures that each working copy of a Git repository is its exact copy with complete revision history.

Component	Software Collection	Description
DevAssistant 0.9.1	*devassist09*	A release of DevAssistant, a tool designed to assist developers with *creating and setting up basic projects* in various programming languages, installing dependencies, setting up a development environment, and working with source control. DevAssistant supports the C, C++, Java, and Python programming languages but it is able to support working with any other language, framework, or tool due to its modular architecture.
Maven 3.0.5	*maven30*	A release of Maven, a *software project management and comprehension tool* used primarily for Java projects. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central piece of information.

[a] A part of this Software Collection requires a JavaScript engine. The *v8314* Software Collection included in Red Hat Software Collections 1.2 provides the V8 JavaScript engine and is supported only as the Software Collection's dependency.
[b] This Software Collection also requires **v8314**. The *v8314* Software Collection included in Red Hat Software Collections 1.2 provides the V8 JavaScript engine and is supported only as the Software Collection's dependency.
[c] In Red Hat Software Collections 1.2, **Node.js** is included as a Technology Preview. For more information about Red Hat Technology Previews, see https://access.redhat.com/support/offerings/techpreview/.

1.3. Changes in Red Hat Software Collections 1.2

1.3.1. New Components

Red Hat Software Collections 1.2 adds these new components:

- *devtoolset-3* — see Section 4.1, “Red Hat Developer Toolset”
- *git19* — see Section 4.5, “Git”
- *devassist09* — see Section 4.6, “DevAssistant”
- *maven30* — see Section 4.7, “Maven”

1.3.2. Changes in nginx

Red Hat Software Collections 1.2 brings a significant change to **nginx**: it has been upgraded to version 1.6.1 and it is now supported. In accordance with this change, the Software Collection has been renamed to *nginx16*. **nginx 1.6.1** introduces a number of new features, including various SSL improvements, support for SPDY 3.1 (limited to Red Hat Enterprise Linux 7), cache revalidation with conditional requests, authentication request module, and more. For a complete list of changes, refer to http://nginx.org/en/CHANGES-1.6.

For information on migrating to the later version, see Section 5.3, “Migrating from nginx 1.4 to nginx 1.6”.

1.3.3. Changes in Ruby on Rails 4.0

In Red Hat Software Collections 1.2, the *ror40-rubygem-jquery-rails* package has been upgraded to version 3.1.0, which contains jQuery JavaScript framework version 1.11.0. For more information about the *ror40* Software Collection, see Section 4.3, “Ruby on Rails 4.0”

1.3.4. Changes in Thermostat

The *thermostat1* Software Collection has been upgraded to version 1.0.4, which provides a number of bug fixes over the previous version. For additional information about using this Software Collection, refer to Section 4.2, “Thermostat 1”

1.4. Compatibility Information

Red Hat Software Collections 1.2 is available for all supported releases of Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 on AMD64 and Intel 64 architectures.

1.5. Known Issues

`nodejs-hawk` component

The *nodejs-hawk* package uses an implementation of the SHA-1 and SHA-256 algorithms adopted from the CryptoJS project. In this release, the client-side JavaScript is obfuscated. The future fix will involve using crypto features directly from the CryptoJS library.

`postgresql` component

The *postgresql92* package for Red Hat Enterprise Linux 6 does not provide the **`sepgsql`** module as this feature requires installation of *libselinux* version 2.0.99, which is not available in Red Hat Enterprise Linux 6.

`coreutils` component

Some utilities, for example, **`su`**, **`login`**, or **`screen`**, do not export environment settings in all cases, which can lead to unexpected results. It is therefore recommended to use **`sudo`** instead of **`su`** and set the **`env_keep`** environment variable in the **`/etc/sudoers`** file. Alternatively, you can run commands in a reverse order; for example:

```
su -l postgres -c "scl enable postgresql92 psql"
```

instead of

```
scl enable postgresql92 bash
su -l postgres -c psql
```

When using tools like **`screen`** or **`login`**, you can use the following command to preserve the environment settings: **`source /opt/rh/<collection_name>/enable`**.

`httpd`, `mariadb`, `mongodb`, `mysql`, `nodejs`, `perl`, `php55`, `python27`, `python33`, `ruby193`, `ror40`, `ruby200`, `thermostat`, and `v8314` components

When uninstalling the *httpd24*, *mariadb55*, *mongodb24*, *mysql55*, *nodejs010*, *perl516*, *php55*, *python27*, *python33*, *ruby193*, *ror40*, *ruby200*, *thermostat1*, or *v8314* packages, the order of uninstalling can be relevant due to ownership of dependent packages. As a consequence, some directories and files might not be removed properly and might remain on the system.

`mariadb`, `mysql`, `postgresql`, `mongodb` components

Red Hat Software Collections contains the **MySQL 5.5**, **MariaDB 5.5**, **PostgreSQL 9.2** and **MongoDB 2.4** databases. The core Red Hat Enterprise Linux 6 provides earlier versions of the **MySQL** and **PostgreSQL** databases (client library and daemon). The core Red Hat Enterprise Linux 7 provides the same versions of the **MariaDB** and **PostgreSQL** databases (client library and daemon). Client libraries are also used in database connectors for dynamic languages, libraries, and so on.

The client library packaged in the Red Hat Software Collections database packages in the **PostgreSQL** component is not supposed to be used, as it is included only for purposes of server utilities and the daemon. Users are instead expected to use the system library and the database connectors provided with the core system.

A protocol, which is used between the client library and the daemon, is stable across database versions, so, for example, using the **PostgreSQL 8.4** client library with the **PostgreSQL 9.2** daemon works as expected.

The core Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 does not include the client library for **MongoDB**. In order to use this client library for your application, you should use the client library from Red Hat Software Collections and always use the **scl enable ...** call every time you run an application linked against this **MongoDB** client library.

mariadb, mysql, mongodb components

MariaDB, MySQL, and MongoDB do not make use of the **/opt/<provider>/<collection>/root** prefix when creating log files. Note that log files are saved in the **/var/log/** directory, not **/opt/<provider>/<collection>/root/var/log/**.

httpd component

Compiling external applications against the Apache Portable Runtime (APR) and APR-util libraries from the *httpd24* Software Collection is not supported. The LD_LIBRARY_PATH is not set in *httpd24* because it is not required by any application in this Software Collection.

httpd, ruby193 components

In Red Hat Enterprise Linux 6.5 and earlier versions, **httpd** is unable to execute the binary files in the **mod_passenger** module, namely **PassengerWatchdog**, **PassengerHelperAgent**, **PassengerLoggingAgent**, and **SpawnPreparer** in the **/opt/rh/ruby193/root/usr/lib64/gems/exts/passenger-4.0.18/agents/** directory. To work around this problem, disable SELinux by running the following command as **root**:

```
setenforce 0
```

nginx component

In Red Hat Enterprise Linux 6.5 and earlier versions, no SELinux policy is applied for the **nginx** daemon.

mariadb component

The permissions for the **/var/log/mariadb55-mariadb/** directory, in which the log file is stored, are set incorrectly. Consequently, when the **/var/log/mariadb55-mariadb/mariadb.log** file is removed in Red Hat Enterprise Linux 7, the service fails to start because the **mysqld** daemon does not have permission to create log files. To work

around this problem, either do not remove the log file or change the owner of the **/var/log/mariadb55-mariadb/** directory to **mysql:mysql**.

mysql, mariadb components

In Red Hat Enterprise Linux 7, the **mariadb55-mariadb** and **mysql55-mysqld** services run under the **mysql** user account by default. When the **/var/lib/mysql/mysql.sock** UNIX socket file is created by a different user, the services have insufficient permissions to check whether a process is listening on the socket but the services have sufficient permissions to delete the socket file. Consequently, the **mariadb55-mariadb** and **mysql55-mysqld** services can delete the socket file while a process is still using it.

perl component

In Red Hat Enterprise Linux 7, the *perl516* Software Collection tapset conflicts with the core system tapset. As a consequence, the **systemtap** utility does not work correctly for *perl516*. To work around this problem:

- Either copy the *perl516* tapset to a file renamed by adding the Software Collection's prefix - to do so, use the following command:

```
cp /opt/rh/perl516/root/usr/share/systemtap/tapset/libperl5.16.3-64.stp /opt/rh/perl516/root/usr/share/systemtap/tapset/perl516-libperl5.16.3-64.stp
```

- Or uninstall the core system *perl-devel* package.

python27 component

In Red Hat Enterprise Linux 7, when the user tries to install the *python27-python-debuginfo* package, the **/usr/src/debug/Python-2.7.5/Modules/socketmodule.c** file conflicts with the corresponding file from the *python-debuginfo* package installed on the core system. Consequently, installation of the *python27-python-debuginfo* fails. To work around this problem, uninstall the *python-debuginfo* package and then install the *python27-python-debuginfo* package.

devassist component

When the user tries to rebuild the *devassist09-PyYAML* package on Red Hat Enterprise Linux 6, the build fails due to a soft dependency, if the Pyrex or Cython programming languages are detected. To work around this problem, make sure the *pyrex* or *cython* packages are not installed on your system.

Other Notes

php54 component

Note that **Alternative PHP Cache (APC)** in Red Hat Software Collections 1.2 is provided for user data cache only. For opcode cache, **Zend OPcache** is provided.

nodejs component

The *nodejs-tobi-cookie* package has been renamed to *nodejs-cookie-jar*.

ruby component

Previously, in Red Hat Software Collections 1.0, the V8 JavaScript engine was part of the *ruby193* Software Collection. Since Red Hat Software Collections 1.1, the *v8* packages have been replaced by the *v8314* Software Collection, which is installed as a dependency. In order to use **therubyracer**, it is necessary to enable the *v8314* Software Collection as well.

nodejs component

Previously, in Red Hat Software Collections 1.0, the V8 JavaScript engine was part of the *nodejs010* Software Collection. Since Red Hat Software Collections 1.1, the *v8* packages have been replaced by the *v8314* Software Collection, which is installed as a dependency.

python component

When the user tries to install both the *python27-scldevel* and *python33-scldevel* packages, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**%scl_python**, **%scl_prefix_python**).

php component

When the user tries to install both the *php54-scldevel* and *php55-scldevel* packages, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**%scl_php**, **%scl_prefix_php**).

ruby component

When the user tries to install both the *ruby193-scldevel* and *ruby200-scldevel* packages, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**%scl_ruby**, **%scl_prefix_ruby**).

libyaml component

A newer version of the *libyaml* package is provided with Red Hat Software Collections 1.2 for use by Red Hat Software Collections.

nodejs component

When installing the *nodejs010* Software Collection, *nodejs010* installs **GCC** in the base Red Hat Enterprise Linux system as a dependency, unless the *gcc* packages are already installed.

mariadb component

In Red Hat Software Collections for Red Hat Enterprise Linux 7, since version 1.1, the **mariadb55-mysqld.service** file has been renamed to **mariadb55-mariadb.service** and the **/var/log/mariadb55-mysqld.log** file has been moved to **/var/log/mariadb55-mariadb/mariadb55-mariadb.log**, for the sake of consistency with Red Hat Enterprise Linux 7.

Chapter 2. Installation

This chapter describes in detail how to get access to the content set, install Red Hat Software Collections 1.2 on the system, and rebuild Red Hat Software Collections.

2.1. Getting Access to Red Hat Software Collections

Depending on the subscription management service with which you registered your Red Hat Enterprise Linux system, you can either enable Red Hat Software Collections by using Red Hat Subscription Management, or by using RHN Classic. For detailed instructions on how to enable Red Hat Software Collections using RHN Classic or Red Hat Subscription Management, see the respective section below. For information on how to register your system with one of these subscription management services, see Using and Configuring Red Hat Subscription Manager.

Important

If you are running a version of Red Hat Enterprise Linux prior to 6.4, you will be unable to download Red Hat Software Collections through Red Hat Subscription Management. To obtain Red Hat Software Collections, either update to Red Hat Enterprise Linux 6.4, or register your system with RHN Classic. For more information, see https://access.redhat.com/solutions/129003.

2.1.1. Using Red Hat Subscription Management

If your system is registered with Red Hat Subscription Management, complete the following steps to attach the subscription that provides access to the repository for Red Hat Software Collections and enable the repository:

1. Display a list of all subscriptions that are available for your system and determine the pool ID of a subscription that provides Red Hat Software Collections. To do so, type the following at a shell prompt as **root**:

   ```
   subscription-manager list --available
   ```

   For each available subscription, this command displays its name, unique identifier, expiration date, and other details related to it. The pool ID is listed on a line beginning with **Pool Id**.

2. Attach the appropriate subscription to your system by running the following command as **root**:

   ```
   subscription-manager subscribe --pool=pool_id
   ```

   Replace *pool_id* with the pool ID you determined in the previous step. To verify the list of subscriptions your system has currently attached, type as **root**:

   ```
   subscription-manager list --consumed
   ```

3. Display the list of available Yum list repositories to retrieve repository metadata and determine the exact name of the Red Hat Software Collections repositories. As **root**, type:

```
yum repolist all
```

The repository names depend on the specific version of Red Hat Enterprise Linux you are using and are in the following format:

```
rhel-variant-rhscl-6-rpms
rhel-variant-rhscl-6-debug-rpms
rhel-variant-rhscl-6-source-rpms

rhel-server-rhscl-6-eus-rpms
rhel-server-rhscl-6-eus-source-rpms
rhel-server-rhscl-6-eus-debug-rpms

rhel-variant-rhscl-7-rpms
rhel-variant-rhscl-7-debug-rpms
rhel-variant-rhscl-7-source-rpms
```

Replace *variant* with the Red Hat Enterprise Linux system variant, that is, **server** or **workstation**. Note that Red Hat Software Collections is supported neither on the **Client** nor on the **ComputeNode** variant.

4. Enable the appropriate repository by running the following command as **root**:

```
yum-config-manager --enable repository
```

Once the subscription is attached to the system, you can install Red Hat Software Collections as described in Section 2.2, “Installing Red Hat Software Collections”. For more information on how to register your system using Red Hat Subscription Management and associate it with subscriptions, see Using and Configuring Red Hat Subscription Manager.

2.1.2. Using RHN Classic

If your system is registered with RHN Classic (applicable to Red Hat Enterprise Linux versions prior to 6.4), complete the following steps to subscribe to Red Hat Software Collections:

1. Display a list of all channels that are available to you and determine the exact name of the Red Hat Software Collections channel. To do so, type the following at a shell prompt as **root**:

```
rhn-channel --available-channels
```

The name of the channel depends on the specific version of Red Hat Enterprise Linux you are using and is in the following format, where *variant* is the Red Hat Enterprise Linux system variant (**server** or **workstation**):

```
rhel-x86_64-variant-6-rhscl-1

rhel-x86_64-server-6.4.z-rhscl-1
rhel-x86_64-server-6.5.z-rhscl-1
rhel-x86_64-server-6.6.z-rhscl-1

rhel-x86_64-variant-7-rhscl-1
```

Note that Red Hat Enterprise Linux 7 channels are accessible only through Red Hat Satellite instances.

2. Subscribe the system to the Red Hat Software Collections channel by running the following command as **root**:

```
rhn-channel --add --channel=channel_name
```

   Replace *channel_name* with the name you determined in the previous step.

3. Verify the list of channels you are subscribed to. As **root**, type:

```
rhn-channel --list
```

Once the system is subscribed, you can install Red Hat Software Collections as described in Section 2.2, “Installing Red Hat Software Collections”. For more information on how to register your system with RHN Classic, see Using and Configuring Red Hat Subscription Manager.

2.2. Installing Red Hat Software Collections

Red Hat Software Collections is distributed as a collection of RPM packages that can be installed, updated, and uninstalled by using the standard package management tools included in Red Hat Enterprise Linux. Note that a valid subscription is required to install Red Hat Software Collections on your system. For detailed instructions on how to associate your system with an appropriate subscription and get access to Red Hat Software Collections, see Section 2.1, “Getting Access to Red Hat Software Collections”.

Important

Some of the Red Hat Software Collections 1.2 packages require the **Optional** channel to be enabled in order to complete the full installation of these packages:

- The *php54-php-imap* and *php55-php-imap* packages require the *libc-client* package from the Optional channel.
- The *php54-php-recode* and *php55-php-recode* packages require the *recode* package from the Optional channel.
- The *perl516-perl-devel* package requires the *gdbm-devel* package from the Optional channel.
- The *mariadb55-mariadb-bench* package requires the *perl-GD* package from the Optional channel.

In Red Hat Enterprise Linux 7, the following packages available only in the **Optional** channel are required:

- The **Node.js** Software Collection depends on the *nodejs010-nodejs-devel* package, which requires the *c-ares-devel* package from the Optional channel (applicable to minimal install).
- The *httpd24-mod_ldap* package requires the *apr-util-ldap* package from the Optional channel.
- The *php54-php-pspell* and *php55-php-pspell* packages require the *aspell* package from the Optional channel.
- The *python27-python-debug* package requires the *tix* package from the Optional channel.
- The *thermostat1-thermostat* package requires the *apache-commons-beanutils*, *jansi*, *hawtjni*, *jansi-native*, and *objectweb-asm* packages from the Optional channel (applicable to minimal install).
- The *thermostat1-thermostat-webapp* package requires the *felix-framework* package from the Optional channel (applicable to minimal install).
- The *thermostat1-netty* package requires the *jzlib* package from the Optional channel (applicable to minimal install).
- The *apache-commons-logging* package requires the *xerces-j2* package from the Optional channel (applicable to minimal install).

For detailed instructions on how to subscribe your system to this channel, see the relevant Knowledgebase article on the Customer Portal.

Important

Use of Red Hat Software Collections 1.2 requires the removal of the pre-release versions. It is not possible to update to any Red Hat Software Collections 1.2 component from the Beta version. If you have previously installed any Red Hat Software Collections 1.2 component from the Beta version of Red Hat Software Collections, uninstall it from your system and install the new version as described in the Section 2.3, “Uninstalling Red Hat Software Collections” and Section 2.2.1, “Installing Individual Software Collections” sections.

Important

The in-place upgrade from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7 is not supported by Red Hat Software Collections. As a consequence, the installed Software Collections might not work correctly after the upgrade. If you want to upgrade from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7, it is strongly recommended to remove all Red Hat Software Collections packages, perform the in-place upgrade, update the Red Hat Software Collections repository, and install the Software Collections packages again. It is advisable to back up all data before upgrading.

2.2.1. Installing Individual Software Collections

To install any of the Software Collections that are listed in Table 1.1, "Red Hat Software Collections 1.2 Components", install the corresponding meta package by typing the following at a shell prompt as **root**:

```
yum install software_collection...
```

Replace *software_collection* with a space-separated list of Software Collections you want to install. For example, to install *php54* and *mariadb55*, type as **root**:

```
~]# yum install php54 mariadb55
```

This installs the main meta package for the selected Software Collection and a set of required packages as its dependencies. For information on how to install additional packages such as additional modules, see Section 2.2.2, "Installing Optional Packages".

2.2.2. Installing Optional Packages

Each component of Red Hat Software Collections is distributed with a number of optional packages that are not installed by default. To list all packages that are part of a certain Software Collection but are not installed on your system, type the following at a shell prompt:

```
yum list available software_collection-\*
```

To install any of these optional packages, type as **root**:

```
yum install package_name...
```

Replace *package_name* with a space-separated list of packages that you want to install. For example, to install the *perl516-perl-CPAN* and *perl516-perl-Archive-Tar*, type:

```
~]# yum install perl516-perl-CPAN perl516-perl-Archive-Tar
```

2.2.3. Installing Debugging Information

To install debugging information for any of the Red Hat Software Collections packages, make sure that the *yum-utils* package is installed and type the following command as **root**:

```
debuginfo-install package_name
```

For example, to install debugging information for the *ruby193-ruby* package, type:

```
~]# debuginfo-install ruby193-ruby
```

Note that in order to use this command, you need to have access to the repository with these packages. If your system is registered with Red Hat Subscription Management, enable the **rhel-*variant*-rhscl-6-debug-rpms** or **rhel-*variant*-rhscl-7-debug-rpms** repository as described in Section 2.1.1, "Using Red Hat Subscription Management". If your system is registered with RHN Classic, subscribe the system to the **rhel-x86_64-*variant*-6-rhscl-1-debuginfo** or **rhel-x86_64-*variant*-7-rhscl-1-debuginfo** channel as described in Section 2.1.2, "Using RHN Classic". For more information on how to get access to debuginfo packages, see https://access.redhat.com/solutions/9907.

2.3. Uninstalling Red Hat Software Collections

To uninstall any of the Software Collections components, type the following at a shell prompt as **root**:

```
yum remove software_collection\*
```

Replace *software_collection* with the Software Collection component you want to uninstall.

Note that uninstallation of the packages provided by Red Hat Software Collections does not affect the Red Hat Enterprise Linux system versions of these tools.

2.4. Rebuilding Red Hat Software Collections

<collection>-build packages are not provided by default. If you wish to rebuild a collection and do not want or cannot use the **rpmbuild --define 'scl foo'** command, you first need to rebuild the metapackage, which provides the *<collection>-build* package.

Note that existing collections should not be rebuilt with different content. To add new packages into an existing collection, you need to create a new collection containing the new packages and make it dependent on packages from the original collection. The original collection has to be used without changes.

For detailed information on building Software Collections, refer to the Red Hat Software Collections Packaging Guide.

Chapter 3. Usage

This chapter describes the necessary steps for rebuilding and using Red Hat Software Collections 1.2 and deploying applications that use Red Hat Software Collections.

3.1. Using Red Hat Software Collections

3.1.1. Running an Executable from a Software Collection

To run an executable from a particular Software Collection, type the following command at a shell prompt:

```
scl enable software_collection... 'command...'
```

Or, alternatively, use the following command:

```
scl enable software_collection... -- command...
```

Replace *software_collection* with a space-separated list of Software Collections you want to use and *command* with the command you want to run. For example, to execute a Perl program stored in a file named **hello.pl** with the Perl interpreter from the *perl516* Software Collection, type:

```
~]$ scl enable perl516 'perl hello.pl'
Hello, World!
```

You can execute any command using the **scl** utility, causing it to be run with the executables from a selected Software Collection in preference to their possible Red Hat Enterprise Linux system equivalents. For a complete list of Software Collections that are distributed with Red Hat Software Collections, see Table 1.1, “Red Hat Software Collections 1.2 Components”.

3.1.2. Running a Shell Session with a Software Collection as Default

To start a new shell session with executables from a selected Software Collection in preference to their Red Hat Enterprise Linux equivalents, type the following at a shell prompt:

```
scl enable software_collection... bash
```

Replace *software_collection* with a space-separated list of Software Collections you want to use. For example, to start a new shell session with the *python27* and *postgresql92* Software Collections as default, type:

```
~]$ scl enable python27 postgresql92 bash
```

The list of Software Collections that are enabled in the current session is stored in the **$X_SCLS** environment variable, for instance:

```
~]$ echo $X_SCLS
python27 postgresql92
```

For a complete list of Software Collections that are distributed with Red Hat Software Collections, see Table 1.1, “Red Hat Software Collections 1.2 Components”.

3.1.3. Running a System Service from a Software Collection

Software Collections that include system services install corresponding init scripts in the **/etc/rc.d/init.d/** directory. To start such a service in the current session, type the following at a shell prompt as **root**:

```
service software_collection-service_name start
```

Replace *software_collection* with the name of the Software Collection and *service_name* with the name of the service you want to start. To configure this service to start automatically at boot time, type the following command as **root**:

```
chkconfig software_collection-service_name on
```

For example, to start the **postgresql** service from the *postgresql92* Software Collection and enable it in runlevels 2, 3, 4, and 5, type as **root**:

```
~]# service postgresql92-postgresql start
Starting postgresql92-postgresql service:                  [  OK  ]
~]# chkconfig postgresql92-postgresql on
```

For more information on how to manage system services in Red Hat Enterprise Linux 6, refer to the Red Hat Enterprise Linux 6 Deployment Guide. For a complete list of Software Collections that are distributed with Red Hat Software Collections, see Table 1.1, “Red Hat Software Collections 1.2 Components”.

3.2. Accessing a Manual Page from a Software Collection

Every Software Collection contains a general manual page that describes the content of this component. Each manual page has the same name as the component and it is located in the **/opt/rh** directory.

To read a manual page for a Software Collection, type the following command:

```
scl enable software_collection 'man software_collection'
```

Replace *software_collection* with the particular Red Hat Software Collections component. For example, to display the manual page for *mariadb55*, type:

```
~]$ scl enable mariadb55 "man mariadb55"
```

3.3. Deploying Applications That Use Red Hat Software Collections

In general, you can use one of the following two approaches to deploy an application that depends on a component from Red Hat Software Collections in production:

- Install all required Software Collections and packages manually and then deploy your application, or
- Create a new Software Collection for your application and specify all required Software Collections and other packages as dependencies.

For more information on how to manually install individual Red Hat Software Collections

components, see Section 2.2, “Installing Red Hat Software Collections”. For further details on how to use Red Hat Software Collections, see Section 3.1, “Using Red Hat Software Collections”. For a detailed explanation of how to create a custom Software Collection or extend an existing one, read the Red Hat Software Collections Packaging Guide.

Chapter 4. Specifics of Individual Software Collections

This chapter is focused on the specifics of certain Software Collections and provides additional details concerning these components.

4.1. Red Hat Developer Toolset

Red Hat Developer Toolset is designed for developers working on the Red Hat Enterprise Linux platform. Red Hat Developer Toolset provides current versions of the **GNU Compiler Collection**, **GNU Debugger**, **Eclipse** development platform, and other development, debugging, and performance monitoring tools. Similarly to other Software Collections, an additional set of tools is installed into the **/opt/** directory. These tools are enabled by the user on demand using the supplied **scl** utility. Similarly to other Software Collections, these do not replace the Red Hat Enterprise Linux system versions of these tools, nor will they be used in preference to those system versions unless explicitly invoked using the **scl** utility.

For a list of features, refer to the Main Features section of the *Red Hat Developer Toolset Release Notes*.

For a complete list of components, see the Red Hat Developer Toolset Components table in the *Red Hat Developer Toolset User Guide*.

4.2. Thermostat 1

The **Thermostat** Software Collection provides a monitoring and instrumentation tool for the OpenJDK HotSpot JVM, with support for monitoring multiple JVM instances. The system is made up of two components: an **Agent**, which collects data, and a **Client**, which allows users to visualize collected data. These components communicate via a storage layer: either directly via **MongoDB** or indirectly via a Web layer for increased security. A pluggable agent and GUI framework allows for collection and visualization of performance data beyond what is included out of the box.

To install the *thermostat1* collection, type the following command as **root**:

```
yum install thermostat1
```

To enable this collection, type the following command at a shell prompt:

```
scl enable thermostat1 bash
```

To deploy the **thermostat1-thermostat-webapp**, start the web storage endpoint in Red Hat Software Collections by typing the following command as **root**:

```
service thermostat1-thermostat-tomcat start
```

For more information, please refer to the Thermostat User Guide. In order to deploy Thermostat securely, see the Configuration and Administration Guide.

4.3. Ruby on Rails 4.0

This Software Collection adds the *ruby200* package together with the *ror40* package. The **Ruby on Rails** collection can be enabled by the following command, which will automatically enable *ruby200*:

```
scl enable ror40 bash
```

These two collections are supported together.

4.4. MongoDB 2.4.9

To install the *mongodb24* collection, type the following command as **root**:

```
yum install mongodb24
```

To run the **MongoDB** shell utility, type the following command:

```
scl enable mongodb24 'mongo'
```

4.4.1. MongoDB 2.4.9 on Red Hat Enterprise Linux 6

If you are using Red Hat Enterprise Linux 6, the following instructions apply to your system.

To start the **MongoDB** daemon, type the following command as **root**:

```
service mongodb24-mongodb start
```

To start the **MongoDB** daemon on boot, type this command as **root**:

```
chkconfig mongodb24-mongodb on
```

To start the **MongoDB** sharding server, type this command as **root**:

```
service mongodb24-mongodb-shard start
```

To start the **MongoDB** sharding server on boot, type the following command as **root**:

```
chkconfig mongodb24-mongodb-shard on
```

4.4.2. MongoDB 2.4.9 on Red Hat Enterprise Linux 7

When using Red Hat Enterprise Linux 7, the following commands are applicable.

To start the **MongoDB** daemon, type the following command as **root**:

```
systemctl start mongodb24-mongodb.service
```

To start the **MongoDB** daemon on boot, type this command as **root**:

```
systemctl enable mongodb24-mongodb.service
```

To start the **MongoDB** sharding server, type the following command as **root**:

```
systemctl start mongodb24-mongodb-shard.service
```

To start the **MongoDB** sharding server on boot, type this command as **root**:

```
systemctl enable mongodb24-mongodb-shard.service
```

4.5. Git

Git is a distributed revision control system with a decentralized architecture. As opposed to centralized version control systems with a client-server model, Git ensures that each working copy of a Git repository is an exact copy with complete revision history. This not only allows you to work on and contribute to projects without the need to have permission to push your changes to their official repositories, but also makes it possible for you to work with no network connection. For detailed information, see the Git chapter in the *Red Hat Enterprise Linux 6 Developer Guide*.

4.6. DevAssistant

DevAssistant is a tool designed to assist developers with creating and setting up basic projects in various programming languages, installing dependencies, setting up a development environment, and working with source control. The *devassist09* Software Collection supports several programming languages, namely C, C++, Java, and Python. Additionally, DevAssistant is able to support working with any other language, framework, or tool due to its modular architecture.

DevAssistant is a framework that runs plug-ins called *assistants*. Each assistant can have several subassistants.

4.6.1. Getting Started with DevAssistant

To install the *devassist09* Software Collection, type the following command as **root**:

```
yum install devassist09
```

To enable this collection, type the following command at a shell prompt:

```
scl enable devassist09 bash
```

To get help for DevAssistant, use the following command:

```
devassistant --help
```

or the shorter variant of the same command:

```
da -h
```

It is advisable to use the **--help** option on each level to list your possible next steps, until you reach the level of an executable subassistant (see Example 4.1, “Creating a New Python Library Project”).

To access the graphical user interface, type this command at a shell prompt:

```
devassistant-gui
```

or the shortened variant:

```
da-gui
```

Please note that the GUI is available only if you install the *devassist09* Software Collection on Red Hat Enterprise Linux 7. The functionalities and procedures are the same as when using the command line interface.

Note that the **devassistant** and **da** commands are equal. Further in the text, we will use only the shorter variant, the **da** command.

4.6.2. Running Assistants

DevAssistant provides the following functionalities: **create**, **modify**, **prepare**, and **task**. To run an assistant, use the following command:

```
da [--debug] {create,modify,prepare,task} [assistant [arguments]] ...
```

The four basic commands and descriptions related to these functionalities are listed in the following table:

Table 4.1. Functionalities of DevAssistant

Command	Shortened Command	Description
da create	**da crt**	Creating a new project from scratch
da modify	**da mod**	Working with an existing project
da prepare	**da prep**	Preparing a development environment for an upstream project
da task		Performing a custom task not related to a specific project

The *devassist09* Software Collection does not include any assistants for the **modify**, **prepare**, and **task** functionalities. These categories are available for users who want to create their own assistants.

4.6.3. Creating Projects with DevAssistant

The *devassist09* Software Collection includes the following assistants for creating projects:

Table 4.2. Assistants for Creating Projects

Assistant	Subassistant	Description
c	app	An application in C
	lib	A dynamically linked library in C
cpp	app	An application in C++
	lib	A dynamically linked library in C++
java	maven	A simple project using Maven
python	lib	A simple library for Python

The following example demonstrates creating a new Python library project by following instructions displayed by the **--help** option.

> **Example 4.1. Creating a New Python Library Project**
>
> To create a new Python library project, complete the following steps:

1. Enable the *devassist09* Software Collection by running this command:

```
~]$ scl enable devassist09 bash
```

2. Display help about DevAssistant by using the **--help** option:

```
~]$ da --help
You can either run assistants with:
da [--debug] {create,modify,prepare,task} [ASSISTANT [ARGUMENTS]]
...

Where:
create   used for creating new projects
modify   used for working with existing projects
prepare  used for preparing environment for upstream projects
task     used for performing custom tasks not related to a
specific project
You can shorten "create" to "crt", "modify" to "mod" and
"prepare" to "prep".

Or you can run a custom action:
da [--debug] [ACTION] [ARGUMENTS]

Available actions:
help     Print detailed help
version  Print version
```

3. List the possible next steps for creating a project by typing:

```
~]$ da create --help
usage:  create [-h] [--deps-only] {c,cpp,java,python} ...

Kickstart new projects easily with DevAssistant.

optional arguments:
  -h, --help           show this help message and exit
  --deps-only          Only install dependencies

subassistants:
  Following subassistants will help you with setting up your
project.

  {c,cpp,java,python}
```

4. Display help on the **python** assistant by typing at a shell prompt:

```
~]$ da create python --help
usage: create python [-h] {lib} ...

This is a base Python assistant, you have to select a
subassistant.

optional arguments:
  -h, --help  show this help message and exit
```

```
subassistants:
  Following subassistants will help you with setting up your
project.

  {lib}
```

5. List your choices for the only **python** subassistant, **lib**, by running this command:

```
~]$ da create python lib --help
usage: create python lib [-h] [-e [ECLIPSE]] -n NAME

Scaffolds a simple Python library project.

optional arguments:
  -h, --help            show this help message and exit
  -e [ECLIPSE], --eclipse [ECLIPSE]
                        Configure as Eclipse project (uses
~/workspace or
                        specified directory)
  -n NAME, --name NAME  Name of project to create
```

6. Run the assistant to create your new Python library project named **mypythonlib** by using the following command:

```
~]$ da create python lib -n mypythonlib
```

To get more information about the upstream version of **DevAssistant**, refer to the DevAssistant User Documentation. Please note that though the basic concept of the upstream application is the same as in the *devassist09* Software Collection, individual plug-ins and their functionalities might differ.

4.7. Maven

The *maven30* Software Collection provides a software project management and comprehension tool. Based on the concept of a project object model (POM), **Maven** can manage a project's build, reporting, and documentation from a central piece of information.

To install the *maven30* collection, type the following command as **root**:

```
yum install maven30
```

To enable this collection, type the following command at a shell prompt:

```
scl enable maven30 bash
```

Global Maven settings, such as remote repositories or mirrors, can be customized by editing the **/opt/rh/maven30/root/etc/maven/settings.xml** file.

For more information about using Maven, refer to the Maven documentation. To find documentation regarding individual plug-ins, please see the index of plug-ins.

Chapter 5. Migration

This chapter provides information on migration from one version to another for specific components of Red Hat Software Collections 1.2.

5.1. Migrating from MySQL 5.1 to MySQL 5.5

5.1.1. Notable Differences Between MySQL 5.1 and MySQL 5.5

The following is a list of the most important changes between MySQL 5.1 and MySQL 5.5

- Starting with MySQL 5.5, the InnoDB storage engine (formerly known as InnoDB Plugin) is the default storage engine.
- InnoDB and some other plug-ins (for example, archive, blackhole and federated) were installable plug-ins in MySQL 5.1. Starting with MySQL 5.5, these plug-ins became compiled-in storage engines, that is, they cannot be installed or uninstalled by default.
- If you used InnoDB Plugin and it was loaded using the **`plugin-load=innodb=ha_innodb_plugin.so`** configuration option, you need to remove this configuration option as it does not work in MySQL 5.5.
- In MySQL 5.1, InnoDB Plugin included a configuration variable **`innodb_file_io_threads`**. However, this variable does not exist in MySQL 5.5; new variables, **`innodb_read_io_threads`** and **`innodb_write_io_threads`**, are used instead. To ensure proper functionality, either remove the former variable from the configuration file or replace it with the current variables.
- When upgrading from MySQL 5.1 to MySQL 5.5 using the in-place upgrading method, the **`mysql.proxies_priv`** table will not exist. To create the missing table, the **`mysql_upgrade`** utility has to be run as soon as the new daemon is started.
- MySQL 5.5 uses latin1 for the **`stopword`** file if the **`character_set_server`** variable is ucs2, utf16 or utf32. Thus, if the table uses FULLTEXT indexes in these cases, users should repair the table using the **`REPAIR TABLE`** ***`table_name`*** **`QUICK`**.
- MySQL 5.1 used the **`language`** variable for specifying the directory which included the error message file. This option is now deprecated and has been replaced by the **`lc_messages_dir`** and **`lc_messages`** options. This also applies for configuration options. Also, error messages no longer contain mixed set of character sets and error messages are returned in the set following the **`character_set_results`** system variable instead. That is, some error messages can be different in MySQL 5.5.

Please note that the EXAMPLE plug-in is no longer distributed in Red Hat Software Collections packages.

For more information about MySQL 5.1 and MySQL 5.5, refer to the release notes available at http://dev.mysql.com/doc/relnotes/mysql/5.1/en/ and http://dev.mysql.com/doc/relnotes/mysql/5.5/en/.

Important

MariaDB is a community-developed drop-in replacement for **MySQL**. The differences between **MySQL 5.1** and **MySQL 5.5** are valid also for **MySQL 5.1** and **MariaDB 5.5**.

5.1.2. Upgrading from MySQL 5.1 to MySQL 5.5

Before migrating from MySQL 5.1 to MySQL 5.5, back up all your data, including any MySQL databases. Because the *mysql55* Software Collection does not conflict with the *mysql* packages from the core systems, it is possible to install the *mysql55* Software Collection together with the *mysql* packages. It is also possible to run both versions at the same time, however, the port number and the socket in the my.cnf files need to be changed to prevent these specific resources from conflicting.

Upgrading can be performed either by using the **mysqldump** and **mysqlimport** utilities or using in-place upgrade:

- In the first scenario, the whole dump of all databases from one database is generated, **mysql** is run with the dump file as an input, using **mysqlimport** or the **LOAD DATA INFILE** SQL command within the other database. At the same time, the appropriate daemons have to be running during both dumping and restoring. You can use the **--all-databases** option in the mysqldump call to include all databases in the dump. The **--routines**, **--triggers** and **--events** options can also be used if needed.
- During the in-place upgrade, the data files are copied from one database directory to another database directory. The daemons should not be running at the time of copying. Set the appropriate permissions and SELinux context for copied files.

After upgrading, start the server and run the **mysql_upgrade** command. Running **mysql_upgrade** is necessary to check and repair internal tables.

Important

All scripts that work with a server form Software Collection, especially the **mysql_upgrade** script, should be run inside the **scl enable** environment.

In case the **root** user has a non-empty password defined (it should have it defined), it is necessary to call the **mysql_upgrade** utility with the **-p** option and specify the password.

The dump and restore upgrade method is recommended. The in-place upgrade method is usually faster, however, there are certain risks and known problems. For more information, refer to the MySQL 5.5 Release Notes.

In addition, once the upgrade is complete, consider changing the appropriate settings in the **my.cnf** file to reflect the environment.

Example 5.1. Dump and Restore Upgrade

```
~]# service mysqld start
Starting mysqld:                                           [  OK  ]
~]# mysqldump --all-databases --routines --events > dump.sql
~]# service mysqld stop
Stopping mysqld:                                           [  OK  ]
~]# service mysql55-mysqld start
Starting mysql55-mysqld:                                   [  OK  ]
~]# scl enable mysql55 'mysql' < dump.sql
~]# scl enable mysql55 'mysql_upgrade -u root -p'
Enter password:
Looking for 'mysql' as: mysql
Looking for 'mysqlcheck' as: mysqlcheck
```

```
Running 'mysqlcheck with default connection arguments
Running 'mysqlcheck with default connection arguments
a.t1                                               OK
mysql.columns_priv                                 OK
<skipped tables list>
mysql.user                                         OK
Running 'mysql_fix_privilege_tables'...
OK
```

Example 5.2. In-place Upgrade

```
~]# service mysqld stop
Stopping mysqld:                                           [  OK  ]
~]# service mysql55-mysqld stop
Stopping mysql55-mysqld:                                   [  OK  ]
~]# rm -rf /opt/rh/mysql55/root/var/lib/mysql/
~]# cp -r /var/lib/mysql/ /opt/rh/mysql55/root/var/lib/mysql/
~]# chown -R mysql:mysql /opt/rh/mysql55/root/var/lib/mysql/
~]# restorecon -R /opt/rh/mysql55/root/var/lib/mysql/
~]# service mysql55-mysqld start
Starting mysql55-mysqld:                                   [  OK  ]
~]# scl enable mysql55 'mysql_upgrade -u root -p'
Enter password:
Looking for 'mysql' as: mysql
Looking for 'mysqlcheck' as: mysqlcheck
Running 'mysqlcheck with default connection arguments
Running 'mysqlcheck with default connection arguments
a.t1                                               OK
mysql.columns_priv                                 OK
<skipped tables list>
mysql.user                                         OK
Running 'mysql_fix_privilege_tables'...
OK
```

For more information about the upgrading process, refer to MySQL 5.5 Reference Manual.

Important

MariaDB is a community-developed drop-in replacement for **MySQL**. The steps for upgrading from **MySQL 5.1** to **MySQL 5.5** are valid also for upgrading from **MySQL 5.1** to **MariaDB 5.5**, with the exception of the following differences:

- The *mariadb55* component name should be used instead of the *mysql55*, so replace all occurrences of **mysql55** with **mariadb55**.
- The **systemd** unit name for **MariaDB 5.5.** is **mariadb55-mariadb** in Red Hat Enterprise Linux 7, while the **SysV** unit script for **MariaDB 5.5** is called **mariadb55-mysqld** in Red Hat Enterprise Linux 6.

5.1.3. Using the mysql55-mysql-devel Package

Red Hat Software Collections contains the server part of MySQL 5.5 database. Red Hat Enterprise Linux 6 provides version 5.1 of this database (client library and server daemon). A protocol which is used between the client library and the daemon is stable across database versions, so using, for example, the MySQL 5.1 client library with the MySQL 5.5 daemon works as expected.

5.1.3.1. Using Database Connectors for Dynamic Languages

Important

When a MariaDB or MySQL database contains old users created using old authentication schema, PHP using the **mysqlnd** driver will not be able to connect to the database. This is because the **old_password** setting in the **/etc/my.cnf** file is turned off by default on Red Hat Enterprise Linux 6 while it is enabled on Red Hat Enterprise Linux 5. To work around this problem, set **old_password** to 0, restart the MariaDB or MySQL service and set a new password for each user.

5.1.3.2. Building Applications for MySQL 5.5 from Red Hat Software Collections

MySQL 5.5 from Red Hat Software Collections does not include database connectors; client libraries packaged in the MySQL 5.5 Red Hat Software Collections database packages are not supposed to be used as they are included only for purposes of server utilities and the daemon. Users are instead expected to use the system libraries and database connectors provided with the core system.

It means that users who would like to link their application against the MySQL client library should compile it and link it to the core Red Hat Enterprise Linux 6 environment, not to the MySQL 5.5 Red Hat Software Collections environment.

The only exception to this are server-side plug-ins, which are expected to be built under the MySQL 5.5 Red Hat Software Collections environment. This means the build process should be run inside the **scl enable mysql55 '...'** call.

5.2. Migrating from PostgreSQL 8.4 to PostgreSQL 9.2

Red Hat Software Collections 1.2 is distributed with PostgreSQL 9.2, which can be safely installed on the same machine in parallel with PostgreSQL 8.4 from Red Hat Enterprise Linux 6. It is also possible to run both versions of PostgreSQL on one machine at the same time, but you need to use different ports or IP addresses and adjust SELinux policy.

5.2.1. Notable Differences Between PostgreSQL 8.4 and PostgreSQL 9.2

The following is a list of the most important changes between PostgreSQL 8.4 and PostgreSQL 9.2:

- The following server configuration parameters have been removed and are no longer supported: **add_missing_from**, **regex_flavor**, **silent_mode**, **wal_sender_delay**, and **custom_variable_classes**. Do not use any of these parameters in the new configuration file.
- The **unix_socket_directory** parameter has been renamed to **unix_socket_directories** and can now be used to specify more than one UNIX socket to listen on. To do so, provide a list of comma-separated directories as the value of this option. The default value remains unchanged and is **/tmp**.

- New configuration parameters **ssl_ca_file**, **ssl_cert_file**, **ssl_crl_file**, and **ssl_key_file** have been added. These configuration parameters can be used to specify the locations of server-side SSL files that were previously hard-coded as relative paths to the **root.crt**, **server.crt**, **root.crl**, and **server.key** files in the data directory.

  Note that the PostgreSQL server no longer reads the **root.crt** and **root.crl** files by default. To load these files, change the corresponding parameters to non-default values.

- The **=>** operator has been removed and users are now advised to use the **hstore(*text*, *text*)** function.

- The default value of the **standard_conforming_strings** configuration parameter is now **on**. This configuration parameter controls if ordinary string literals (strings enclosed in single quotes) treat backslashes literally as specified in the SQL standard.

- A new configuration parameter, **backslash_quote**, has been added. This configuration parameter can be used to control whether a single quotation mark can be represented by **\'** in string literals. The default value is **safe_encoding**, which permits the use of **\'** only when the client encoding does not allow ASCII backslashes in multi-byte characters. As a consequence, **\'** can now be interpreted differently only in specific cases and only in string literals that do not conform to standards, including escape string syntax, **E'*value*'**.

- PostgreSQL 9.0 introduced access privileges for large objects. Consequently, a new configuration parameter, **lo_compat_privileges**, has been added to allow you to disable security checks related to the large objects affected by this change. To disable these security checks, change the value of this configuration parameter to **on**. The default value is **off**.

For a detailed list of known compatibility issues with earlier versions, see the official notes for PostgreSQL 9.0, PostgreSQL 9.1, and PostgreSQL 9.2. For an in-depth list of changes in behavior, see the upstream Release Notes.

5.2.2. Upgrading from PostgreSQL 8.4 to PostgreSQL 9.2

To migrate your data from PostgreSQL 8.4 that is distributed with Red Hat Enterprise Linux 6 to PostgreSQL 9.2 that is included in Red Hat Software Collections 1.2, you can either perform an in-place upgrade (recommended), or dump the database data into a text file with SQL commands and import it in the new database. Note that the second method is usually significantly slower and may require manual fixes; see the official documentation for more information about this upgrade method. If you need to migrate PosgreSQL databases to Red Hat Enterprise Linux 7, see https://access.redhat.com/articles/541873 and https://access.redhat.com/articles/694413.

Important

Before migrating your data from PostgreSQL 8.4 to PostgreSQL 9.2, make sure that you back up all your data, including the PostgreSQL database files that are by default located in the **/var/lib/pgsql/data/** directory.

Procedure 5.1. Performing In-place Upgrade

To perform an in-place upgrade of your PostgreSQL server, complete the following steps:

1. Stop the old PostgreSQL server to ensure that the data is not in an inconsistent state. To do so, type the following at a shell prompt as **root**:

```
service postgresql stop
```

To verify that the server is not running, type:

```
service postgresql status
```

2. Verify that the new data directory located in **/opt/rh/postgresql92/root/var/lib/pgsql/data/** does not exist:

```
file /opt/rh/postgresql92/root/var/lib/pgsql/data/
```

If you are running a fresh installation of PostgreSQL 9.2, this directory should not be present in your system. If it is, back it up by running the following command as **root**:

```
mv /opt/rh/postgresql92/root/var/lib/pgsql/data{,-scl-backup}
```

3. Copy the old database data to the new location by typing the following at a shell prompt as **root**:

```
cp -ra /var/lib/pgsql/data/
/opt/rh/postgresql92/root/var/lib/pgsql/
```

4. Open the **/opt/rh/postgresql92/root/var/lib/pgsql/data/pg_hba.conf** configuration file and verify that the **postgres** user is allowed to connect to the PostgreSQL server from **localhost** without a password. If not, you can edit this file and temporarily set the authentication method for the **postgres** user to **trust** or **ident**. For a detailed description of the **pg_hba.conf** file and a complete list of available configuration options, see the official documentation.

5. Upgrade the database data for the new server by running the following command as **root**:

```
service postgresql92-postgresql upgrade
```

It is recommended that you read the resulting **/opt/rh/postgresql92/root/var/lib/pgsql/pgupgrade.log** log file to see if there were any problems with the upgrade.

6. Start the new server as **root**:

```
service postgresql92-postgresql start
```

It is also advised that you run the **analyze_new_cluster.sh** script as follows:

```
su - postgres -c 'scl enable postgresql92
~/analyze_new_cluster.sh'
```

7. Optionally, you can configure the PostgreSQL 9.2 server to start automatically at boot time. To disable the old PostgreSQL 8.4 server, type the following command as **root**:

```
chkconfig postgresql off
```

To enable the PostgreSQL 9.2 server, type as **root**:

```
chkconfig postgresql92-postgresql on
```

Procedure 5.2. Performing a Dump and Restore Upgrade

To perform a dump and restore upgrade of your PostgreSQL server, complete the following steps:

1. Ensure that the old PostgreSQL server is running by typing the following at a shell prompt as **root**:

```
service postgresql start
```

2. Dump all data in the PostgreSQL database into a script file. As **root**, type:

```
su - postgres -c 'pg_dumpall > ~/pgdump_file.sql'
```

3. Stop the old server by running the following command as **root**:

```
service postgresql stop
```

4. Initialize the data directory for the new server as **root**:

```
service postgresql92-postgresql initdb
```

5. Start the new server as **root**:

```
service postgresql92-postgresql start
```

6. Import data from the previously created SQL file:

```
su - postgres -c 'scl enable postgresql92 "psql -f
~/pgdump_file.sql postgres"'
```

7. Optionally, you can configure the PostgreSQL 9.2 server to start automatically at boot time. To disable the old PostgreSQL 8.4 server, type the following command as **root**:

```
chkconfig postgresql off
```

   To enable the PostgreSQL 9.2 server, type as **root**:

```
chkconfig postgresql92-postgresql on
```

8. If your configuration differs from the default one, make sure to update configuration files, especially the **/opt/rh/postgresql92/root/var/lib/pgsql/data/pg_hba.conf** configuration file. Otherwise only the **postgres** user will be allowed to access the database.

5.3. Migrating from nginx 1.4 to nginx 1.6

In Red Hat Software Collections 1.2, **nginx** has been upgraded from version 1.4.4 to 1.6.1. The Software Collection has been renamed to *nginx16* and it is now supported.

The *nginx16* Software Collection uses a new prefix in accordance with the name of the collection and a different path to the root directory, which is now located in **/opt/rh/nginx16/root/**. The error log is now stored in **/var/log/nginx16/error.log** by default, and the initscript is called **nginx16-nginx**.

Configuration files in nginx 1.6 have the same format as in the previous version and they are compatible between version 1.4 and 1.6.

Important

Before upgrading from nginx 1.4 to nginx 1.6, back up all your data, including web pages and configuration files located in the **/opt/rh/nginx14/root/** tree.

If you have made any specific changes, such as changing configuration files or setting up web applications, in the **/opt/rh/nginx14/root/** tree, replicate those changes in the new **/opt/rh/nginx16/root/** directory, too.

For the official **nginx** documentation, please refer to http://nginx.org/en/docs/.

Chapter 6. Additional Resources

For more information about Red Hat Software Collections 1.2 and Red Hat Enterprise Linux, refer to the resources listed below.

6.1. Red Hat Enterprise Linux Developer Program Group

Users of Red Hat Software Collections can access the Red Hat Enterprise Linux Developer Program Group in the Red Hat Customer Portal to get developer related information for the development tools available for Red Hat Enterprise Linux. In addition, users can find developer related papers and videos on topics that are of interest to developers, for example RPM building, threaded programming, performance tuning, debugging, and so on.

To visit the Red Hat Enterprise Linux Developer Program Group, log in to the Customer Portal, click **Products** at the top of the page, choose **Services**, and then **Red Hat Enterprise Linux Developer Program** from the list.

6.2. Red Hat Product Documentation

The following documents are directly or indirectly relevant to this book:

- Red Hat Software Collections 1.2 Packaging Guide — The *Packaging Guide* for Red Hat Software Collections explains the concept of Software Collections, documents the **scl** utility, and provides a detailed explanation of how to create a custom Software Collection or extend an existing one.
- Red Hat Developer Toolset 3.0 Release Notes — The *Release Notes* for Red Hat Developer Toolset document known problems, possible issues, changes, and other important information about this Software Collection.
- Red Hat Developer Toolset 3.0 User Guide — The *User Guide* for Red Hat Developer Toolset contains more information about installing and using this Software Collection.
- Using and Configuring Red Hat Subscription Manager — The *Using and Configuring Red Hat Subscription Manager* book provides detailed information on how to register Red Hat Enterprise Linux systems, manage subscriptions, and view notifications for the registered systems.
- Red Hat Enterprise Linux 6 Developer Guide — The *Developer Guide* for Red Hat Enterprise Linux 6 provides more information for developers on the Red Hat Enterprise Linux platform.
- Red Hat Enterprise Linux 7 Developer Guide — The *Developer Guide* for Red Hat Enterprise Linux 7 provides an introduction to application development tools in Red Hat Enterprise Linux 7.
- Red Hat Enterprise Linux 6 Deployment Guide — The *Deployment Guide* for Red Hat Enterprise Linux 6 provides relevant information regarding the deployment, configuration, and administration of this system.
- Red Hat Enterprise Linux 7 System Administrator's Guide — The *System Administrator's Guide* for Red Hat Enterprise Linux 7 provides information on deployment, configuration, and administration of this system.

6.3. Red Hat Developer Blog

Red Hat Developer Blog content is directed to designers and developers of applications based on

Red Hat technologies. It contains links to product team blogs and other relevant internal and external resources. Its goal is to inform and engage the developer community with up-to-date information, best practices, opinion, product and program announcements as well as pointers to sample code and other resources.

Revision History

Revision 1.1-22 **Thu Oct 30 2014** **Lenka Špačková**

Release of Red Hat Software Collections 1.2 Release Notes.

Revision 1.1-20 **Tue Oct 07 2014** **Lenka Špačková**

Release of Red Hat Software Collections 1.2 Beta-2 Release Notes.

Revision 1.1-19 **Tue Sep 09 2014** **Lenka Špačková**

Release of Red Hat Software Collections 1.2 Beta-1 Release Notes.