EditorConfig Specification

Release 0.14.0

May 03, 2021

Contents

1 Introduction (informative) 3

2 Terminology 5

3 File Format 7

4 Glob Expressions 9

5 File Processing 11

6 Supported Pairs 13

7 Suggestions for Plugin Developers 15

8 Versioning 17

This is version 0.14.0 of this specification.

Table of Contents

- *EditorConfig Specification*
  - *Introduction (informative)*
  - *Terminology*
  - *File Format*
  - *Glob Expressions*
  - *File Processing*
  - *Supported Pairs*
  - *Suggestions for Plugin Developers*
  - *Versioning*

CHAPTER 1

Introduction (informative)

All content in this document is normative unless marked "(informative)".

EditorConfig helps maintain consistent coding styles for multiple developers working on the same project across various editors and IDEs. The EditorConfig project consists of a file format for defining coding styles and a collection of text editor plugins that enable editors to read the file format and adhere to defined styles. EditorConfig files are easily readable and they work nicely with version control systems.

CHAPTER 2

Terminology

In EditorConfig:

- "EditorConfig files" (usually named `.editorconfig`) store settings, and must conform to this specification.
- "Cores" parse files conforming to this specification.
- "Plugins" apply settings to files being edited, and use cores to determine the settings.
- "Editors" permit editing files, and use plugins to apply settings.

A conforming core or plugin must pass the tests in the core-tests repository or plugin-tests repository, respectively.

(informative) Some plugins include or bundle their own cores, and some rely on external cores. Some editors include or bundle plugin or core functionality. Editors, plugins, and cores may all come from different people. Those people may or may not have any direct interaction with the EditorConfig organization.

CHAPTER 3

File Format

EditorConfig files are in an INI-like file format. In an EditorConfig file, all beginning whitespace on each line is considered irrelevant. Each line must be one of the following:

- Blank: contains only whitespace characters.
- **Comment: starts with a `;` or a `#`.**
  - Inserting a `#` or `;` after non-whitespace characters in a line (i.e., inline) shall neither be parsed as a comment nor as part of the section name, pair (defined below) key or value in which it was inserted. This may change in the future; thus, is not recommended.
- **Section Header: starts with a `[` and ends with a `]`.**
  - May not use any non-whitespace characters outside of the surrounding brackets.
  - May contain any characters between the square brackets (e.g., `[` and `]` and even spaces and tabs are allowed).
  - Forward slashes (`/`) are used as path separators.
  - Backslashes (`\\`) are not allowed as path separators (even on Windows).
- **Key-Value Pair (or Pair): contains a key and a value, separated by an `=`.**
  - Key: the part before the first `=` (trimmed of whitespace).
  - Value: The part after the first `=` (trimmed of whitespace).

Any line that is not one of the above is invalid.

EditorConfig files should be UTF-8 encoded, with LF or CRLF line separators.

Additionally, EditorConfig defines the following terms:

- Preamble: the lines that precedes the first section. The preamble is optional and may contain key-value pairs, comments and blank lines.
- Section Name: the string between the beginning `[` and the ending `]`.
- Section: the lines starting from a Section Header until the beginning of the next Section Header or the end of the file.

CHAPTER 4

Glob Expressions

Section names in EditorConfig files are filepath globs, similar to the format accepted by `.gitignore`. They support pattern matching through Unix shell-style wildcards. These filepath globs recognize the following as special characters for wildcard matching:

Special Characters	Matching
`*`	any string of characters, except path separators (`/`)
`**`	any string of characters
`?`	any single character, except path separators (`/`)
`[seq]`	any single character in seq
`[!seq]`	any single character not in seq
`{s1,s2,s3}`	any of the strings given (separated by commas, can be nested)
`{num1..num2}`	any integer numbers between `num1` and `num2`, where `num1` and `num2` can be either positive or negative

The backslash character (`\\`) can be used to escape a character so it is not interpreted as a special character.

Cores must accept section names with length up to and including 1024 characters. Beyond that, each implementation may choose to define its own upper limit or no explicit upper limit at all.

CHAPTER 5

File Processing

When a filename is given to EditorConfig a search is performed in the directory of the given file and all parent directories for an EditorConfig file (named ".editorconfig" by default). Non-existing directories are treated as if they exist and are empty. All found EditorConfig files are searched for sections with section names matching the given filename. The search shall stop if an EditorConfig file is found with the `root` key set to `true` in the preamble or when reaching the root filesystem directory.

Files are read top to bottom and the most recent rules found take precedence. If multiple EditorConfig files have matching sections, the rules from the closer EditorConfig file are read last, so pairs in closer files take precedence.

CHAPTER 6

Supported Pairs

EditorConfig file sections contain key-value pairs separated by an equal sign (=). With the exception of the `root` key, all pairs MUST be located under a section to take effect. EditorConfig plugins shall ignore unrecognized keys and invalid/unsupported values for those keys.

Here is the list of all keys defined by this version of this specification, and the supported values associated with them:

Key	Supported values
`indent_style`	Set to `tab` or `space` to use hard tabs or soft tabs respectively. The values are case insensitive.
`indent_size`	Set to a whole number defining the number of columns used for each indentation level and the width of soft tabs (when supported). If this equals `tab`, the `indent_size` shall be set to the tab size, which should be `tab_width` (if specified); else, the tab size set by the editor. The values are case insensitive.
`tab_width`	Set to a whole number defining the number of columns used to represent a tab character. This defaults to the value of `indent_size` and should not usually need to be specified.
`end_of_line`	Set to `lf`, `cr`, or `crlf` to control how line breaks are represented. The values are case insensitive.
`charset`	Set to `latin1`, `utf-8`, `utf-8-bom`, `utf-16be` or `utf-16le` to control the character set. Use of `utf-8-bom` is discouraged.
`trim_trailing_whitespace`	Set to `true` to remove all whitespace characters preceding newline characters in the file and `false` to ensure it doesn't.
`insert_final_newline`	Set to `true` to ensure file ends with a newline when saving and `false` to ensure it doesn't.
`root`	Must be specified in the preamble. Set to `true` to stop the `.editorconfig` file search on the current file. The value is case insensitive.

For any pair, a value of `unset` removes the effect of that pair, even if it has been set before. For example, add `indent_size = unset` to undefine the `indent_size` pair (and use editor defaults).

Pair keys are case insensitive. All keys are lowercased after parsing.

Cores must accept keys and values with lengths up to and including 1024 and 4096 characters respectively. Beyond that, each implementation may choose to define its own upper limits or no explicit upper limits at all.

CHAPTER 7

Suggestions for Plugin Developers

TODO. For now please read the Plugin Guidelines on GitHub wiki.

CHAPTER 8

Versioning

This section applies beginning with version 0.14.0 of this specification.

This specification has a version, tagged in the specification repository. Each specification version corresponds to the same version in the core-tests repository.

The version numbering of the specification follows Semantic Versioning 2.0.0 ("SemVer"). The version numbering of the core-tests repository also follows SemVer.

Each EditorConfig core, to pass the core tests, must process version numbers given with the `-b` switch, and must report version numbers when given `-v` or `--version`. The version numbers used for `-b`, `-v`, and `--version` are versions of this specification. For example, the Vimscript core might respond to `-v` with:

```
EditorConfig Vimscript core v1.0.0 - Specification Version 0.14.0
```

Cores, plugins, or editors supporting EditorConfig have their own version numbers. Those version numbers are independent of the version number of this specification.