NAME

bundle−config − Set bundler configuration options

SYNOPSIS

bundle config [list|get|set|unset] [*name* [*value*]]

DESCRIPTION

This command allows you to interact with Bundler´s configuration system.

Bundler loads configuration settings in this order:

1. Local config (**app/.bundle/config**)
2. Environmental variables (**ENV**)
3. Global config (**˜/.bundle/config**)
4. Bundler default config

Executing **bundle config list** with will print a list of all bundler configuration for the current bundle, and where that configuration was set.

Executing **bundle config get <name>** will print the value of that configuration setting, and where it was set.

Executing **bundle config set <name> <value>** will set that configuration to the value specified for all bundles executed as the current user. The configuration will be stored in **˜/.bundle/config**. If *name* already is set, *name* will be overridden and user will be warned.

Executing **bundle config set −−global <name> <value>** works the same as above.

Executing **bundle config set −−local <name> <value>** will set that configuration to the local application. The configuration will be stored in **app/.bundle/config**.

Executing **bundle config unset <name>** will delete the configuration in both local and global sources.

Executing **bundle config unset −−global <name>** will delete the configuration only from the user configuration.

Executing **bundle config unset −−local <name> <value>** will delete the configuration only from the local application.

Executing bundle with the **BUNDLE_IGNORE_CONFIG** environment variable set will cause it to ignore all configuration.

Executing **bundle config set disable_multisource true** upgrades the warning about the Gemfile containing multiple primary sources to an error. Executing **bundle config unset disable_multisource** downgrades this error to a warning.

REMEMBERING OPTIONS

Flags passed to **bundle install** or the Bundler runtime, such as **−−path foo** or **−−without production**, are remembered between commands and saved to your local application´s configuration (normally, **./.bundle/config**).

However, this will be changed in bundler 3, so it´s better not to rely on this behavior. If these options must be remembered, it´s better to set them using **bundle config** (e.g., **bundle config set path foo**).

The options that can be configured are:

bin Creates a directory (defaults to **˜/bin**) and place any executables from the gem there. These executables run in Bundler´s context. If used, you might add this directory to your environment´s **PATH** variable. For instance, if the **rails** gem comes with a **rails** executable, this flag will create a **bin/rails** executable that ensures that all referred dependencies will be resolved using the bundled gems.

deployment
In deployment mode, Bundler will ´roll−out´ the bundle for **production** use. Please check carefully if you want to have this option enabled in **development** or **test** environments.

path The location to install the specified gems to. This defaults to Rubygems´ setting. Bundler shares this location with Rubygems, **gem install ...** will have gem installed there, too. Therefore, gems installed without a **−−path ...** setting will show up by calling **gem list**. Accordingly, gems installed to other locations will not get listed.

without
A space−separated list of groups referencing gems to skip during installation.

with A space−separated list of groups referencing gems to include during installation.

BUILD OPTIONS

You can use **bundle config** to give Bundler the flags to pass to the gem installer every time bundler tries to install a particular gem.

A very common example, the **mysql** gem, requires Snow Leopard users to pass configuration flags to **gem install** to specify where to find the **mysql_config** executable.

```
gem install mysql −− −−with−mysql−config=/usr/local/mysql/bin/mysql_config
```

Since the specific location of that executable can change from machine to machine, you can specify these flags on a per−machine basis.

```
bundle config set build.mysql −−with−mysql−config=/usr/local/mysql/bin/mysql_config
```

After running this command, every time bundler needs to install the **mysql** gem, it will pass along the flags you specified.

CONFIGURATION KEYS

Configuration keys in bundler have two forms: the canonical form and the environment variable form.

For instance, passing the **−−without** flag to bundle install(1) *bundle−install.1.html* prevents Bundler from installing certain groups specified in the Gemfile(5). Bundler persists this value in **app/.bundle/config** so that calls to **Bundler.setup** do not try to find gems from the **Gemfile** that you didn´t install. Additionally, subsequent calls to bundle install(1) *bundle−install.1.html* remember this setting and skip those groups.

The canonical form of this configuration is **"without"**. To convert the canonical form to the environment variable form, capitalize it, and prepend **BUNDLE_**. The environment variable form of **"without"** is **BUNDLE_WITHOUT**.

Any periods in the configuration keys must be replaced with two underscores when setting it via environment variables. The configuration key **local.rack** becomes the environment variable **BUNDLE_LOCAL__RACK**.

LIST OF AVAILABLE KEYS

The following is a list of all configuration keys and their purpose. You can learn more about their operation in bundle install(1) *bundle−install.1.html*.

- **allow_bundler_dependency_conflicts** (**BUNDLE_ALLOW_BUNDLER_DEPENDENCY_CONFLICTS**): Allow resolving to specifications that have dependencies on **bundler** that are incompatible with the running Bundler version.
- **allow_deployment_source_credential_changes** (**BUNDLE_ALLOW_DEPLOYMENT_SOURCE_CREDENTIAL_CHANGES**): When in deployment mode, allow changing the

credentials to a gem´s source. Ex: **https://some.host.com/gems/path/ −> https://user_name:password@some.host.com/gems/path**

- **allow_offline_install** (**BUNDLE_ALLOW_OFFLINE_INSTALL**): Allow Bundler to use cached data when installing without network access.
- **auto_clean_without_path** (**BUNDLE_AUTO_CLEAN_WITHOUT_PATH**): Automatically run **bundle clean** after installing when an explicit **path** has not been set and Bundler is not installing into the system gems.
- **auto_install** (**BUNDLE_AUTO_INSTALL**): Automatically run **bundle install** when gems are missing.
- **bin** (**BUNDLE_BIN**): Install executables from gems in the bundle to the specified directory. Defaults to **false**.
- **cache_all** (**BUNDLE_CACHE_ALL**): Cache all gems, including path and git gems.
- **cache_all_platforms** (**BUNDLE_CACHE_ALL_PLATFORMS**): Cache gems for all platforms.
- **cache_path** (**BUNDLE_CACHE_PATH**): The directory that bundler will place cached gems in when running **bundle package**, and that bundler will look in when installing gems. Defaults to **vendor/cache**.
- **clean** (**BUNDLE_CLEAN**): Whether Bundler should run **bundle clean** automatically after **bundle install**.
- **console** (**BUNDLE_CONSOLE**): The console that **bundle console** starts. Defaults to **irb**.
- **default_install_uses_path** (**BUNDLE_DEFAULT_INSTALL_USES_PATH**): Whether a **bundle install** without an explicit **−−path** argument defaults to installing gems in **.bundle**.
- **deployment** (**BUNDLE_DEPLOYMENT**): Disallow changes to the **Gemfile**. When the **Gemfile** is changed and the lockfile has not been updated, running Bundler commands will be blocked.
- **disable_checksum_validation** (**BUNDLE_DISABLE_CHECKSUM_VALIDATION**): Allow installing gems even if they do not match the checksum provided by RubyGems.
- **disable_exec_load** (**BUNDLE_DISABLE_EXEC_LOAD**): Stop Bundler from using **load** to launch an executable in−process in **bundle exec**.
- **disable_local_branch_check** (**BUNDLE_DISABLE_LOCAL_BRANCH_CHECK**): Allow Bundler to use a local git override without a branch specified in the Gemfile.
- **disable_multisource** (**BUNDLE_DISABLE_MULTISOURCE**): When set, Gemfiles containing multiple sources will produce errors instead of warnings. Use **bundle config unset disable_multisource** to unset.
- **disable_platform_warnings** (**BUNDLE_DISABLE_PLATFORM_WARNINGS**): Disable warnings during bundle install when a dependency is unused on the current platform.
- **disable_shared_gems** (**BUNDLE_DISABLE_SHARED_GEMS**): Stop Bundler from accessing gems installed to RubyGems´ normal location.
- **disable_version_check** (**BUNDLE_DISABLE_VERSION_CHECK**): Stop Bundler from checking if a newer Bundler version is available on rubygems.org.
- **force_ruby_platform** (**BUNDLE_FORCE_RUBY_PLATFORM**): Ignore the current machine´s platform and install only **ruby** platform gems. As a result, gems with native extensions will be compiled from source.
- **frozen** (**BUNDLE_FROZEN**): Disallow changes to the **Gemfile**. When the **Gemfile** is changed and the lockfile has not been updated, running Bundler commands will be blocked. Defaults to **true** when **−−deployment** is used.
- **gem.push_key** (**BUNDLE_GEM__PUSH_KEY**): Sets the **−−key** parameter for **gem push** when using the **rake release** command with a private gemstash server.

- **gemfile** (**BUNDLE_GEMFILE**): The name of the file that bundler should use as the **Gemfile**. This location of this file also sets the root of the project, which is used to resolve relative paths in the **Gemfile**, among other things. By default, bundler will search up from the current working directory until it finds a **Gemfile**.
- **global_gem_cache** (**BUNDLE_GLOBAL_GEM_CACHE**): Whether Bundler should cache all gems globally, rather than locally to the installing Ruby installation.
- **ignore_messages** (**BUNDLE_IGNORE_MESSAGES**): When set, no post install messages will be printed. To silence a single gem, use dot notation like **ignore_messages.httparty true**.
- **init_gems_rb** (**BUNDLE_INIT_GEMS_RB**) Generate a **gems.rb** instead of a **Gemfile** when running **bundle init**.
- **jobs** (**BUNDLE_JOBS**): The number of gems Bundler can install in parallel. Defaults to 1.
- **no_install** (**BUNDLE_NO_INSTALL**): Whether **bundle package** should skip installing gems.
- **no_prune** (**BUNDLE_NO_PRUNE**): Whether Bundler should leave outdated gems unpruned when caching.
- **only_update_to_newer_versions** (**BUNDLE_ONLY_UPDATE_TO_NEWER_VERSIONS**): During **bundle update**, only resolve to newer versions of the gems in the lockfile.
- **path** (**BUNDLE_PATH**): The location on disk where all gems in your bundle will be located regardless of **$GEM_HOME** or **$GEM_PATH** values. Bundle gems not found in this location will be installed by **bundle install**. Defaults to **Gem.dir**. When −−deployment is used, defaults to vendor/bundle.
- **path.system** (**BUNDLE_PATH__SYSTEM**): Whether Bundler will install gems into the default system path (**Gem.dir**).
- **path_relative_to_cwd** (**BUNDLE_PATH_RELATIVE_TO_CWD**) Makes **−−path** relative to the CWD instead of the **Gemfile**.
- **plugins** (**BUNDLE_PLUGINS**): Enable Bundler´s experimental plugin system.
- **prefer_patch** (BUNDLE_PREFER_PATCH): Prefer updating only to next patch version during updates. Makes **bundle update** calls equivalent to **bundler update −−patch**.
- **print_only_version_number** (**BUNDLE_PRINT_ONLY_VERSION_NUMBER**) Print only version number from **bundler −−version**.
- **redirect** (**BUNDLE_REDIRECT**): The number of redirects allowed for network requests. Defaults to **5**.
- **retry** (**BUNDLE_RETRY**): The number of times to retry failed network requests. Defaults to **3**.
- **setup_makes_kernel_gem_public** (**BUNDLE_SETUP_MAKES_KERNEL_GEM_PUBLIC**): Have **Bundler.setup** make the **Kernel#gem** method public, even though RubyGems declares it as private.
- **shebang** (**BUNDLE_SHEBANG**): The program name that should be invoked for generated binstubs. Defaults to the ruby install name used to generate the binstub.
- **silence_deprecations** (**BUNDLE_SILENCE_DEPRECATIONS**): Whether Bundler should silence deprecation warnings for behavior that will be changed in the next major version.
- **silence_root_warning** (**BUNDLE_SILENCE_ROOT_WARNING**): Silence the warning Bundler prints when installing gems as root.
- **skip_default_git_sources** (**BUNDLE_SKIP_DEFAULT_GIT_SOURCES**): Whether Bundler should skip adding default git source shortcuts to the Gemfile DSL.
- **specific_platform** (**BUNDLE_SPECIFIC_PLATFORM**): Allow bundler to resolve for the specific running platform and store it in the lockfile, instead of only using a generic platform. A specific platform is the exact platform triple reported by **Gem::Platform.local**, such as **x86_64−darwin−16** or

universal−java−1.8. On the other hand, generic platforms are those such as **ruby**, **mswin**, or **java**. In this example, **x86_64−darwin−16** would map to **ruby** and **universal−java−1.8** to **java**.

- **ssl_ca_cert** (**BUNDLE_SSL_CA_CERT**): Path to a designated CA certificate file or folder containing multiple certificates for trusted CAs in PEM format.
- **ssl_client_cert** (**BUNDLE_SSL_CLIENT_CERT**): Path to a designated file containing a X.509 client certificate and key in PEM format.
- **ssl_verify_mode** (**BUNDLE_SSL_VERIFY_MODE**): The SSL verification mode Bundler uses when making HTTPS requests. Defaults to verify peer.
- **suppress_install_using_messages** (**BUNDLE_SUPPRESS_INSTALL_USING_MESSAGES**): Avoid printing **Using ...** messages during installation when the version of a gem has not changed.
- **system_bindir** (**BUNDLE_SYSTEM_BINDIR**): The location where RubyGems installs binstubs. Defaults to **Gem.bindir**.
- **timeout** (**BUNDLE_TIMEOUT**): The seconds allowed before timing out for network requests. Defaults to **10**.
- **unlock_source_unlocks_spec** (**BUNDLE_UNLOCK_SOURCE_UNLOCKS_SPEC**): Whether running **bundle update −−source NAME** unlocks a gem with the given name. Defaults to **true**.
- **update_requires_all_flag** (**BUNDLE_UPDATE_REQUIRES_ALL_FLAG**) Require passing **−−all** to **bundle update** when everything should be updated, and disallow passing no options to **bundle update**.
- **user_agent** (**BUNDLE_USER_AGENT**): The custom user agent fragment Bundler includes in API requests.
- **with** (**BUNDLE_WITH**): A **:**−separated list of groups whose gems bundler should install.
- **without** (**BUNDLE_WITHOUT**): A **:**−separated list of groups whose gems bundler should not install.

In general, you should set these settings per−application by using the applicable flag to the bundle install(1) *bundle−install.1.html* or bundle package(1) *bundle−package.1.html* command.

You can set them globally either via environment variables or **bundle config**, whichever is preferable for your setup. If you use both, environment variables will take preference over global settings.

LOCAL GIT REPOS

Bundler also allows you to work against a git repository locally instead of using the remote version. This can be achieved by setting up a local override:

```
bundle config set local.GEM_NAME /path/to/local/git/repository
```

For example, in order to use a local Rack repository, a developer could call:

```
bundle config set local.rack ~/Work/git/rack
```

Now instead of checking out the remote git repository, the local override will be used. Similar to a path source, every time the local git repository change, changes will be automatically picked up by Bundler. This means a commit in the local git repo will update the revision in the **Gemfile.lock** to the local git repo revision. This requires the same attention as git submodules. Before pushing to the remote, you need to ensure the local override was pushed, otherwise you may point to a commit that only exists in your local machine.

You´ll also need to CGI escape your usernames and passwords as well.

Bundler does many checks to ensure a developer won´t work with invalid references. Particularly, we force a developer to specify a branch in the **Gemfile** in order to use this feature. If the branch specified in the **Gemfile** and the current branch in the local git repository do not match, Bundler will abort. This ensures that a developer is always working against the correct branches, and prevents accidental locking to a different branch.

Finally, Bundler also ensures that the current revision in the **Gemfile.lock** exists in the local git repository. By doing this, Bundler forces you to fetch the latest changes in the remotes.

MIRRORS OF GEM SOURCES

Bundler supports overriding gem sources with mirrors. This allows you to configure rubygems.org as the gem source in your Gemfile while still using your mirror to fetch gems.

```
bundle config set mirror.SOURCE_URL MIRROR_URL
```

For example, to use a mirror of rubygems.org hosted at rubygems−mirror.org:

```
bundle config set mirror.http://rubygems.org http://rubygems−mirror.org
```

Each mirror also provides a fallback timeout setting. If the mirror does not respond within the fallback timeout, Bundler will try to use the original server instead of the mirror.

```
bundle config set mirror.SOURCE_URL.fallback_timeout TIMEOUT
```

For example, to fall back to rubygems.org after 3 seconds:

```
bundle config set mirror.https://rubygems.org.fallback_timeout 3
```

The default fallback timeout is 0.1 seconds, but the setting can currently only accept whole seconds (for example, 1, 15, or 30).

CREDENTIALS FOR GEM SOURCES

Bundler allows you to configure credentials for any gem source, which allows you to avoid putting secrets into your Gemfile.

```
bundle config set SOURCE_HOSTNAME USERNAME:PASSWORD
```

For example, to save the credentials of user **claudette** for the gem source at **gems.longerous.com**, you would run:

```
bundle config set gems.longerous.com claudette:s00pers3krit
```

Or you can set the credentials as an environment variable like this:

```
export BUNDLE_GEMS__LONGEROUS__COM="claudette:s00pers3krit"
```

For gems with a git source with HTTP(S) URL you can specify credentials like so:

```
bundle config set https://github.com/bundler/bundler.git username:password
```

Or you can set the credentials as an environment variable like so:

```
export BUNDLE_GITHUB__COM=username:password
```

This is especially useful for private repositories on hosts such as Github, where you can use personal OAuth tokens:

```
export BUNDLE_GITHUB__COM=abcd0123generatedtoken:x−oauth−basic
```

CONFIGURE BUNDLER DIRECTORIES

Bundler´s home, config, cache and plugin directories are able to be configured through environment variables. The default location for Bundler´s home directory is **~/.bundle**, which all directories inherit from by default. The following outlines the available environment variables and their default values

```
BUNDLE_USER_HOME : $HOME/.bundle
BUNDLE_USER_CACHE : $BUNDLE_USER_HOME/cache
BUNDLE_USER_CONFIG : $BUNDLE_USER_HOME/config
BUNDLE_USER_PLUGIN : $BUNDLE_USER_HOME/plugin
```