Supported Command Line Switches

> *Command line switches supported by Electron.*

You can use app.commandLine.appendSwitch to append them in your app's main script before the ready event of the app module is emitted:

```
const { app } = require('electron')
app.commandLine.appendSwitch('remote-debugging-port', '8315')
app.commandLine.appendSwitch('host-rules', 'MAP * 127.0.0.1')

app.whenReady().then(() => {
  // Your code here
})
```

Electron CLI Flags

--auth-server-whitelist= `url`

A comma-separated list of servers for which integrated authentication is enabled.

For example:

```
--auth-server-whitelist='*example.com, *foobar.com, *baz'
```

then any `url` ending with `example.com`, `foobar.com`, `baz` will be considered for integrated authentication. Without `*` prefix the URL has to match exactly.

--auth-negotiate-delegate-whitelist= `url`

A comma-separated list of servers for which delegation of user credentials is required. Without `*` prefix the URL has to match exactly.

--disable-ntlm-v2

Disables NTLM v2 for posix platforms, no effect elsewhere.

--disable-http-cache

Disables the disk cache for HTTP requests.

--disable-http2

Disable HTTP/2 and SPDY/3.1 protocols.

--disable-renderer-backgrounding

Prevents Chromium from lowering the priority of invisible pages' renderer processes.

This flag is global to all renderer processes, if you only want to disable throttling in one window, you can take the hack of playing silent audio.

--disk-cache-size= `size`

Forces the maximum disk space to be used by the disk cache, in bytes.

--enable-logging[=file]

Prints Chromium's logging to stderr (or a log file).

The `ELECTRON_ENABLE_LOGGING` environment variable has the same effect as passing `--enable-logging`.

Passing `--enable-logging` will result in logs being printed on stderr. Passing `--enable-logging=file` will result in logs being saved to the file specified by `--log-file=...`, or to `electron_debug.log` in the user-data directory if `--log-file` is not specified.

> ***Note:*** *On Windows, logs from child processes cannot be sent to stderr. Logging to a file is the most reliable way to collect logs on Windows.*

See also `--log-file`, `--log-level`, `--v`, and `--vmodule`.

--force-fieldtrials= `trials`

Field trials to be forcefully enabled or disabled.

For example: `WebRTC-Audio-Red-For-Opus/Enabled/`

--host-rules= `rules`

A comma-separated list of `rules` that control how hostnames are mapped.

For example:

- `MAP * 127.0.0.1` Forces all hostnames to be mapped to 127.0.0.1
- `MAP *.google.com proxy` Forces all google.com subdomains to be resolved to "proxy".
- `MAP test.com [::1]:77` Forces "test.com" to resolve to IPv6 loopback. Will also force the port of the resulting socket address to be 77.
- `MAP * baz, EXCLUDE www.google.com` Remaps everything to "baz", except for "www.google.com".

These mappings apply to the endpoint host in a net request (the TCP connect and host resolver in a direct connection, and the `CONNECT` in an HTTP proxy connection, and the endpoint host in a `SOCKS` proxy connection).

--host-resolver-rules= `rules`

Like `--host-rules` but these `rules` only apply to the host resolver.

--ignore-certificate-errors

Ignores certificate related errors.

--ignore-connections-limit= `domains`

Ignore the connections limit for `domains` list separated by `,`.

--js-flags= `flags`

Specifies the flags passed to the Node.js engine. It has to be passed when starting Electron if you want to enable the `flags` in the main process.

```
$ electron --js-flags="--harmony_proxies --harmony_collections" your-app
```

See the Node.js documentation or run `node --help` in your terminal for a list of available flags. Additionally, run `node --v8-options` to see a list of flags that specifically refer to Node.js's V8 JavaScript engine.

--lang

Set a custom locale.

--log-file= `path`

If `--enable-logging` is specified, logs will be written to the given path. The parent directory must exist.

Setting the `ELECTRON_LOG_FILE` environment variable is equivalent to passing this flag. If both are present, the command-line switch takes precedence.

--log-net-log= `path`

Enables net log events to be saved and writes them to `path`.

--log-level= `N`

Sets the verbosity of logging when used together with `--enable-logging`. `N` should be one of Chrome's LogSeverities.

Note that two complimentary logging mechanisms in Chromium -- `LOG()` and `VLOG()` -- are controlled by different switches. `--log-level` controls `LOG()` messages, while `--v` and `--vmodule` control `VLOG()` messages. So you may want to use a combination of these three switches depending on the granularity you want and what logging calls are made by the code you're trying to watch.

See Chromium Logging source for more information on how `LOG()` and `VLOG()` interact. Loosely speaking, `VLOG()` can be thought of as sub-levels / per-module levels inside `LOG(INFO)` to control the firehose of `LOG(INFO)` data.

See also `--enable-logging`, `--log-level`, `--v`, and `--vmodule`.

--no-proxy-server

Don't use a proxy server and always make direct connections. Overrides any other proxy server flags that are passed.

--no-sandbox

Disables the Chromium sandbox. Forces renderer process and Chromium helper processes to run un-sandboxed. Should only be used for testing.

--proxy-bypass-list= `hosts`

Instructs Electron to bypass the proxy server for the given semi-colon-separated list of hosts. This flag has an effect only if used in tandem with `--proxy-server`.

For example:

```
const { app } = require('electron')
app.commandLine.appendSwitch('proxy-bypass-list',
'<local>;*.google.com;*foo.com;1.2.3.4:5678')
```

Will use the proxy server for all hosts except for local addresses (`localhost` , `127.0.0.1` etc.), `google.com` subdomains, hosts that contain the suffix `foo.com` and anything at `1.2.3.4:5678` .

--proxy-pac-url= `url`

Uses the PAC script at the specified `url` .

--proxy-server= `address:port`

Use a specified proxy server, which overrides the system setting. This switch only affects requests with HTTP protocol, including HTTPS and WebSocket requests. It is also noteworthy that not all proxy servers support HTTPS and WebSocket requests. The proxy URL does not support username and password authentication [per Chromium issue](#).

--remote-debugging-port= `port`

Enables remote debugging over HTTP on the specified `port` .

--v= `log_level`

Gives the default maximal active V-logging level; 0 is the default. Normally positive values are used for V-logging levels.

This switch only works when `--enable-logging` is also passed.

See also `--enable-logging` , `--log-level` , and `--vmodule` .

--vmodule= `pattern`

Gives the per-module maximal V-logging levels to override the value given by `--v` . E.g. `my_module=2,foo*=3` would change the logging level for all code in source files `my_module.*` and `foo*.*` .

Any pattern containing a forward or backward slash will be tested against the whole pathname and not only the module. E.g. `*/foo/bar/*=2` would change the logging level for all code in the source files under a `foo/bar` directory.

This switch only works when `--enable-logging` is also passed.

See also `--enable-logging` , `--log-level` , and `--v` .

--force_high_performance_gpu

Force using discrete GPU when there are multiple GPUs available.

--force_low_power_gpu

Force using integrated GPU when there are multiple GPUs available.

Node.js Flags

Electron supports some of the [CLI flags](#) supported by Node.js.

Note: Passing unsupported command line switches to Electron when it is not running in `ELECTRON_RUN_AS_NODE` will have no effect.

--inspect-brk[=[host:]port]

Activate inspector on host:port and break at start of user script. Default host:port is 127.0.0.1:9229.

Aliased to `--debug-brk=[host:]port`.

--inspect-port=[host:]port

Set the `host:port` to be used when the inspector is activated. Useful when activating the inspector by sending the SIGUSR1 signal. Default host is `127.0.0.1`.

Aliased to `--debug-port=[host:]port`.

--inspect[=[host:]port]

Activate inspector on `host:port`. Default is `127.0.0.1:9229`.

V8 inspector integration allows tools such as Chrome DevTools and IDEs to debug and profile Electron instances. The tools attach to Electron instances via a TCP port and communicate using the Chrome DevTools Protocol.

See the Debugging the Main Process guide for more details.

Aliased to `--debug[=[host:]port`.

--inspect-publish-uid=stderr,http

Specify ways of the inspector web socket url exposure.

By default inspector websocket url is available in stderr and under /json/list endpoint on http://host:port/json/list.