

Mist Browser [Deprecated]

downloads 5.7M

build unknown

 build unknown

gitter join chat

code helpers 35

Mist and Ethereum Wallet have been deprecated. See the announcement and view the migration guide.

The Mist browser is the tool of choice to browse and use Dapps.

For the Mist API see MISTAPI.md.

This repository is also the Electron host for the Meteor-based wallet dapp.

Help and troubleshooting

In order to get help regarding Mist or Ethereum Wallet:

1. Please check the Mist troubleshooting guide.
2. Go to our Gitter channel to connect with the community for instant help.
3. Search for similar issues and potential help.
4. Or create a new issue and provide as much information as you can to recreate your problem.

How to contribute

Contributions via Pull Requests are welcome. You can see where to help looking for issues with the Enhancement or Bug labels. We can help guide you towards the solution.

You can also help by responding to issues. Sign up on CodeTriage and it'll send you gentle notifications with a configurable frequency. It is a nice way to help while learning.

Installation

If you want to install the app from a pre-built version on the release page, you can simply run the executable after download.

For updating, simply download the new version and copy it over the old one (keep a backup of the old one if you want to be sure).

Linux .zip installs In order to install from .zip files, please install `libgconf2-4` first:

```
1 apt-get install libgconf2-4
```

Config folder

The data folder for Mist depends on your operating system:

- Windows %APPDATA%\Mist
- macOS ~/Library/Application\ Support/Mist
- Linux ~/.config/Mist

Development

For development, a Meteor server assists with live reload and CSS injection.

Once a Mist version is released the Meteor frontend part is bundled using the `meteor-build-client` npm package to create pure static files.

Dependencies

To run mist in development you need:

- Node.js v7.x (use the preferred installation method for your OS)
- Meteor javascript app framework
- Yarn package manager

Install the latter ones via:

```
1 $ curl https://install.meteor.com/ | sh
2 $ curl -o- -L https://yarnpkg.com/install.sh | bash
```

Initialization

Now you're ready to initialize Mist for development:

```
1 $ git clone https://github.com/ethereum/mist.git
2 $ cd mist
3 $ git submodule update --init --recursive
4 $ yarn
```

Run Mist

For development we start the interface with a Meteor server for auto-reload etc.

Start the interface in a separate terminal window:

```
1 $ yarn dev:meteor
```

In the original window you can then start Mist with:

```
1 $ cd mist
2 $ yarn dev:electron
```

NOTE: Client binaries (e.g. geth) specified in clientBinaries.json will be checked during every startup and downloaded if out-of-date, binaries are stored in the config folder.

NOTE: use `--help` to display available options, e.g. `--loglevel debug` (or `trace`) for verbose output

Run the Wallet

Start the wallet app for development, *in a separate terminal window*:

```
1 $ yarn dev:meteor
```

In another terminal:

```
1 $ cd my/path/meteor-dapp-wallet/app && meteor --port 3050
```

In the original window you can then start Mist using wallet mode:

```
1 $ cd mist
2 $ yarn dev:electron --mode wallet
```

Connect your own node

This is useful if you are already running your own node or would like to connect with a private or development network.

```
1 $ yarn dev:electron --rpc path/to/geth.ipc
```

Passing options to Geth

You can pass command-line options directly to Geth by prefixing them with `--node-` in the command-line invocation:

```
1 $ yarn dev:electron --mode mist --node-rpcport 19343 --node-networkid 2
```

The `--rpc` Mist option is a special case. If you set this to an IPC socket file path then the `--ipcpath` option automatically gets set, i.e.:

```
1 $ yarn dev:electron --rpc path/to/geth.ipc
```

...is the same as doing...

```
1 $ yarn dev:electron --rpc /my/geth.ipc --node-ipcpath /path/to/geth.ipc
```

Creating a local private net

If you would like to quickly set up a local private network on your computer, run:

```
1 geth --dev
```

Look for the IPC path in the resulting geth output, then start Mist with:

```
1 $ yarn dev:electron --rpc path/to/geth.ipc
```

Deployment

Our build system relies on gulp and electron-builder.

Dependencies Cross-platform builds require additional dependencies needed by Electron Builder. Please follow their instructions for up to date dependency information.

Generate packages To generate the binaries for Mist run:

```
1 $ yarn build:mist
```

To generate the Ethereum Wallet:

```
1 $ yarn build:wallet
```

The generated binaries will be under `dist_mist/release` or `dist_wallet/release`. Starting from 0.11.0, both Ethereum Wallet and Mist ships with a meteor-dapp-wallet instance (<https://github.com/ethereum/meteor-dapp-wallet>).

Options

platform To build binaries for specific platforms (default: all available) use the following flags:

```
1 $ yarn build:mist --mac      # mac
2 $ yarn build:mist --linux    # linux
3 $ yarn build:mist --win      # windows
```

skipTasks When building a binary, you can optionally skip some tasks — generally for testing purposes.

```
1 $ yarn build:mist --mac --skipTasks=build-interface,release-dist
```

Checksums Prints the SHA-256 checksums of the distributables.

It expects installer/zip files to be in the generated folders e.g. `dist_mist/release`

```
1 $ yarn task checksums [--wallet]
```

Tasks found in gulpfile.js and gulpTasks/ Any other gulp task can be run using `yarn task`.

```
1 $ yarn task clean-dist
```

Testing

Tests run using Spectron, a webdriver.io runner built for Electron.

First make sure to build Mist with:

```
1 $ yarn build:mist
```

Then run the tests:

```
1 $ yarn test:unit:once
2 $ yarn test:e2e
```

Note: Integration tests are not yet supported on Windows.