Syntax Highlighting for kaiv

A Guide

The kaiv Project

for kaiv 0.6.0

1 Introduction

Your editor can speak kaiv. Two plugins cover the two editor families we use ourselves, and a small language server adds live validation on top of both:

- `editors` — a VS Code extension: a TextMate grammar for the whole kaiv file family plus a client that wires in the language server when it is installed.
- `kaiv-vim` — a Vim/Neovim plugin: filetype detection and a syntax file mirroring the same grammar.
- `kaiv-lsp` — a language server in the `kaiv-rs` repository. It runs the reference pipeline over every open document and publishes the errors your build would hit — the same errors, with the same names, at the same lines.

kaiv is in alpha, and its editor tooling is too: the plugins on this page are installed manually, from source; the language server is one `cargo install` away. When kaiv reaches beta, the plugins will be promoted to the standard channels — the VS Code Marketplace and Open VSX, the usual Vim plugin managers — and the manual steps below will collapse into the one-liners you expect. Section 6 says exactly what alpha means here.

2 One grammar, ten extensions

Every file kind in the kaiv family — authored data, canonical data, schemas, compiled schemas, type libraries, unit definitions, queries,

mappings — shares one line grammar. A line is classified by what it starts with (a comment sign, a declaration sigil, a block delimiter, a metadata leader) or by the first `=` it contains; nesting lives in namepaths, not indentation. That regularity is what makes precise highlighting cheap: one grammar covers the entire family, and both editors register it for all ten extensions:

```
.kaiv .raiv .daiv .saiv .csaiv .taiv .qaiv .faiv .maiv
                        .msaiv
```

What you see colored, in both editors:

Construct	Example
comments and doc comments	`# note // doc`
declarations	`.!kaiv .!saiv 1 acme/fleet`
structure lines	`[/@servers] (/server) []`
table headers	`[/@servers host=!,port=! max=3]`
type annotations	`!int !float:km !str#[2,8]`
constraints	`[1,65535] {red,green} /^a+$/`
namepaths and sigils	`/server/api::port @ ::`
operators	`= += ;= := +:=`
variables and references	`.name $.name $/.ns`
provenance	`?sensor1@20250115T093000Z`
units	`m 1.495978707e11 &au=`

The grammar is deliberately permissive across the family variants — a schema-only construct will still color inside a data file. Correctness is not the grammar's job; it is the language server's, which runs the real pipeline (Section 5).

3 VS Code

Clone the extension and run the install script:

```
$ git clone https://gitlab.com/kaiv-format/editors.git
$ editors/scripts/install-vscode.sh
```

The script builds the extension with npm, packages a VSIX, and installs it with `code --install-extension`. If you prefer a development-mode install (a symlink into `~/.vscode/extensions`, so a `git pull` updates it in place), pass `--symlink`.

Opening any file in the family now highlights it. Two settings control the language server integration: `kaiv.lsp.enabled` (default on) and `kaiv.lsp.path` (where to find `kaiv-lsp`; when empty, `PATH` and then `~/.cargo/bin` are searched). Without the server the extension stays grammar-only and says so once.

4 Vim and Neovim

The plugin's roots (`ftdetect/`, `syntax/`, `ftplugin/`) sit at the top of the repository, so the repository *is* the plugin. With pathogen-style bundles, cloning it into place is the whole install:

```
$ git clone https://gitlab.com/kaiv-format/kaiv-vim.git \
    ~/.vim/bundle/kaiv-vim
```

For Vim 8+ / Neovim native packages, the bundled script symlinks the repository into the package directories instead:

```
$ git clone https://gitlab.com/kaiv-format/kaiv-vim.git
$ kaiv-vim/install.sh
```

Either way, any file in the family gets `filetype=kaiv`, the syntax rules, and `#` as the comment string.

5 Live diagnostics: kaiv-lsp

Highlighting tells you what a line is; the language server tells you whether it is *valid*. `kaiv-lsp` is a thin stdio server over the reference implementation: every time you open or edit a document, it runs the pipeline stage the extension calls for — authored data compiles and denormalizes, schemas compile, compiled schemas parse, type libraries check — and the first error appears as a diagnostic on its line, carrying the spec's stable error name as its code.

They are exactly the errors the toolchain reports. Take a file with a malformed constraint on line 3:

```
.!kaiv
host=x
!int[1;2]
port=8080
```

The CLI says:

```
$ kaiv build editor-broken.kaiv
kaiv: INVALID_CONSTRAINT_ERROR (line 3)
```

and your editor shows the same `INVALID_CONSTRAINT_ERROR` squiggled across line 3 as you type, clearing the moment you fix it. Diagnostics are line-granular in this release — the pipeline reports lines, not columns, so the whole line is underlined.

Install the server with cargo:

```
$ cargo install kaiv-lsp
```

VS Code picks it up automatically on the next editor restart. Neovim (0.11+) needs a client config in your `init.lua` — the plugin ships it in `nvim/kaiv-lsp.lua`:

```
vim.lsp.config('kaiv_lsp', {
  cmd = { 'kaiv-lsp' },
  filetypes = { 'kaiv' },
  root_markers = { '.git' },
})
vim.lsp.enable('kaiv_lsp')
```

Resolution mirrors the CLI: the server honors the nearest `kaiv.kaiv` configuration up from the document's directory, and resolves offline otherwise. Query files (`.qaiv`) get highlighting but no diagnostics yet; validating a `.daiv` against its schema (rather than for well-formedness) is likewise a later increment.

6 Alpha now, standard channels at beta

The manual installs above are a deliberate alpha-stage choice, not an accident. While kaiv is in alpha the spec reserves the right to refine syntax before setting it in stone — and a marketplace extension that lags a syntax refinement would mis-highlight with authority. Installing from source keeps the tooling honestly pinned to the moving spec: a `git pull` and re-install tracks it.

When the spec is finalized and kaiv becomes beta, the tooling is promoted to the standard channels:

- the VS Code extension to the **VS Code Marketplace** and **Open VSX**,
- `kaiv-vim` to the usual plugin managers (any `Plug`/`packer`/`lazy` one-liner already works today, pointed at the repository).

The language server is ahead of the curve: `kaiv-lsp` is on **crates.io** already, so its install is final.
The repositories will remain the source of truth either way; the channels only remove the manual steps. Until then, if the highlighting and the toolchain ever disagree, the toolchain is right — and we would like to hear about it in the issue tracker.