CPack variables

Some basic information about CPack variables can be found at CPackConfiguration. Generator specific settings can also be found on the CPackPackageGenerators.

Variables can be listed by:

```
$ cpack --help-variable-list
```

Help on an individual variables can be found by:

```
$ cpack --help-variable CPACK_SET_DESTDIR
```

This information is for CPack version 2.8.9.

1. Variables common to all CPack generators

CPACK_ABSOLUTE_DESTINATION_FILES

List of files which have been installed using an ABSOLUTE DESTINATION path.

This variable is a Read-Only variable which is set internally by CPack during installation and before packaging using:

```
CMAKE_ABSOLUTE_DESTINATION_FILES
```

defined in cmake_install.cmake scripts. The value can be used within CPack project configuration file and/or CPack<GEN>.cmake file of <GEN> generator.

CPACK_BINARY_<GENNAME>

CPack generated options for binary generators

The CPack.cmake module generates (when CPACK_GENERATOR is not set) a set of CMake options (see CMake option command) which may then be used to select the CPack generator(s) to be used when launching the package target.

CPACK_CMAKE_GENERATOR

What CMake generator should be used if the project is CMake project.

Defaults to the value of CMAKE_GENERATOR few users will want to change this setting.

CPACK_CREATE_DESKTOP_LINKS

List of desktop links to create

CPACK_GENERATOR

List of CPack generators to use

If not specified, CPack will create a set of options CPACK_BINARY_<GENNAME> (e.g., CPACK_BINARY_NSIS) allowing the user to enable/disable individual generators. This variable may be used on the command line as well as in:

```
cpack -D CPACK_GENERATOR="ZIP;TGZ" /path/to/build/tree
```

CPACK_INCLUDE_TOPLEVEL_DIRECTORY

Boolean toggle to include/exclude top level directory.

When preparing a package CPack installs the item under the so-called top level directory. The purpose of is to include (set to 1 or ON or TRUE) the top level directory in the package or not (set to 0 or OFF or FALSE).

Each CPack generator as a built-in default value for this variable. E.g. Archive generators (ZIP, TGZ, ...) includes the top level whereas RPM or DEB don't. The user may override the default value by setting this variable.

There is a similar variable:

```
CPACK_COMPONENT_INCLUDE_TOPLEVEL_DIRECTORY
```

which may be used to override the behavior for the component packaging case which may have different default value for historical (now backward compatibility) reason.

CPACK_INSTALLED_DIRECTORIES

Extra directories to install

CPACK_INSTALL_CMAKE_PROJECTS

List of four values that specify what project to install.

The four values are: Build directory, Project Name, Project Component, Directory. If omitted, CPack will build an installer that installers everything.

CPACK_INSTALL_COMMANDS

Extra commands to install components

CPACK_INSTALL_SCRIPT

Extra CMake script provided by the user.

If set this CMake script will be executed by CPack during its local [CPack-private] installation which is done right before packaging the files. The script is not called by e.g.: make install.

CPACK_MONOLITHIC_INSTALL

Disables the component-based installation mechanism.

When set the component specification is ignored and all installed items are put in a single "MONOLITHIC" package. Some CPack generators do monolithic packaging by default and may be asked to do component packaging by setting:

```
CPACK_<GENNAME>_COMPONENT_INSTALL
```

to 1/TRUE.

CPACK_OUTPUT_CONFIG_FILE

The name of the CPack binary configuration file.

This file is the CPack configuration generated by the CPack module for binary installers. Defaults to CPackConfig.cmake.

CPACK_PACKAGE_DESCRIPTION_FILE

A text file used to describe the project.

Used, for example, the introduction screen of a CPack-generated Windows installer to describe the project.

CPACK_PACKAGE_DESCRIPTION_SUMMARY

Short description of the project (only a few words).

CPACK_PACKAGE_DIRECTORY

The directory in which CPack is doing its packaging.

If it is not set then this will default (internally) to the build dir. This variable may be defined in CPack config file or from the cpack command line option "-B". If set the command line option override the value found in the config file.

CPACK_PACKAGE_EXECUTABLES

Lists each of the executables and associated text label to be used to create Start Menu shortcuts.

For example, setting this to the list `ccmake;CMake` will create a shortcut named "CMake" that will execute the installed executable ccmake. Not all CPack generators use it (at least NSIS and OSXX11 do).

CPACK_PACKAGE_FILE_NAME

The name of the package file to generate, not including the extension.

For example, `cmake-2.6.1-Linux-i686`. The default value is:

```
${CPACK_PACKAGE_NAME}-${CPACK_PACKAGE_VERSION}-${CPACK_SYSTEM_NAME}.
```

CPACK_PACKAGE_ICON

A branding image that will be displayed inside the installer (used by GUI installers).

CPACK_PACKAGE_INSTALL_DIRECTORY

Installation directory on the target system.

This may be used by some CPack generators like NSIS to create an installation directory e.g., "CMake 2.5" below the installation prefix. All installed element will be put inside this directory.

CPACK_PACKAGE_INSTALL_REGISTRY_KEY

Registry key used when installing this project.

This is only used by installer for Windows.

CPACK_PACKAGE_NAME

The name of the package (or application)

If not specified, defaults to the project name.

CPACK_PACKAGE_VENDOR

The name of the package vendor

(e.g., "Kitware").

CPACK_PACKAGE_VERSION

Package full version, used internally

By default, this is built from:

```
CPACK_PACKAGE_VERSION_MAJOR
CPACK_PACKAGE_VERSION_MINOR
CPACK_PACKAGE_VERSION_PATCH
```

CPACK_PACKAGE_VERSION_MAJOR

Package major Version

CPACK_PACKAGE_VERSION_MINOR

Package minor Version

CPACK_PACKAGE_VERSION_PATCH

Package patch Version

CPACK_PACKAGING_INSTALL_PREFIX

The prefix used in the built package.

Each CPack generator has a default value (like `/usr`). This default value may be overwritten from the `CMakeLists.txt` or the `cpack` command line by setting an alternative value. e.g.:

```
set(CPACK_PACKAGING_INSTALL_PREFIX "/opt")
```

This is not the same purpose as `CMAKE_INSTALL_PREFIX` which is used when installing from the build tree without building a package.

CPACK_PROJECT_CONFIG_FILE

CPack-time project CPack configuration file.

This file included at cpack time, once per generator after CPack has set `CPACK_GENERATOR` to the actual generator being used. It allows per-generator setting of `CPACK_*` variables at cpack time.

CPACK_RESOURCE_FILE_LICENSE

License to be embedded in the installer

It will typically be displayed to the user by the produced installer (often with an explicit "Accept" button, for graphical installers) prior to installation. This license file is NOT added to installed file but is used by some CPack generators like NSIS. If you want to install a license file (may be the same as this one) along with your project you must add an appropriate CMake `INSTALL` command in your `CMakeLists.txt`.

CPACK_RESOURCE_FILE_README

ReadMe file to be embedded in the installer

It typically describes in some detail the purpose of the project during the installation. Not all CPack generators uses this file.

CPACK_RESOURCE_FILE_WELCOME

Welcome file to be embedded in the installer.

It welcomes users to this installer. Typically used in the graphical installers on Windows and Mac OS X.

CPACK_SET_DESTDIR

Boolean toggle to make CPack use DESTDIR mechanism when packaging.

`DESTDIR` means DESTination DIRectory. It is commonly used by makefile users in order to install software at non-default location. It is a basic relocation mechanism. It is usually invoked like this:

```
make DESTDIR=/home/john install
```

which will install the concerned software using the installation prefix, e.g. "/usr/local" prepended with the `DESTDIR` value which finally gives "/home/john/usr/local". When preparing a package, CPack first installs the items to be packaged in a local (to the build tree) directory by using the same `DESTDIR` mechanism. Nevertheless, if `CPACK_SET_DESTDIR` is set then CPack will set `DESTDIR` before doing the local install. The most noticeable difference is that without `CPACK_SET_DESTDIR`, CPack uses `CPACK_PACKAGING_INSTALL_PREFIX` as a prefix whereas with `CPACK_SET_DESTDIR` set, CPack will use `CMAKE_INSTALL_PREFIX` as a prefix.

Manually setting `CPACK_SET_DESTDIR` may help (or simply be necessary) if some install rules uses absolute DESTINATION (see CMake `INSTALL` command). However, starting with CPack/CMake 2.8.3 RPM and DEB installers tries to handle `DESTDIR` automatically so that it is seldom necessary for the user to set it.

CPACK_SOURCE_GENERATOR

List of generators used for the source packages.

As with CPACK_GENERATOR, if this is not specified then CPack will create a set of options (e.g., CPACK_SOURCE_ZIP) allowing users to select which packages will be generated.

CPACK_SOURCE_IGNORE_FILES

Pattern of files in the source tree that won't be packaged when building a source package.

This is a list of regular expression patterns (that must be properly escaped), e.g.:

```
/CVS/;/\\.svn/;\\.swp$;\\.#;/#;.*~;cscope.*
```

CPACK_SOURCE_OUTPUT_CONFIG_FILE

The name of the CPack source configuration file.

This file is the CPack configuration generated by the CPack module for source installers. Defaults to CPackSourceConfig.cmake.

CPACK_SOURCE_PACKAGE_FILE_NAME

The name of the source package

For example cmake-2.6.1.

CPACK_SOURCE_STRIP_FILES

List of files in the source tree that will be stripped.

Starting with CMake 2.6.0 CPACK_SOURCE_STRIP_FILES will be a boolean variable which enables stripping of all files (a list of files evaluates to TRUE in CMake, so this change is compatible).

CPACK_STRIP_FILES

List of files to be stripped

Starting with CMake 2.6.0 CPACK_STRIP_FILES will be a boolean variable which enables stripping of all files (a list of files evaluates to TRUE in CMake, so this change is compatible).

CPACK_SYSTEM_NAME

System name, defaults to the value of ${CMAKE_SYSTEM_NAME}.

CPACK_TOPLEVEL_TAG

Directory for the installed files

CPACK_WARN_ON_ABSOLUTE_INSTALL_DESTINATION

Ask CPack to warn each time a file with absolute INSTALL DESTINATION is encountered.

This variable triggers the definition of:

```
CMAKE_WARN_ON_ABSOLUTE_INSTALL_DESTINATION
```

when CPack runs cmake_install.cmake scripts.

2. Variables specific to CPack Bundle generator

CPACK_BUNDLE_ICON

Path to an OS X icon file that will be used as the icon for the generated bundle.

This is the icon that appears in the OS X finder for the bundle, and in the OS X dock when the bundle is opened. *Required.*

CPACK_BUNDLE_NAME

The name of the generated bundle

This appears in the OS X finder as the bundle name. *Required.*

CPACK_BUNDLE_PLIST

Path to an OS X plist file that will be used for the generated bundle.

This assumes that the caller has generated or specified their own Info.plist file. *Required.*

CPACK_BUNDLE_STARTUP_COMMAND

Path to a startup script

This is a path to an executable or script that will be run whenever an end-user double-clicks the generated bundle in the OS X Finder. *Optional.*

3. Variables concerning CPack Components

CPACK_<GENNAME>_COMPONENT_INSTALL

Enable/Disable component install for CPack generator <GENNAME>.

Each CPack Generator (RPM, DEB, ARCHIVE, NSIS, DMG, etc...) has a legacy default behavior. e.g. RPM builds monolithic whereas NSIS builds component. One can change the default behavior by setting this variable to 0/1 or OFF/ON.

CPACK_COMPONENTS_ALL

The list of component to install

The default value of this variable is computed by CPack and contains all components defined by the project. The user may set it to only include the specified components.

CPACK_COMPONENTS_GROUPING

Specify how components are grouped for multi-package component-aware CPack generators.

Some generators like RPM or ARCHIVE family (TGZ, ZIP, ...) generates several packages files when asked for component packaging. They group the component differently depending on the value of this variable:

- ONE_PER_GROUP (default): creates one package file per component group
- ALL_COMPONENTS_IN_ONE: creates a single package with all (requested) components
- IGNORE : creates one package per component, i.e. IGNORE component group One can specify different grouping for different CPack generator by using a CPACK_PROJECT_CONFIG_FILE.

CPACK_COMPONENT_<compName>_DEPENDS

The dependencies (list of components) on which this component depends.

CPACK_COMPONENT_<compName>_DESCRIPTION

The description of a component

CPACK_COMPONENT_<compName>_DISPLAY_NAME

The name to be displayed for a component

CPACK_COMPONENT_<compName>_GROUP

The group of a component

CPACK_COMPONENT_<compName>_REQUIRED

True if this component is required

4. Variables specific to CPack Cygwin generator

`CPACK_CYGWIN_BUILD_SCRIPT`

The Cygwin build script

FIXME: This documentation is incomplete.

`CPACK_CYGWIN_PATCH_FILE`

The Cygwin patch file

FIXME: This documentation is incomplete.

`CPACK_CYGWIN_PATCH_NUMBER`

The Cygwin patch number

FIXME: This documentation is incomplete.

5. Variables specific to CPack Debian (DEB) generator

`CPACK_DEBIAN_PACKAGE_ARCHITECTURE`

Mandatory YES

Default Output of `dpkg --print-architecture` (or `i386` if `dpkg` is not found)

The debian package architecture

`CPACK_DEBIAN_PACKAGE_BREAKS`

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

When one binary package declares that it breaks another, `dpkg` will refuse to allow the package which declares Breaks be installed unless the broken package is deconfigured first, and it will refuse to allow the broken package to be reconfigured.

`CPACK_DEBIAN_PACKAGE_CONFLICTS`

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

When one binary package declares a conflict with another using a "Conflicts" field, `dpkg` will refuse to allow them to be installed on the system at the same time.

`CPACK_DEBIAN_PACKAGE_CONTROL_EXTRA`

Mandatory NO

Default —

This variable allow advanced user to add custom script to the `control.tar.gz` Typical usage is for `conffiles`, `postinst`, `postrm`, `prerm`.

Usage:

```
SET(CCSR "${CMAKE_CURRENT_SOURCE_DIR}")
SET(CPACK_DEBIAN_PACKAGE_CONTROL_EXTRA
    "${CCSR}/prerm;${CCSR}/postrm")
```

`CPACK_DEBIAN_PACKAGE_DEBUG`

Mandatory NO

Default —

May be set when invoking `cpack` in order to trace debug information during CPackDeb run.

`CPACK_DEBIAN_PACKAGE_DEPENDS`

Mandatory NO

Default —

May be used to set deb dependencies.

`CPACK_DEBIAN_PACKAGE_DESCRIPTION`

Mandatory YES

Default `CPACK_PACKAGE_DESCRIPTION_SUMMARY`

The debian package description

`CPACK_DEBIAN_PACKAGE_ENHANCES`

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

This field is similar to Suggests but works in the opposite direction. It is used to declare that a package can enhance the functionality of another package.

`CPACK_DEBIAN_PACKAGE_HOMEPAGE`

Mandatory NO

Default —

The URL of the web site for this package, preferably (when applicable) the site from which the original source can be obtained and any additional upstream documentation or information may be found. The content of this field is a simple URL without any surrounding characters such as <>.

`CPACK_DEBIAN_PACKAGE_MAINTAINER`

Mandatory YES

Default `CPACK_PACKAGE_CONTACT`

The debian package maintainer

`CPACK_DEBIAN_PACKAGE_NAME`

Mandatory YES

Default `CPACK_PACKAGE_NAME` (lower case)

The debian package summary

`CPACK_DEBIAN_PACKAGE_PREDEPENDS`

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

This field is like Depends, except that it also forces `dpkg` to complete installation of the packages named before even starting the installation of the package which declares the pre-dependency.

`CPACK_DEBIAN_PACKAGE_PRIORITY`

Mandatory YES

Default "optional"

The debian package priority

`CPACK_DEBIAN_PACKAGE_PROVIDES`

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

A virtual package is one which appears in the Provides control field of another package.

CPACK_DEBIAN_PACKAGE_RECOMMENDS

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

Allows packages to declare a strong, but not absolute, dependency on other packages.

CPACK_DEBIAN_PACKAGE_REPLACES

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

Packages can declare in their control file that they should overwrite files in certain other packages, or completely replace other packages.

CPACK_DEBIAN_PACKAGE_SECTION

Mandatory YES

Default "devel"

The debian package section

CPACK_DEBIAN_PACKAGE_SHLIBDEPS

Mandatory NO

Default OFF

May be set to ON in order to use `dpkg-shlibdeps` to generate better package dependency list. You may need set `CMAKE_INSTALL_RPATH` to appropriate value if you use this feature, because if you don't `dpkg-shlibdeps` may fail to find your own shared libs. See http://www.cmake.org/Wiki/CMake_RPATH_handling.

CPACK_DEBIAN_PACKAGE_SUGGESTS

Mandatory NO

Default —

see http://www.debian.org/doc/debian-policy/ch-relationships.html#s-binarydeps

Allows packages to declare a suggested package install grouping.

CPACK_DEBIAN_PACKAGE_VERSION

Mandatory YES

Default `CPACK_PACKAGE_VERSION`

The debian package version

6. Variables specific to CPack DragNDrop generator

CPACK_COMMAND_HDIUTIL

Path to the `hdiutil(1)` command used to operate on disk image files on Mac OS X.

This variable can be used to override the automatically detected command (or specify its location if the auto-detection fails to find it.)

CPACK_COMMAND_REZ

Path to the `Rez(1)` command used to compile resources on Mac OS X.

This variable can be used to override the automatically detected command (or specify its location if the auto-detection fails to find it.)

CPACK_COMMAND_SETFILE

Path to the `SetFile(1)` command used to set extended attributes on files and directories on Mac OS X.

This variable can be used to override the automatically detected command (or specify its location if the auto-detection fails to find it.)

CPACK_DMG_BACKGROUND_IMAGE

Path to a background image file

This file will be used as the background for the Finder Window when the disk image is opened. By default no background image is set. The background image is applied after applying the custom `.DS_Store` file.

CPACK_DMG_DS_STORE

Path to a custom DS_Store file

This `.DS_Store` file e.g. can be used to specify the Finder window position/geometry and layout (such as hidden toolbars, placement of the icons etc.). This file has to be generated by the Finder (either manually or through OSA-script) using a normal folder from which the `.DS_Store` file can then be extracted.

CPACK_DMG_FORMAT

The disk image format

Common values are UDRO (UDIF read-only), UDZO (UDIF zlib-compressed) or UDBZ (UDIF bzip2-compressed). Refer to `hdiutil(1)` for more information on other available formats.

CPACK_DMG_VOLUME_NAME

The volume name of the generated disk image.

Defaults to `CPACK_PACKAGE_FILE_NAME`.

7. Variables specific to CPack NSIS generator

CPACK_NSIS_COMPRESSOR

The arguments that will be passed to the NSIS `SetCompressor` command.

CPACK_NSIS_CONTACT

Contact information for questions and comments about the installation process.

CPACK_NSIS_CREATE_ICONS_EXTRA

Additional NSIS commands for creating start menu shortcuts.

CPACK_NSIS_DELETE_ICONS_EXTRA

Additional NSIS commands to uninstall start menu shortcuts.

CPACK_NSIS_DISPLAY_NAME

The display name string that appears in the Windows Add/Remove Program control panel

CPACK_NSIS_ENABLE_UNINSTALL_BEFORE_INSTALL

Ask about uninstalling previous versions first.

If this is set to "ON", then an installer will look for previous installed versions and if one is found, ask the user whether to uninstall it before proceeding with the install.

CPACK_NSIS_EXECUTABLES_DIRECTORY

Creating NSIS start menu links assumes that they are in "bin" unless this variable is set.

For example, you would set this to "exec" if your executables are in an exec directory.

`CPACK_NSIS_EXTRA_INSTALL_COMMANDS`

Extra NSIS commands that will be added to the end of the install Section, after your install tree is available on the target system.

`CPACK_NSIS_EXTRA_PREINSTALL_COMMANDS`

Extra NSIS commands that will be added to the beginning of the install Section, before your install tree is available on the target system.

`CPACK_NSIS_EXTRA_UNINSTALL_COMMANDS`

Extra NSIS commands that will be added to the uninstall Section, before your install tree is removed from the target system.

`CPACK_NSIS_HELP_LINK`

URL to a web site providing assistance in installing your application.

`CPACK_NSIS_INSTALLED_ICON_NAME`

A path to the executable that contains the installer icon.

`CPACK_NSIS_INSTALLER_MUI_ICON_CODE`

undocumented

`CPACK_NSIS_INSTALL_ROOT`

The default installation directory presented to the end user by the NSIS installer is under this root dir.

The full directory presented to the end user is:

```
${CPACK_NSIS_INSTALL_ROOT}/${CPACK_PACKAGE_INSTALL_DIRECTORY}
```

`CPACK_NSIS_MENU_LINKS`

Specify links in [application] menu

This should contain a list of pair "link" "link name". The link may be an URL or a path relative to installation prefix. Like:

```
set(VER "@CMake_VERSION_MAJOR@.@CMake_VERSION_MINOR@")
set(CPACK_NSIS_MENU_LINKS
    "doc/cmake-${VER}/cmake.html"
    "CMake Help"
    "http://www.cmake.org"
    "CMake Web Site")
```

`CPACK_NSIS_MODIFY_PATH`

Modify PATH toggle

If this is set to "ON", then an extra page will appear in the installer that will allow the user to choose whether the program directory should be added to the system PATH variable.

`CPACK_NSIS_MUI_FINISHPAGE_RUN`

Specify an executable to add an option to run on the finish page of the NSIS installer.

`CPACK_NSIS_MUI_ICON`

An icon filename

The name of an `.ico` file used as the main icon for the generated install program.

`CPACK_NSIS_MUI_UNIICON`

An icon filename

The name of a `.ico` file used as the main icon for the generated uninstall program.

`CPACK_NSIS_PACKAGE_NAME`

The title displayed at the top of the installer.

`CPACK_NSIS_URL_INFO_ABOUT`

URL to a web site providing more information about your application.

8. Variables specific to CPack PackageMaker generator

`CPACK_OSX_PACKAGE_VERSION`

The version of Mac OS X that the resulting PackageMaker archive should be compatible with.

Different versions of Mac OS X support different features. For example, CPack can only build component-based installers for Mac OS X 10.4 or newer, and can only build installers that download component son-the-fly for Mac OS X 10.5 or newer. If left blank, this value will be set to the minimum version of Mac OS X that supports the requested features. Set this variable to some value (e.g., 10.4) only if you want to guarantee that your installer will work on that version of Mac OS X, and don't mind missing extra features available in the installer shipping with later versions of Mac OS X.

9. Variables specific to CPack RPM generator

`CPACK_RPM_CHANGELOG_FILE`

RPM changelog file

Mandatory NO

Default —

May be used to embed a changelog in the spec file. The refered file will be read and directly put after the `%changelog` section.

`CPACK_RPM_COMPRESSION_TYPE`

RPM compression type

Mandatory NO

Default —

May be used to override RPM compression type to be used to build the RPM. For example some Linux distribution now default to lzma or xz compression whereas older cannot use such RPM. Using this one can enforce compression type to be used. Possible value are: `lzma`, `xz`, `bzip2` and `gzip`.

`CPACK_RPM_GENERATE_USER_BINARY_SPECFILE_TEMPLATE`

Spec file template

Mandatory NO

Default —

If set CPack will generate a template for USER specified binary spec file and stop with an error. For example launch CPack like:

```
cpack \
  -D CPACK_RPM_GENERATE_USER_BINARY_SPECFILE_TEMPLATE=1 \
  -G RPM
```

The user may then use this file in order to hand-craft is own binary spec file which may be used with CPACK_RPM_USER_BINARY_SPECFILE.

CPACK_RPM_PACKAGE_ARCHITECTURE

The RPM package architecture

Mandatory NO

Default —

This may be set to "noarch" if you know you are building a noarch package.

CPACK_RPM_PACKAGE_DEBUG

Toggle CPackRPM debug output

Mandatory NO

Default —

May be set when invoking cpack in order to trace debug information during CPack RPM run. For example you may launch CPack like:

```
cpack -D CPACK_RPM_PACKAGE_DEBUG=1 -G RPM
```

CPACK_RPM_PACKAGE_DESCRIPTION

RPM package description

Mandatory YES

Default CPACK_PACKAGE_DESCRIPTION_FILE if set or "no package description available"

CPACK_RPM_PACKAGE_GROUP

The RPM package group

Mandatory YES

Default "unknown"

CPACK_RPM_PACKAGE_LICENSE

The RPM package license policy

Mandatory YES

Default "unknown"

CPACK_RPM_PACKAGE_NAME

The RPM package name

Mandatory YES

Default CPACK_PACKAGE_NAME

CPACK_RPM_PACKAGE_OBSOLETES

RPM spec obsoletes field

Mandatory NO

Default —

May be used to set RPM packages that are obsoleted by this one.

CPACK_RPM_PACKAGE_PROVIDES

RPM spec provides field

Mandatory NO

Default —

May be used to set RPM dependencies (provides). The provided package list of an RPM file could be printed with:

```
rpm -qp --provides file.rpm
```

CPACK_RPM_PACKAGE_RELEASE

The RPM package release

Mandatory YES

Default 1

This is the numbering of the RPM package itself, i.e. the version of the packaging and not the version of the content (see CPACK_RPM_PACKAGE_VERSION). One may change the default value if the previous packaging was buggy and/or you want to put here a fancy Linux distro specific numbering.

CPACK_RPM_PACKAGE_RELOCATABLE

build a relocatable RPM

Mandatory NO

Default CPACK_PACKAGE_RELOCATABLE

If this variable is set to TRUE or ON CPackRPM will try to build a relocatable RPM package. A relocatable RPM may be installed using rpm –prefix or --relocate in order to install it at an alternate place see rpm(8). Note that currently this may fail if CPACK_SET_DESTDIR is set to ON. If CPACK_SET_DESTDIR is set then you will get a warning message but if there is file installed with absolute path you'll get unexpected behavior.

CPACK_RPM_PACKAGE_REQUIRES

RPM spec requires field

Mandatory NO

Default —

May be used to set RPM dependencies (requires). Note that you must enclose the complete requires string between quotes, for example:

```
set(CPACK_RPM_PACKAGE_REQUIRES "python >= 2.5.0, cmake >= 2.8")
```

The required package list of an RPM file could be printed with:

```
rpm -qp --requires file.rpm
```

CPACK_RPM_PACKAGE_SUGGESTS

RPM spec suggest field

Mandatory NO

Default —

May be used to set weak RPM dependencies (suggests). Note that you must enclose the complete requires string between quotes.

CPACK_RPM_PACKAGE_SUMMARY

The RPM package summary

Mandatory YES

Default CPACK_PACKAGE_DESCRIPTION_SUMMARY

CPACK_RPM_PACKAGE_URL

The project's URL

Mandatory NO

Default —

CPACK_RPM_PACKAGE_VENDOR

The RPM package vendor

Mandatory YES

Default CPACK_PACKAGE_VENDOR if set or "unknown"

CPACK_RPM_PACKAGE_VERSION

The RPM package version

Mandatory YES

Default CPACK_PACKAGE_VERSION

CPACK_RPM_POST_INSTALL_SCRIPT_FILE/CPACK_RPM_POST_UNINSTALL_SCRIPT_FILE

Mandatory NO

Default —

May be used to embed a post (un)installation script in the spec file. The refered script file(s) will be read and directly put after the %post or %postun section If CPACK_RPM_COMPONENT_INSTALL is set to ON the (un)install script for each component can be overridden with:

```
CPACK_RPM_<COMPONENT>_POST_INSTALL_SCRIPT_FILE
```

and:

```
CPACK_RPM_<COMPONENT>_POST_UNINSTALL_SCRIPT_FILE
```

One may verify which scriptlet has been included with:

```
rpm -qp --scripts package.rpm
```

CPACK_RPM_PRE_INSTALL_SCRIPT_FILE/CPACK_RPM_PRE_UNINSTALL_SCRIPT_FILE

Mandatory NO

Default —

May be used to embed a pre (un)installation script in the spec file. The refered script file(s) will be read and directly put after the %pre or %preun section If CPACK_RPM_COMPONENT_INSTALL is set to ON the (un)install script for each component can be overridden with:

```
CPACK_RPM_<COMPONENT>_PRE_INSTALL_SCRIPT_FILE
```

and:

```
CPACK_RPM_<COMPONENT>_PRE_UNINSTALL_SCRIPT_FILE
```

One may verify which scriptlet has been included with:

```
rpm -qp --scripts package.rpm
```

CPACK_RPM_SPEC_INSTALL_POST

[deprecated]

Mandatory NO

Default —

This way of specifying post-install script is deprecated, use:

```
CPACK_RPM_POST_INSTALL_SCRIPT_FILE
```

May be used to set an RPM post-install command inside the spec file. For example setting it to "/bin/true" may be used to prevent rpmbuild to strip binaries.

CPACK_RPM_SPEC_MORE_DEFINE

RPM extended spec definitions lines

Mandatory NO

Default —

May be used to add any %define lines to the generated spec file.

CPACK_RPM_USER_BINARY_SPECFILE

A user provided spec file

Mandatory NO

Default —

May be set by the user in order to specify a USER binary spec file to be used by CPackRPM instead of generating the file. The specified file will be processed by CONFIGURE_FILE(@ONLY).

CPACK_RPM_USER_FILELIST/CPACK_RPM_<COMPONENT>_USER_FILELIST

Mandatory NO

Default —

May be used to explicitly specify %(<directive>) file line in the spec file. Like %config(noreplace) or any other directive that be found in the %files section. Since CPackRPM is generating the list of files (and directories) the user specified files of the:

```
CPACK_RPM_<COMPONENT>_USER_FILELIST
```

list will be removed from the generated list.

10. Copyright

11. See Also

The following resources are available to get help using CMake:

Home Page

Home page

The primary starting point for learning about CMake.

Frequently Asked Questions

FAQ

A Wiki is provided containing answers to frequently asked questions.

Online Documentation

Online docs

Links to available documentation may be found on this web page.

Mailing List

Mailing list

For help and discussion about using cmake, a mailing list is provided at cmake@cmake.org. The list is member-post-only but one may sign up on the CMake web page. Please first read the full documentation at http://www.cmake.org before posting questions to the list.

Summary of helpful links:

- Home
- Docs
- Mail
- FAQ