

zapier.georgelambert.org · Markdown indexes
VERAE TIME × ZAPIER

cli

site/docs/modules/verae-fleet/cli.md

verae-fleet/cli

Package: verae-fleet

Source: packages/verae-fleet/src/cli.js

Lines: 146

What this module is

Implementation module in `verae-fleet`. The tables below are extracted from the source (signatures + JSDoc).

Exports

No named `export` / `module.exports` keys detected.

Types / interfaces / classes

None extracted.

Functions

Name	Parameters	Param types (JSDoc)	Returns	Calls (same file / helpers)
<code>print</code>	<code>obj</code>	—	<code>unknown</code>	see Call graph
<code>help</code>	(none)	—	<code>unknown</code>	see Call graph
<code>api</code>	<code>method</code> , <code>pathname</code>	—	<code>unknown</code>	see Call graph
<code>main</code>	(none)	—	<code>unknown</code>	see Call graph

Methods (class / object)

Name	Parameters
print	`verae-fleet`

list services, config files, replica min/max serve start floors + monitor + UI `${BASE}/` monitor same without printing extra status GET daemon status start turn on, spawn up to min stop `<service|instance>` turn off (service disable | | print | `r.body` | | print | { error: err.message, hint: 'Is `verae-fleet serve` running?' }` |

What it imports / requires

- `node:http`
- `./load.js`
- `./supervisor.js`
- `./monitor.js`
- `./server.js`

Call graph (identifiers invoked)

`loadFleet`, `slice`, `print`, `write`, `stringify`, `help`, `off`, `pause`, `json`, `identityFile`, `login`, `api`, `request`, `on`, `push`, `concat`, `toString`, `resolve`, `parse`, `end`, `main`, `listServices`, `padEnd`, `padStart`, `import`, `find`, `sshCheck`, `startControlServer`, `keys`, `startService`, `tick`, `start`, `async`, `stop`, `stopAll`, `exit`, `test`

Each identifier is a call site in this file. Follow the import list to see the defining module; open that module's MD for parameter and return types.

Return values (how to read this)

- HTTP route handlers return Express `res.json(...)` bodies (see route docs).
- Zapier `perform` functions return a **single object** (creates) or an **array** (triggers/searches).
- Pricing functions return integer **cents** on `Quote.totalCents`.