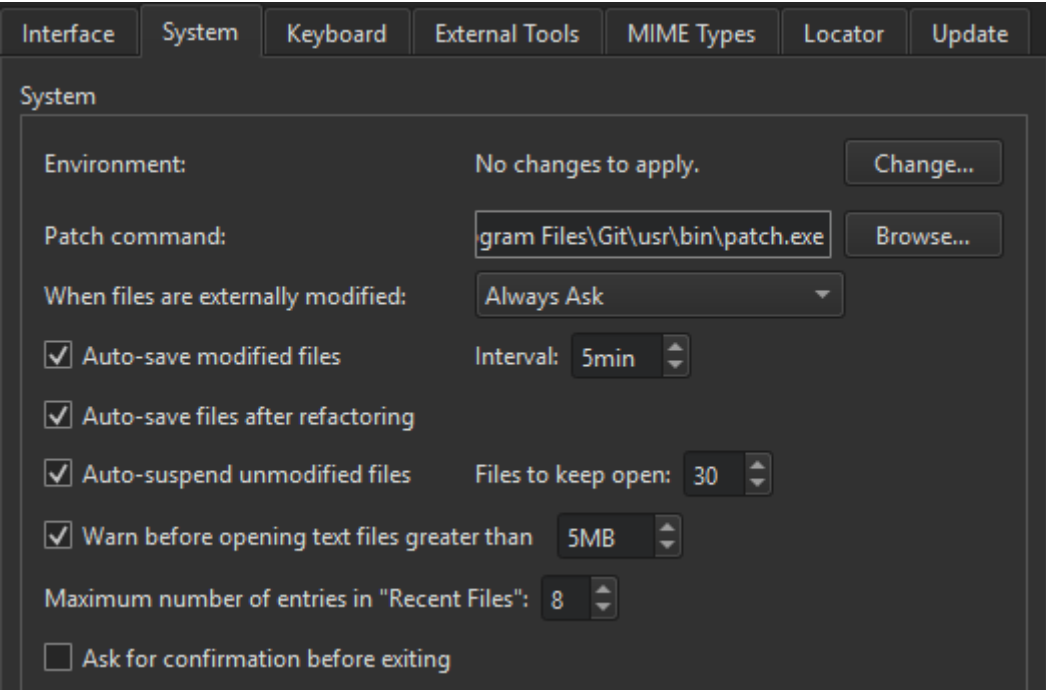


# Specifying Environment Settings

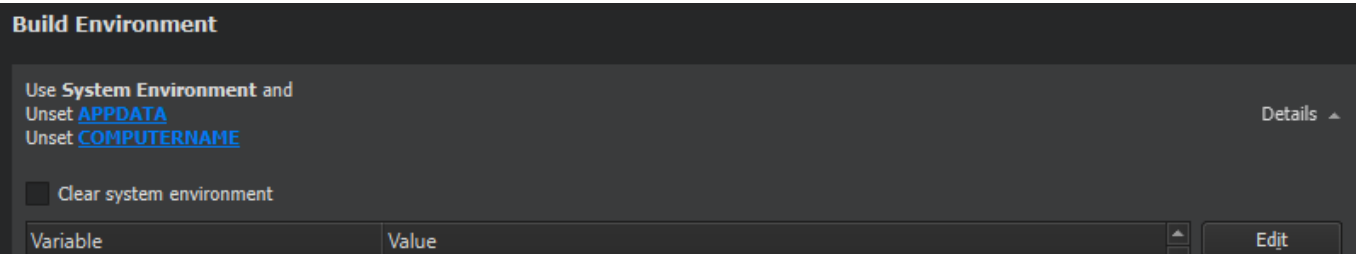
You can specify the environment you want to use for building a project in the **Build Environment** section of the **Build Settings**.

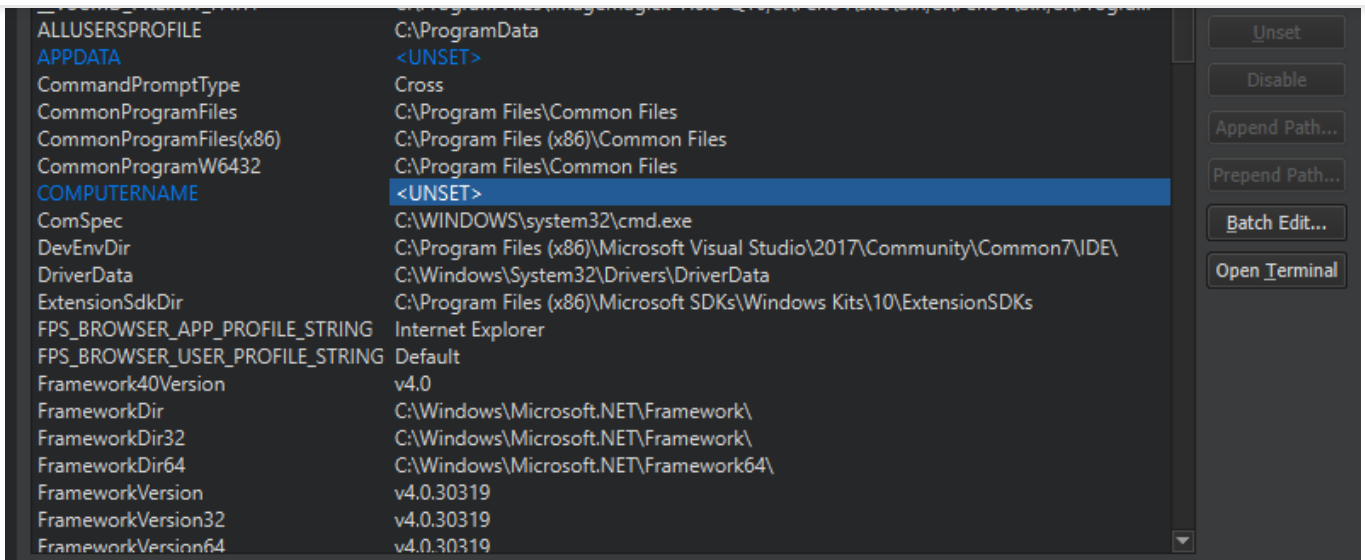
By default, the environment in which Qt Creator was started is used and modified to include the Qt version. Depending on the selected Qt version, Qt Creator automatically sets the necessary environment variables. You can edit existing environment variables or add, reset and unset new variables based on your project requirements.

To globally change the system environment from the one in which Qt Creator is started, select **Edit > Preferences > Environment > System**, and then select **Change** in the **Environment** field.



In addition, you can specify custom environment variables in the **Project Settings > Environment** settings. They are added to all build environments. The final build environment is specified separately for each kit. The project-specific environment settings enable you to amend the build environment for all kits that you use to build the project simultaneously, rather than having to edit it separately for each kit.





The changes are stored in the local project specific `CMakeLists.txt.user` or `.pro.user` file, depending on the build system you use. Therefore, they are not suitable for sharing between developers or development PCs. To share settings, incorporate them into the build system. For example, if you use CMake, make the changes in the `CMakeLists.txt` file, and if you use qmake, make the changes in the `.pro` file.

## Batch Editing

To modify environment variable values for build or run environments, select **Batch Edit** in the **Build Environment** or **Environment** pane and enter environment variables in the **Edit Environment** dialog.

To remove a variable value from the environment, enter the variable name. For example, `TEST` sets the value of the `TEST` variable empty when building or running the project.

To add a variable value to the environment, enter the variable name and value, separated by the equals sign. For example, the following line prepends the `/opt/bin` folder to the existing `PATH`:

- › On Windows: `PATH=C:\opt\bin;${PATH}`
- › On Linux: `PATH=/opt/bin:${PATH}`

To add or remove several variables, place them on separate lines. The order is important. If you remove a value on a line, you cannot refer to it on the following lines. However, you can remove a value after you have referred to it on an earlier line.

To temporarily disable a variable, add a hash character (`#`) to the beginning of the line.

## Clearing the System Environment

To build with a clean system environment, select the **Clear system environment** check box. Qt Creator discards the current environment, and populates a clean system environment with the environment variables that the compilers and tools need. Therefore, the environment is never totally empty, even after you clear it.


## Using Environment Variables

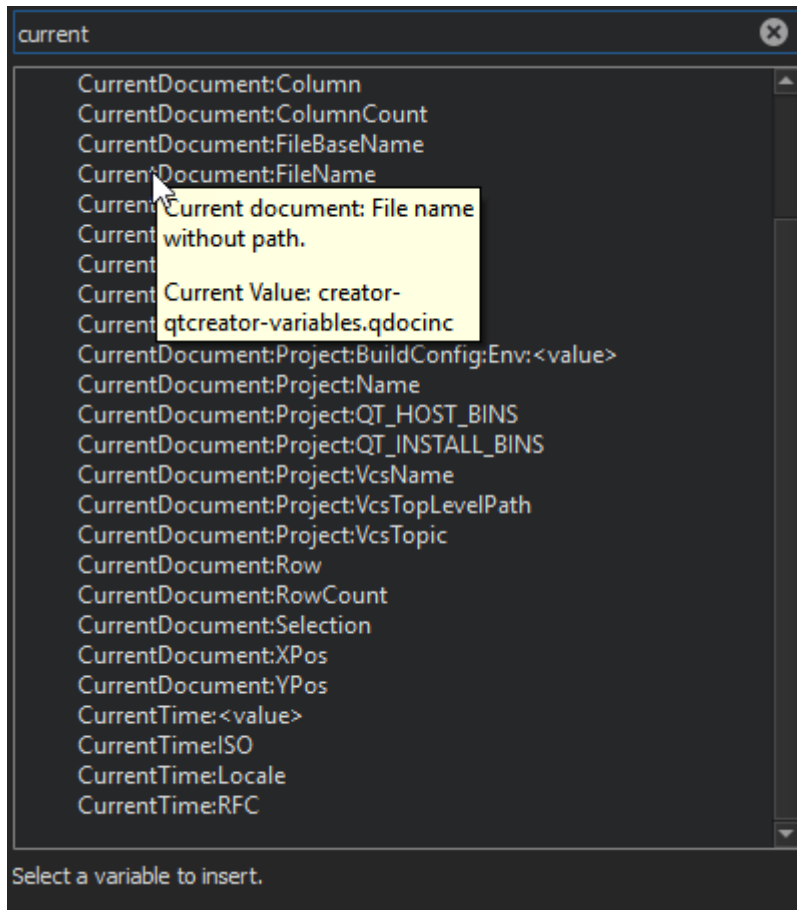
You can use any environment variables in build, deploy, and run configurations. For a list of variable names, select **Build Settings > Build Environment > Details**. Environment variables are referenced using the native syntax:

`$VARIABLE` or `%VARIABLE%` on Unix and `%VARIABLE%` on Windows

## Using Qt Creator Variables

You can use Qt Creator variables in arguments, executable paths, and working directories. The variables take care of quoting their expansions, so you do not need to put them in quotes.

Select the  (**Variables**) button in a field to select from a list of variables that are available in a particular context. For more information about each variable, move the cursor over it in the list.



The following syntax enables you to use environment variables as Qt Creator variables: `%{Env:VARNAME}`.

Qt Creator uses pattern substitution when expanding variable names. To replace the first match of *pattern* within *variable* with *replacement*, use:

```
%{variable/pattern/replacement}
```

To replace all matches of *pattern* within *variable* with *replacement*, use:

```
%{variable//pattern/replacement}
```

The pattern can be a regular expression and the replacement can contain backreferences. For example, if `%{variable}` is `my123var`, then `%{variable/(..)(\d+)/\2\1}` is expanded to `123myvar`.

Instead of the forward slash, you can also use the pound sign (`#`) as the substitution character. This can be helpful if

To use the default value if the variable is not set, use:

```
%{variable:-default}
```

< Specifying Dependencies

Using Custom Output Parsers >

© 2022 The Qt Company Ltd. Documentation contributions included herein are the copyrights of their respective owners. The documentation provided herein is licensed under the terms of the [GNU Free Documentation License version 1.3](#) as published by the Free Software Foundation. Qt and respective logos are trademarks of The Qt Company Ltd in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.



Contact Us

Company

- About Us
- Investors
- Newsroom
- Careers
- Office Locations

Licensing

- Terms & Conditions
- Open Source
- FAQ

Support

- Support Services
- Professional Services
- Partners
- Training

For Customers

- Support Center
- Downloads
- Qt Login
- Contact Us
- Customer Success

Community

- Contribute to Qt
- Forum
- Wiki
- Downloads

