Environment Variables

Here's a list of the environment variables you can provide to your application.

BIND_IP

(*production*)

Bind the application server to a specific network interface by IP address, for example: `BIND_IP=192.168.0.2`.

See also: `PORT`.

> In development, this can be accomplished with `meteor run --port a.b.c.d:port`.

DDP_DEFAULT_CONNECTION_URL

(*develoment, production*)

There are some situations where it is valuable for the meteor client to use a different DDP server than the `ROOT_URL` server.

Setting `DDP_DEFAULT_CONNECTION_URL` when running a meteor server (development: `meteor run` or production: `node main.js`) will set the DDP server to the value in `DDP_DEFAULT_CONNECTION_URL`.

Setting `DDP_DEFAULT_CONNECTION_URL` when building (`meteor build`) will define the DDP server for `cordova` builds.

DISABLE_WEBSOCKETS

(*development, production*)

In the event that your own deployment platform does not support WebSockets, or you are confident that you will not benefit from them, setting this variable with `DISABLE_WEBSOCKETS=1` will explicitly disable WebSockets and forcibly resort to the fallback polling-mechanism, instead of trying to detect this automatically.

HTTP_FORWARDED_COUNT

(*production*)

Set this to however many number of proxies you have running before your Meteor application. For example, if have an NGINX server acting as a proxy before your Meteor application, you would set `HTTP_FORWARDED_COUNT=1`. If you have a load balancer in front of that NGINX server, the count is 2.

MAIL_URL

(*development, production*)

Use this variable to set the SMTP server for sending e-mails. Postmark, Mandrill, MailGun and SendGrid (among others) are companies who can provide this service. The `MAIL_URL` contains all of the information for connecting to the SMTP server and, like a URL, should look like `smtp://user:pass@yourservice.com:587` or `smtps://user:pass@yourservice.com:465`.

The `smtp://` form is for mail servers which support encryption via `STARTTLS` or those that do not use encryption at all and is most common for servers on port 587 and *sometimes* port 25. On the other hand, the `smtps://` form (the `s` stands for "secure") should be used if the server only supports TLS/SSL (and does not support connection upgrade with `STARTTLS`) and is most common for servers on port 465.

METEOR_DISABLE_OPTIMISTIC_CACHING

(*production*)

When running `meteor build` or `meteor deploy` you can set `METEOR_DISABLE_OPTIMISTIC_CACHING=1` to speed up your build time.

Since optimistic in-memory caching is one of the more memory-intensive parts of the build system, setting the environment variable `METEOR_DISABLE_OPTIMISTIC_CACHING=1` can help improve memory usage during meteor build, which seems to improve the total build times. This configuration is perfectly safe because the whole point of optimistic caching is to keep track of previous results for future rebuilds, but in the case of meteor `build` or `deploy` there's only ever one initial build, so the extra bookkeeping is unnecessary.

METEOR_PROFILE

(*development*)

In development, you may need to diagnose what has made builds start taking a long time. To get the callstack and times during builds, you can run `METEOR_PROFILE=1 meteor`.

METEOR_PACKAGE_DIRS

(*development, production*)

Colon-delimited list of local package directories to look in, outside your normal application structure, for example: `METEOR_PACKAGE_DIRS="/usr/local/my_packages/"`. Note that this used to be `PACKAGE_DIRS` but was changed in Meteor 1.4.2.

METEOR_SETTINGS

(*production*)

When running your bundled application in production mode, pass a string of JSON containing your settings with `METEOR_SETTINGS='{ "server_only_setting": "foo", "public": { "client_and_server_setting": "bar" } }'`.

> In development, this is accomplished with `meteor --settings [file.json]` in order to provide full-reactivity when changing settings. Those settings are simply passed as a string here. Please see the Meteor.settings documentation for further information.

METEOR_SQLITE_JOURNAL_MODE

(*development*)

The Meteor package catalog uses the `WAL` SQLite Journal Mode by default. The Journal mode for the package catalog can be modifed by setting `METEOR_SQLITE_JOURNAL_MODE`.

When running multiple concurrent meteor servers on Windows Subsystem for Linux (WSL) some meteor developers have seen issues with the package catalog. Setting the environment variable `METEOR_SQLITE_JOURNAL_MODE=TRUNCATE` can overcome the issue.

MONGO_OPLOG_URL

(*development, production*)

MongoDB server oplog URL. If you're using a replica set (which you should), construct this url like so: `MONGO_OPLOG_URL="mongodb://user:password@myserver.com:10139/local?replicaS replica set)&authSource=(your auth source)"`

MONGO_URL

(*development, production*)

MongoDB server URL. Give a fully qualified URL (or comma-separated list of URLs) like `MONGO_URL="mongodb://user:password@myserver.com:10139"`. For more information see the MongoDB docs.

PORT

(*production*)

Which port the app should listen on, for example: `PORT=3030`

See also: `BIND_IP`.

> In development, this can be accomplished with `meteor run --port <port>`.

ROOT_URL

(*development, production*)

Used to generate URLs to your application by, among others, the accounts package. Provide a full URL to your application like this: `ROOT_URL="https://www.myapp.com"`.

TOOL_NODE_FLAGS

(*development, production*)

Used to pass flags/variables to Node inside Meteor build. For example you can use this to pass a link to icu data: `TOOL_NODE_FLAGS="--icu-data-dir=node_modules/full-icu"` For full list of available flags see the Node documentation.

UNIX_SOCKET_GROUP

(*production*)

This overrides the default UNIX group of the socket file configured in `UNIX_SOCKET_PATH`. It can be set to a group name or a numerical gid.

UNIX_SOCKET_PATH

(*production*)

Configure Meteor's HTTP server to listen on a UNIX socket file path (e.g. `UNIX_SOCKET_PATH=/tmp/meteor.sock`) instead of a TCP port. This is useful when running a local reverse proxy server like Nginx to handle client HTTP requests and direct them to your Meteor application. Leveraging UNIX domain sockets for local communication on the same host avoids the Operating System overhead required by TCP based communication and can also improve security. This UNIX socket file is created when Meteor starts and removed when Meteor exits.

UNIX_SOCKET_PERMISSIONS

(*production*)

This overrides the default UNIX file permissions on the UNIX socket file configured in `UNIX_SOCKET_PATH`. For example, `UNIX_SOCKET_PERMISSIONS=660` would set read/write permissions for both the user and group.