CLI Documentation

`ts` shorthand for TraceSentry

Commands

Enter `help` to get a list of all available commands. Or `help <command>` to get more information about a specific command.

---

1. `run`

This command starts the daemon process by specifying the path to the JAR file if it is not already running. If the daemon is already running, it will not be restarted. If the path is not provided, the command will attempt to infer the path from the environment variable `TRACE_SENTRY_DIR`.

Usage:

```
ts run <path>
```

Parameters:

- **`path`**: The full path to the daemon JAR file. Either the relative path from the current directory or the absolute path can be used. (Optional: default value is the composed path from the installation directory, as described above)

Examples:

```
ts run ./path/to/daemon.jar # relative path unix
ts run C:\\Path\\To\\Daemon.jar # absolute path windows
```

Success message:

```
Daemon started successfully.
```

Already running message:

```
Daemon is already running.
```

Error message:

```
Failed to start daemon.
```

---

2. `status`

This command checks the status of the daemon process.

Usage:

```
  ts status
```

Example:

```
  ts status
```

Output:

```
daemon is running
```

If the daemon is not running:

```
daemon is not running
```

---

3. `kill`

This command stops the daemon process if it is running.

Usage:

```
  ts kill
```

Example:

```
  ts kill
```

Success message:

```
daemon killed
```

If the daemon is not running:

```
daemon is not running
```

Other errors:

```
error killing daemon
```

---

4. `search`

This command initiates a search operation in the specified directory. The search mode can be customized, and subdirectory scanning is enabled by default unless the `--no-subdirs` flag is set. Additionally, a regex pattern can be used to filter files when the `pattern` mode is selected.

Usage:

```
ts search <path> [--mode <log|cache|full|pattern>] [--pattern <regex>] [--no-subdirs]
```

Parameters:

- **`path`** : The full path to the directory you want to search. Either the relative path from the current directory or the absolute path can be used.
- **`--mode`** : Defines the search mode:
  - `log` : Search for log files.
  - `cache` : Search for cache files.
  - `full` : Search for both log and cache files (default).
  - `pattern` : Search for files that match a custom regex pattern (requires `--pattern` parameter).
- **`--pattern`** : If the mode is set to `pattern`, this parameter defines the regular expression to match files.
- **`--no-subdirs`** : If this flag is present, the search will not include subdirectories (default is to search subdirectories).

Examples:

- Search a directory in full mode (default):

  ```
  ts search /etc/path/to/directory # absolute path unix
  ts search Path\\To\\Directory # relative path windows
  ```

- Search only for log files without scanning subdirectories:

  ```
  ts search /path/to/directory --mode log --no-subdirs
  ```

- Search using a custom regex pattern:

  ```
  ts search /path/to/directory --mode pattern --pattern ".*\.log$"
  ```

Successful search:

```
Listing 2 files in /path/to/directory...
relative/path/from/search/directory/file1.log
relative/path/from/search/directory/file2.log
```

Error message:

```
Failed to search directory.
```

Notes:

- Ensure that the path you provide is absolute or relative to the current working directory.
- If no mode is specified, the search defaults to `full` mode.
- The `pattern` mode requires the `--pattern` parameter to define the regex for matching files.

---

5. `monitor add`

This command adds a specified path for monitoring. Paths added will be tracked for changes or new log/cache files.

Usage:

```
  ts monitor add --path <path/to/monitor> [--mode <log|cache|full|pattern>] [--pattern <regex>] [--no-subdirs]
```

Parameters:

- `--path` : The full path you want to add for monitoring. This can be a relative path from the current directory or an absolute path. It might be a directory or a specific file.
- `--mode` : Defines the monitoring mode:
  - `log` : Monitor log files.
  - `cache` : Monitor cache files.
  - `full` : Monitor both log and cache files (default).
  - `pattern` : Monitor files that match a custom regex pattern (requires `--pattern` parameter).
- `--pattern` : If the mode is set to `pattern`, this parameter defines the regular expression to match files.
- `--no-subdirs` : If this flag is present, the monitoring will not include subdirectories (default is to monitor subdirectories).

Example:

- Add an absolute path to the monitoring database:

  ```
  ts monitor add --path /etc/path/to/monitor # Unix
  ```

- Add a relative path to the monitoring database:

  ```
  ts monitor add --path File\\To\\Monitor\\cache.log # Windows
  ```

Successful addition:

```
Successfully added /path/to/monitor to the monitoring database.
```

Exact path already monitored:

```
Error: /path/to/monitor is already being monitored.
```

---

6. `monitor list`

This command retrieves and displays all paths currently being monitored. Each entry shows the ID, path, and date of addition.

Usage:

```
  ts monitor list
```

Examples:

- List all monitored paths:

```
ts monitor list
```

Example output:

```
+----+-----------------------------------+-------+----------+----------+----------+
|ID  |path                               |mode   |pattern   |no-subdirs|created at|
+----+-----------------------------------+-------+----------+----------+----------+
|0001|C:\Users                           |FULL   |          |false     |2024-11-26|
+----+-----------------------------------+-------+----------+----------+----------+
|0002|C:\Users\CoolDude                  |PATTERN|.*\.cookie|true      |2024-11-26|
+----+-----------------------------------+-------+----------+----------+----------+
|0003|C:\Users\CooolDude\IdeaProjects\_bfh|PATTERN|env       |false     |2024-11-27|
+----+-----------------------------------+-------+----------+----------+----------+
```

If no paths are being monitored:

```
No paths are currently being monitored.
```

---

7. `monitor remove`

This command removes a specified path from monitoring by its unique ID. The specified path is removed from the monitoring database.

Usage:

```
  ts monitor remove --id <ID>
```

Parameters:

- **`--id`**: The unique identifier for the path to remove from monitoring. This ID can be obtained from the `monitor list` command.

Example:

- Remove a monitored path by ID:

  ```
  ts monitor remove --id 3210
  ```

Successful removal:

```
Successfully removed path with ID 3210 from the monitoring database.
```

Error message if the ID does not exist:

```
Error: No monitored path found with ID 3210.
```

---

8. `snapshots list`

This command lists all snapshots taken for a monitored path.

Usage:

```
  ts snapshots list --id <ID>
```

Parameters:

- `--id` : The unique identifier for the monitored path to get the snapshots from.

Example:

- Get the snapshots of a monitored path by ID:

```
ts snapshots list --id 1
```

Example output of successful execution:

```
+------+-------------------+--+
|Number|Timestamp          |ID|
+------+-------------------+--+
|1     |18.12.2024 22:42:01|2 |
+------+-------------------+--+
|2     |18.12.2024 22:41:15|1 |
+------+-------------------+--+
```

Example outputs of unsuccessful execution:

If the path is not monitored:

```
No monitored path found with ID [id].
```

If no snapshots are available:

```
No snapshots found for monitored path with ID [id].
```

Other:

```
Error: could not list snapshots.
```

---

9. `snapshots compare`

This command outputs the comparison of a user-defined range of snapshots taken from a given monitored path. The parameters `start` and `end` refer to the `Number` column in the output of the `snapshots list` command.

Usage:

```
  ts snapshots compare --id <ID> --start <startNumber> --end <endNumber>
```

Parameters:

- `--id` : The unique identifier for the monitored path to get the comparison from.
- `--start` : Snapshot-number (begin at 1 and snapshots are ordered from latest to oldest) to start the comparison from. (Optional: default value is 1)
- `--end` : Snapshot-number (begin at 2 and snapshots are ordered from latest to oldest) to end the comparison at. (Optional: default value is 2)

Example:

- Get the comparison of a monitored path by ID and of the last 5 taken snapshots:

```
ts snapshots compare --id 1 --start 1 --end 5
```

Example output of successful execution:

```
Listing comparison of /test from 01.12.2024 15:30:00 to 01.12.2024 17:30:00...
+-------------------+-------------------+-----------+
| Path              | Snapshot IDs      | Comparison|
+-------------------+-------------------+-----------+
| cache.txt         | 4                 | CHANGED   |
|                   | 6                 | LAST TRACK|
+-------------------+-------------------+-----------+
| log/log.txt       | 2                 | CHANGED   |
|                   | 8                 | CHANGED   |
+-------------------+-------------------+-----------+
```

Example outputs of unsuccessful execution:

If the range is too big for the existing snapshots:

```
Error: Not enough snapshots existing at the moment for this range.
```

If the index range is not valid:

```
Error: Start index needs to be smaller than the end index and not negative.
```

If it is an unexpected error:

```
Error: could not compare snapshots.
```

---

10. `inspect`

> *This command sends the contents of the file to the OpenAI API for analysis.*
> *Ensure that no sensitive, confidential, or private data is included in the file before proceeding.*
> *By using this command, you agree to the terms and conditions of the OpenAI API.*

This command allows you to inspect a specific file. The output comes directly from an OpenAI AI-model. The `OPENAI_API_KEY` environment variable containing the API key is required so that this can be queried.

Usage:

```
  ts inspect <path>
```

Parameters:

- **`path`** : The full path to the file you want to inspect. Either the relative path from the current directory or the absolute path can be used.

Examples:

- Inspect a file:

  ```
  ts inspect /path/to/apache.log
  ```

Successful inspection:

```
Intended use:
The file `apache.log` appears to be a web server access log, capturing various HTTP
requests [...] not valid IP addresses and may indicate malicious activity or probing
attempts. [...] the abnormal entries warrant further investigation.

Assessment:
Potentially harmful

Recommended to Wipe:
Clear file

Additional recommendations:
Perform a thorough analysis of the web server's security posture and monitor for
unusual traffic patterns. Validate legitimate access attempts to identify any
potential breaches or vulnerabilities.
```

File not found:

```
Error: File not found.
```

Error message:

```
Error: Failed to inspect file. Make sure the file is accessible, the connection to the
internet is working and the environment variable OPENAI_API_KEY is set.
```

---

11. `wipe`

This command wipes a given file. This either means to clear the content of the file or to delete it. Which can be specified by the `remove` flag.

Usage:

```
  ts wipe <path> [--remove]
```

Parameters:

- **`path`** : The full path to the file you want to wipe. Either the relative path from the current directory or the absolute path can be used.
- **`--remove`** : If this flag is present, the file will be deleted. Otherwise, the content will be cleared.

Examples:

- Wipe the content of a file:

  ```
  ts wipe /path/to/cool.log
  ```

- Remove a file:

  ```
  ts wipe /path/to/cool.log --remove
  ```

Successful clearing:

```
Successfully cleared file.
```

Successful removal:

```
Successfully removed file.
```

File not processable:

```
Error: File could not be processed.
```

Error message:

```
Error: Failed to wipe file.
```