Documentation Standards

- Command annotation
- Class annotation
- Messages within command execution
- Command parameter description

To promote similarity and consistency between WP-CLI commands, we've produced these documentation standards we'd encourage you to follow. Commonality is a key component of usability, because it reduces the mental overhead required when switching between commands.

Pull requests against the WP-CLI project are reviewed for these standards. Please follow these standards for your custom commands as well.

Command annotation

Here's an example of the PHPdoc annotating the `wp cron event schedule` command:

```
/**
 * Schedule a new cron event.
 *
 * ## OPTIONS
 *
 * <hook>
 * : The hook name.
 *
 * [<next-run>]
 * : A Unix timestamp or an English textual datetime description compatible with
`strtotime()`. Defaults to now.
 *
 * [<recurrence>]
 * : How often the event should recur. See `wp cron schedule list` for available schedule
names. Defaults to no recurrence.
 *
```

```
 * [--<field>=<value>]
 * : Associative args for the event.
 *
 * ## EXAMPLES
 *
 *     # Schedule a new cron event.
 *     $ wp cron event schedule cron_test
 *     Success: Scheduled event with hook 'cron_test' for 2016-05-31 10:19:16 GMT.
 *
 *     # Schedule new cron event with hourly recurrence.
 *     $ wp cron event schedule cron_test now hourly
 *     Success: Scheduled event with hook 'cron_test' for 2016-05-31 10:20:32 GMT.
 *
 *     # Schedule new cron event and pass associative arguments.
 *     $ wp cron event schedule cron_test '+1 hour' --foo=1 --bar=2
 *     Success: Scheduled event with hook 'cron_test' for 2016-05-31 11:21:35 GMT.
 */
```

To break the example down piece by piece:

- "Schedule a new cron event." is the command description. The command description should be under 50 characters and written in an active, present tense.
- The options section should begin with `## OPTIONS`. Keep a blank line before and after the heading.
- Each option should be named in a manner that concisely describes its purpose.
- The example section should start with `## EXAMPLES`. Keep a blank line before and after the heading.
- Include a blank line before and after each example to help visually indicate each as a separate example.
- Each example should have 3 parts.
  - Description
    - Must start with `#` and a space.
    - Sentence should start with a capital letter.
    - Comment must end in full-stops, exclamation marks, or question marks.
    - Eg: `# Create database.`
  - Command
    - Must start with `$` and a space. Eg: `$ wp db create`
  - Sample Output
    - Keep exact output. Note spaces and indentation in the output. Exception: If output is very long, it could be truncated to show suitable section only.
    - Eg: `Success: Database created.`
- If possible, keep at least two examples for each command. One showing basic use and another advanced use. More use cases the better.

For more details on how WP-CLI understands the PHPdoc, please see the commands cookbook.

Class annotation

For classes representing a collection of subcommands, we'd encourage you to use class-level annotation to provide an introduction to the collection.

```
/**
 * Manage options.
 *
 * ## EXAMPLES
 *
 *     # Get site URL.
 *     $ wp option get siteurl
 *     http://example.com
 *
 *     # Add option.
 *     $ wp option add my_option foobar
 *     Success: Added 'my_option' option.
 *
 *     # Update option.
 *     $ wp option update my_option '{"foo": "bar"}' --format=json
 *     Success: Updated 'my_option' option.
 *
 *     # Delete option.
 *     $ wp option delete my_option
 *     Success: Deleted 'my_option' option.
 */
```

- Class could have multiple examples. Please follow standards for examples on a command doc.

Messages within command execution

```
$ wp theme activate twentysixteen
Success: Switched to 'Twenty Sixteen' theme.
```

- Message must start with a capital letter.

    - Exception: When message starts with a special key and is wrapped with quotes. Eg - `'movie' is not a registered post type.`
- If single line message, it must end with `.`.
    - Exception: There should be no trailing character in the end when colon `:` is used like this. `Invalid ID: 123`
    - Exception: Message to display in progress bar can omit trailing period. Eg - `Generating comments`
- Filenames and folder names must be wrapped with quotes (`'`).
- Roles, sidebar ID, post type key, taxonomy key must be wrapped with quotes.
- Message in the context of ongoing action could end with `...`. Eg - `Downloading from https://github.com/wp-cli/wp-cli/releases/download/v0.23.1/wp-cli-0.23.1.phar...`

Command parameter description

If there is a set of allowed values for a command parameter, we have set special format (mostly similar to YAML) for default value and available options.

```
[--format=<format>]
: Render output in a particular format.
---
default: table
options:
  - table
  - csv
  - json
  - count
  - yaml
---
```

- Section starts with `---` in the separate line next to the param description.
- In the next line keep default value with `default: value`.
- For options, start with `options:` in the following line.
- Keep value in the following pattern: 2 space, `-`, 1 space, value
- Close section with `---`.

□□□ #4
□ cocobook-sync □□□ 2026-10-28 11:01:28 UTC
□ cocobook-sync □□□ 2026-10-29 01:42:31 UTC