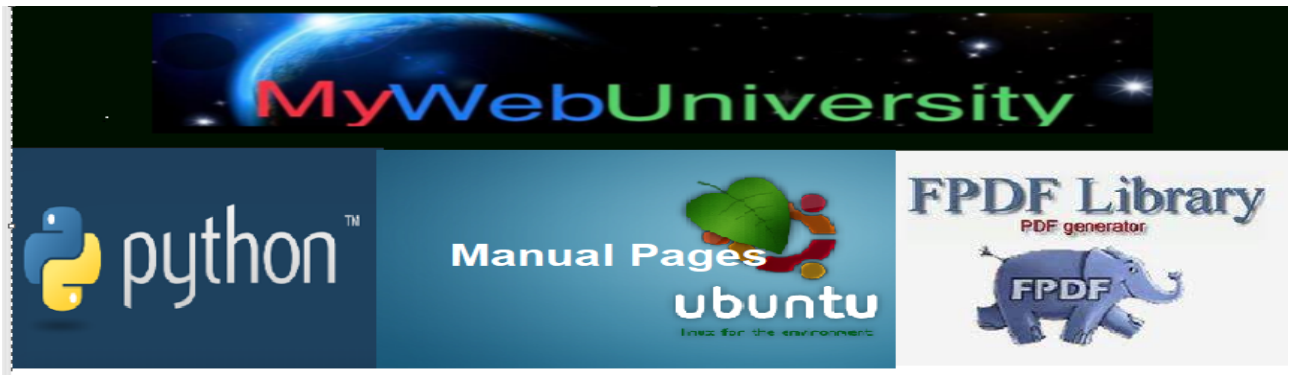

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'tap.1'

$ man tap.1

RUN.JS(1) User Commands RUN.JS(1)

NAME

run.js - Test-Anything-Protocol module for Node.js

DESCRIPTION

Usage:

tap [options] [<files>]

tap v16.3.7 - A Test-Anything-Protocol library for JavaScript

Executes all the files and interprets their output as TAP formatted test result data. If no files are specified, then tap will search for testy-looking files, and run those. (See '--test-regex' below.)

To parse TAP data from stdin, specify "-" as a filename.

Short options are parsed gnu-style, so for example '-bCRspec' would be equivalent to '--bail --no-color --reporter=spec'

If the --check-coverage or --coverage-report options are provided explicitly, and no test files are specified, then a coverage report or coverage check will be run on the data from the last test run.

Coverage is never enabled for stdin.

Much more documentation available at: https://www.node-tap.org/

Basic Options:

-R<type> --reporter=<type>

Use the specified reporter. Defaults to 'base' when colors are in use, or 'tap' when colors are disabled.

In addition to the built-in reporters provided by the

treport and tap-mocha-reporter modules, the reporter option can also specify a com?
mand-line program or a module to load via require().

Command-line programs receive the raw TAP output on
their stdin.

Modules loaded via require() must export either a
writable stream class or a React.Component subclass. Writable streams are instanti?
ated and piped into. React components are rendered using Ink, with tap={tap} as their
only property.

Available built-in reporters: classic doc dot dump json
jsonstream landing list markdown min nyan progress silent spec tap xunit

-r<arg> --reporter-arg=<arg>
Args to pass to command-line reporters. Ignored when using built-in reporters or mod?
ule reporters. Can be set multiple times

-F --save-fixture
Do not clean up fixtures created with t.testdir()

--no-save-fixture
switch off the --save-fixture flag

-b --bail
Bail out on first failure

-B --no-bail
Do not bail out on first failure (default)

--comments
Print all tap comments to process.stderr

--no-comments
switch off the --comments flag

-c --color
Use colors (Default for TTY)

-C --no-color
Do not use colors (Default for non-TTY)

-S --snapshot
Set to generate snapshot files for 't.matchSnapshot()' assertions.

--no-snapshot
switch off the --snapshot flag

-w --watch

Watch for changes in the test suite or covered program.

Runs the suite normally one time, and from then on,

re-run just the portions of the suite that are required whenever a file changes.

Opens a REPL to trigger tests and perform various

actions.

--no-watch

switch off the --watch flag

-n --changed

Only run tests for files that have changed since the last run.

This requires coverage to be enabled, because tap uses

NYC's process info tracking to monitor which file is loaded by which tests.

If no prior test run data exists, then all default

files are run, as if --changed was not specified.

--no-changed

switch off the --changed flag

-s<file> --save=<file> If <file> exists, then it should be a line- delimited

list of test files to run. If <file> is not present, then all command-line positional

arguments are run.

After the set of test files are run, any failed test

files are written back to the save file.

This way, repeated runs with -s<file> will re-run

failures until all the failures are passing, and then once again run all tests.

Its a good idea to .gitignore the file used for this

purpose, as it will churn a lot.

-O --only

Only run tests with {only: true} option, or created with t.only(...) function.

--no-only

switch off the --only flag

-g<pattern> --grep=<pattern>

Only run subtests tests matching the specified pattern.

Patterns are matched against top-level subtests in each

file. To filter tests at subsequent levels, specify this option multiple times.

To specify regular expression flags, format pattern

like a JavaScript RegExp literal. For example: '/xyz/i' for case-insensitive match?

ing.

Can be set multiple times

-i --invert

Invert the matches to --grep patterns. (Like grep -v)

-I --no-invert

switch off the --invert flag

-t<n> --timeout=<n>

Time out test files after <n> seconds. Defaults to 30, or the value of the TAP_TIME?

OUT environment variable. Setting to 0 allows tests to run forever.

When a test process calls t.setTimeout(n) on the

top-level tap object, it also updates this value for that specific process.

-T --no-timeout

Do not time out tests. Equivalent to --timeout=0.

--files=<files>

Alternative way to specify test set rather than using positional arguments. Supported

as an option so that test file arguments can be specified in .taprc and package.json

files. Can be set multiple times

Running Parallel Tests:

Tap can run multiple test files in parallel. This generally results in a speedier

test run, but can also cause problems if your test files are not designed to be inde?

pendent from one another.

To designate a set of files as ok to run in parallel, add them to a folder containing

a file named 'tap-parallel-ok'.

To designate a set of files as not ok to run in parallel, add them to a folder con?

taining a file named 'tap-parallel-not-ok'.

These folders may be nested within one another, and tap will do the right thing.

-j<n> --jobs=<n>

Run up to <n> test files in parallel.

By default, this will be set to the number of CPUs on

the system.

Set --jobs=1 to disable parallelization entirely.

-J --jobs-auto

Run test files in parallel (auto calculated)

This is the default as of v13, so this option serves

little purpose except to re-set the parallelization back to the default if an earlier

option (or config file) set it differently.

--before=<module>

A node program to be run before test files are executed.

Exiting with a non-zero status code or a signal will

fail the test run and exit the process in error.

--after=<module>

A node program to be executed after tests are finished.

This will be run even if a test in the series fails

with a bailout, but it will *not* be run if a --before script fails.

Exiting with a non-zero status code or a signal will

fail the test run and exit the process in error.

Code Coverage Options:

Tap uses the nyc module internally to provide code coverage, so there is no need to

invoke nyc yourself or depend on it directly unless you want to use it in other sce?

narios.

--100 Enforce full coverage, 100%. Sets branches, statements, functions, and lines to 100.

This is the default. To specify a lower limit (or no

limit) set --lines, --branches, --functions, or --statements to a lower number than

100, or disable coverage checking with --no-check-coverage, or disable coverage en?

tirely with --no-coverage.

-M<module> --coverage-map=<module>

Provide a path to a node module that exports a single function. That function takes a

test file as an argument, and returns an array of files to instrument with coverage

when that file is run.

This is useful in cases where a unit test should cover

a single portion of the system under test.

Return 'null' to not cover any files by this test.

Return an empty array [] to cover the set that nyc

would pull in by default. Ie, returning [] is equivalent to not using a coverage map

at all.

--no-coverage-map

Do not use a coverage map. Primarily useful for disabling a coverage-map that is set in a config file.

-cov --coverage

Capture coverage information using 'nyc' This is enabled by default.

If a COVERALLS_REPO_TOKEN environment variable is set,

and the 'coveralls' module is installed, then coverage is sent to the coveralls.io service.

Note that tap does not automatically install coveralls,

it must already be present in your project to use this feature.

-no-cov --no-coverage

Do not capture coverage information. Note that if nyc is already loaded, then the coverage info will still be captured.

--coverage-report=<type>

Output coverage information using the specified istanbul/nyc reporter type.

Default is 'text' when running on the command line, or

'text-lcov' when piping to coveralls.

If 'html' is used, then the report will be opened in a

web browser after running.

This can be run on its own at any time after a test run

that included coverage.

Built-in NYC reporters: clover cobertura html json

json-summary lcov lcovonly none teamcity text text-lcov text-summary

Can be set multiple times

--no-coverage-report

Do not output a coverage report, even if coverage information is generated.

--browser

Open a browser when an html coverage report is generated. (this is the default behav?

ior)

--no-browser

Do not open a web browser after generating an html coverage report

-pstree --show-process-tree

Enable coverage and display the tree of spawned processes.

--no-show-process-tree switch off the --show-process-tree flag

Coverage Enfocement Options:

These options enable you to specify that the test will fail if a given coverage level is not met. Setting any of the options below will trigger the --coverage and --check-coverage flags.

The most stringent is --100. You can find a list of projects running their tests like this at: https://www.node-tap.org/100

If you run tests in this way, please add your project to the list.

--check-coverage

Check whether coverage is within thresholds provided. Setting this explicitly will default --coverage to true.

This can be run on its own any time after a test run that included coverage.

--no-check-coverage

switch off the --check-coverage flag

--branches=<n>

what % of branches must be covered?

--functions=<n>

what % of functions must be covered?

--lines=<n>

what % of lines must be covered?

--statements=<n>

what % of statements must be covered?

Other Options:

-h --help

Show this helpful output

--no-help

switch off the --help flag

-v --version

Show the version of this program.

--no-version

switch off the --version flag

--test-regex=<pattern> A regular expression pattern indicating tests to run if

no positional arguments are provided.

By default, tap will search for all files ending in

.ts, .tsx, .js, .jsx, .cjs, or .mjs, in a top-level folder named test, tests, or

__tests__, or any file ending in '.spec.' or '.test.' before a supported extension,

or a top-level file named 'test.(js,jsx,...)' or 'tests.(js,jsx,...)'

Ie, the default value for this option is:

((\/|^)(tests?|__tests?__)\/.*|\.(tests?|spec)|^\/?test s?)\.([mc]js|[jt]sx?)$

Note that .jsx files will only be run when --jsx is

enabled, .ts files will only be run when --ts is enabled, and .tsx files will only be

run with both --ts and --jsx are enabled.

--test-ignore=<pattern>

When no positional arguments are provided, use the supplied regular expression pat?

tern to exclude tests that would otherwise be matched by the test-regexp.

Defaults to '$.', which intentionally matches nothing.

Note: folders named tap-snapshots, node_modules, .git,

and .hg are ALWAYS excluded from the default test file set. If you wish to run tests

in these folders, then name the test files on the command line as positional argu?

ments.

--test-arg=<arg>

Pass an argument to test files spawned by the tap command line executable. This can

be specified multiple times to pass multiple args to test scripts. Can be set multi?

ple times

--test-env=<key[=<value>]>

Pass a key=value (ie, --test-env=key=value) to set an environment variable in the

process where tests are run.

If a value is not provided, then the key is ensured to

not be set in the environment. To set a key to the empty string, use --test-env=key=

Can be set multiple times

--nyc-arg=<arg>

Pass an argument to nyc when running child processes with coverage enabled. This can

be specified multiple times to pass multiple args to nyc. Can be set multiple times

--node-arg=<arg>

Pass an argument to Node binary in all child processes. Run 'node --help' to see a list of all relevant arguments. This can be specified multiple times to pass multiple args to Node. Can be set multiple times

-gc --expose-gc

Expose the gc() function to Node.js tests

--debug

Turn on debug mode

--no-debug

switch off the --debug flag

--debug-brk

Run JavaScript tests with node --debug-brk

--harmony

Enable all Harmony flags in JavaScript tests

--strict

Run JS tests in 'use strict' mode

--flow Removes flow types

--no-flow

switch off the --flow flag

--ts Automatically load .ts and .tsx tests ts-node module. Note: you must provide ts-node as a dependency yourself, tap does not automatically bundle it. (Default: false)

--no-ts

switch off the --ts flag

--jsx Automatically load .jsx tests using tap's bundled import-jsx loader (Default: false)

--no-jsx

switch off the --jsx flag

--nyc-help

Print nyc usage banner. Useful for viewing options for --nyc-arg.

--no-nyc-help

switch off the --nyc-help flag

--nyc-version

Print version of nyc used by tap.

--no-nyc-version

switch off the --nyc-version flag

--parser-version

Print the version of tap-parser used by tap.

--no-parser-version

switch off the --parser-version flag

--versions

Print versions of tap, nyc, and tap-parser

--no-versions

switch off the --versions flag

--dump-config

Dump the config options in YAML format

--no-dump-config

switch off the --dump-config flag

--rcfile=<file>

Load any of these configurations from a YAML-formatted file at the path specified.

Defaults to .taprc in the current working directory.

Run 'tap --dump-config' to see available options and

formatting.

--libtap-settings=<module>

A module which exports an object of fields to assign onto 'libtap/settings'. These

are advanced configuration options for modifying the behavior of tap's internal run?

time.

Module path is resolved relative to the current working

directory.

Allowed fields: rmdirRecursive, rmdirRecursiveSync,

StackUtils, stackUtils, output, snapshotFile.

See libtap documentation for expected values and usage.

https://github.com/tapjs/libtap

-o<file> --output-file=<file>

Send the raw TAP output to the specified file. Reporter output will still be printed

to stdout, but the file will contain the raw TAP for later replay or analysis.

-d<dir> --output-dir=<dir>

Send the raw TAP output to the specified directory. A separate .tap file will be cre?

ated for each test file that is run. Reporter output will still be printed to stdout,

but the files will contain the raw TAP for later replay or analysis.

Files will be created to match the folder structure and

filenames of test files run, but with '.tap' appended to the filenames.

-- Stop parsing flags, and treat any additional command line arguments as filenames.

Environment Variables:

COVERALLS_REPO_TOKEN

Set to a Coveralls token to automatically send coverage information to https://cover?

alls.io, if the 'coveralls' module is installed in the project.

TAP_CHILD_ID

Test files have this value set to a numeric value when run through the test runner.

It also appears on the root tap object as `tap.childId`.

TAP_SNAPSHOT

Set to '1' to generate snapshot files for 't.matchSnapshot()' assertions.

TAP_RCFILE

A yaml formatted file which can set any of the above options. Defaults to ./.taprc

TAP_LIBTAP_SETTINGS

A path (relative to current working directory) of a file that exports fields to over?

ride the default libtap settings

TAP_TIMEOUT

Default value for --timeout option.

TAP_COLORS

Set to '1' to force color output, or '0' to prevent color output.

TAP_BAIL

Bail out on the first test failure. Used internally when '--bailout' is set.

TAP Set to '1' to force standard TAP output, and suppress any reporters. Used when run?

ning child tests so that their output is parseable by the test harness.

TAP_DIAG

Set to '1' to show diagnostics by default for passing tests. Set to '0' to NOT show

diagnostics by default for failing tests. If not one of these two values, then diag?

nostics are printed by default for failing tests, and not for passing tests.

TAP_BUFFER

Set to '1' to run subtests in buffered mode by default.

TAP_DEV_LONGSTACK

Set to '1' to include node-tap internals in stack traces. By default, these are in?
cluded only when the current working directory is the tap project itself. Note that
node internals are always excluded.

TAP_DEBUG

Set to '1' to turn on debug mode.

NODE_DEBUG

Include 'tap' to turn on debug mode.

TAP_GREP

A '\n'-delimited list of grep patterns to apply to root level test objects. (This is
an implementation detail for how the '--grep' option works.)

TAP_GREP_INVERT

Set to '1' to invert the meaning of the patterns in TAP_GREP. (Implementation detail
for how the '--invert' flag works.)

TAP_ONLY

Set to '1' to set the --only flag

TAP_TS Set to '1' to enable automatic typescript support

TAP_JSX

Set to '1' to enable automatic jsx support

Config Files:

You can create a yaml file with any of the options above. By default, the file at
./.taprc will be loaded, but the --rcfile option or TAP_RCFILE environment variable
can modify this.

Run 'tap --dump-config' for a listing of what can be set in that file. Each of the
keys corresponds to one of the options above.