Elderberry 1.0.4 - Documentation

By **Daniele Molinari** · Published in **Documentation** · September 23, 2023 · **9 min read**

Elderberry is a basic meeting script, with audio, video and screen sharing. It is based on mediasoup, and it uses getUserMedia underneath.

This documentation covers version 1.0.4, sold through CodeCanyon here. If you haven't purchased it yet, feel free to have a look!

Introduction

First of all, Thank you so much for purchasing this item and for being our loyal customer. *You are awesome!*

With your purchase, you are entitled to get free lifetime updates to this product.

This documentation will show you all of Elderberry's features. Please go

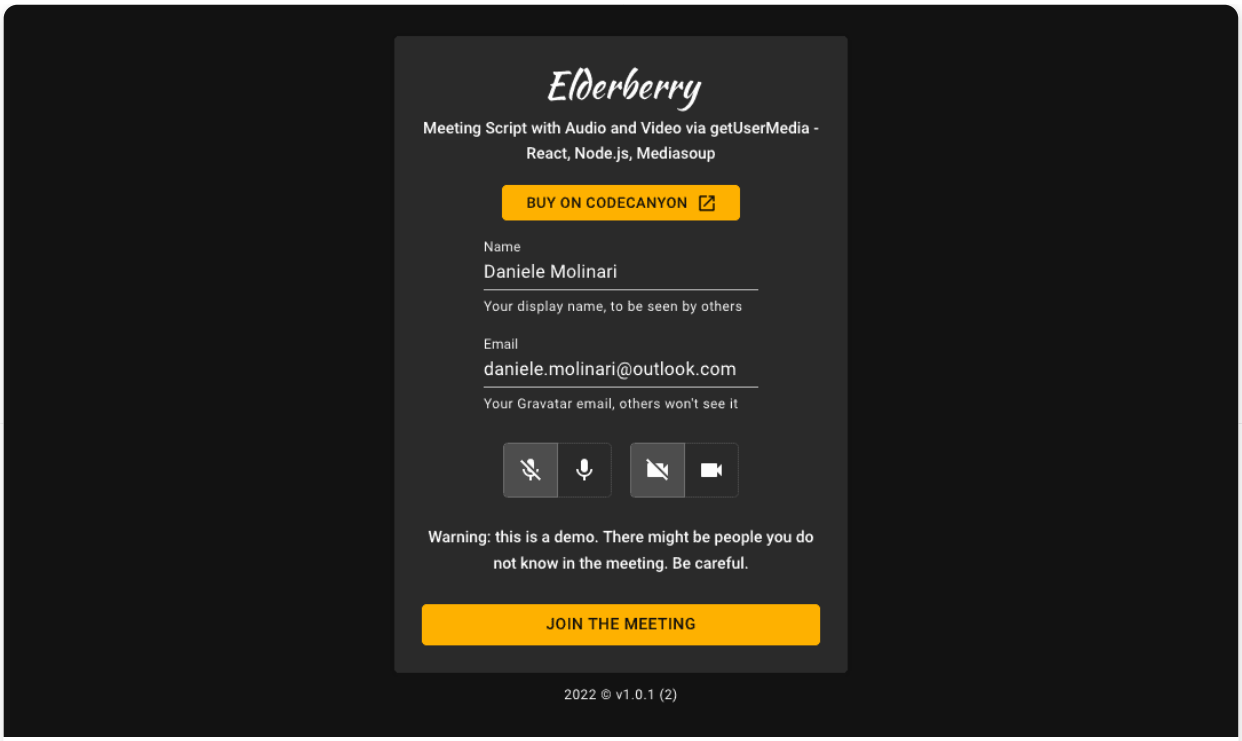
through the documentation carefully to understand how to configure your Elderberry messaging & conferencing system properly. No coding experience is required. In fact, you won't have to code at all! You will just have to edit a couple of configuration files and run a couple of scripts.

> *If you have questions or need support, please reach out using forum.honeyside.it, so that your questions will help future users with the same issue. If you don't want the trouble to go through the installation process yourself, contact us at support@honeyside.it in order to request paid installation service.*
>
> *We will also answer to support requests via email for free (after requesting proof of purchase). However, we strongly recommend to use our forum, in order to help each other. Proof of purchase is not required for joining the forum or asking questions on the forum, because we value community above bureaucracy.*

Features

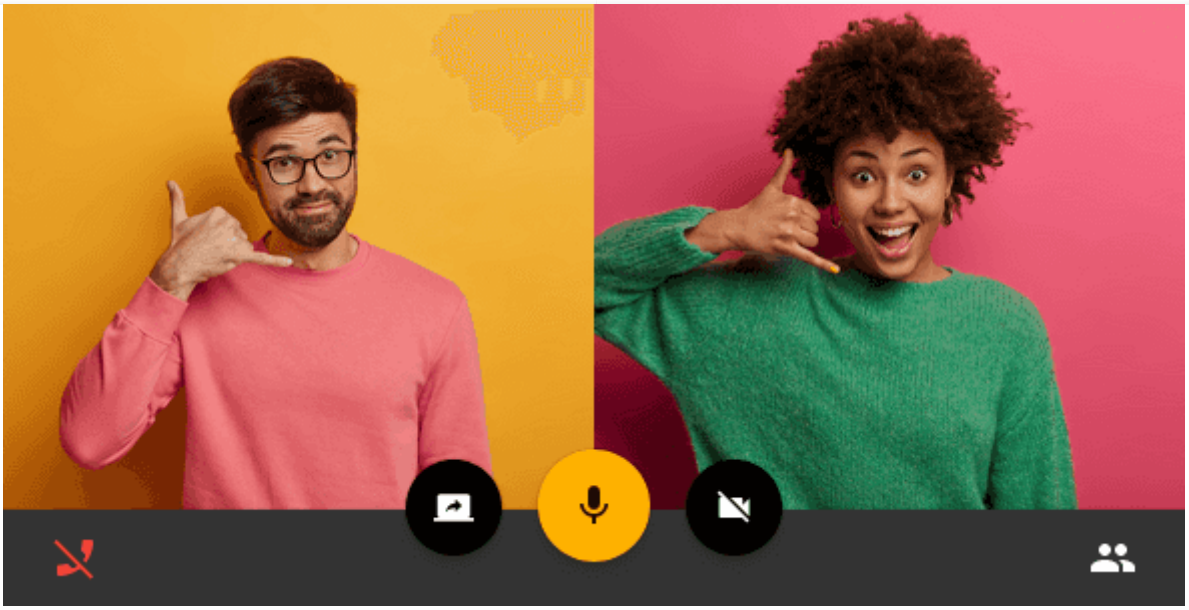
Elderberry is a basic meeting script. It is composed of two views:

- The "join" view, which prompts the user for name and email. The user can also enable or disable microphone and camera before joining the meeting (with camera preview).



The join view.

- The “meeting” view, which is the actual meeting.

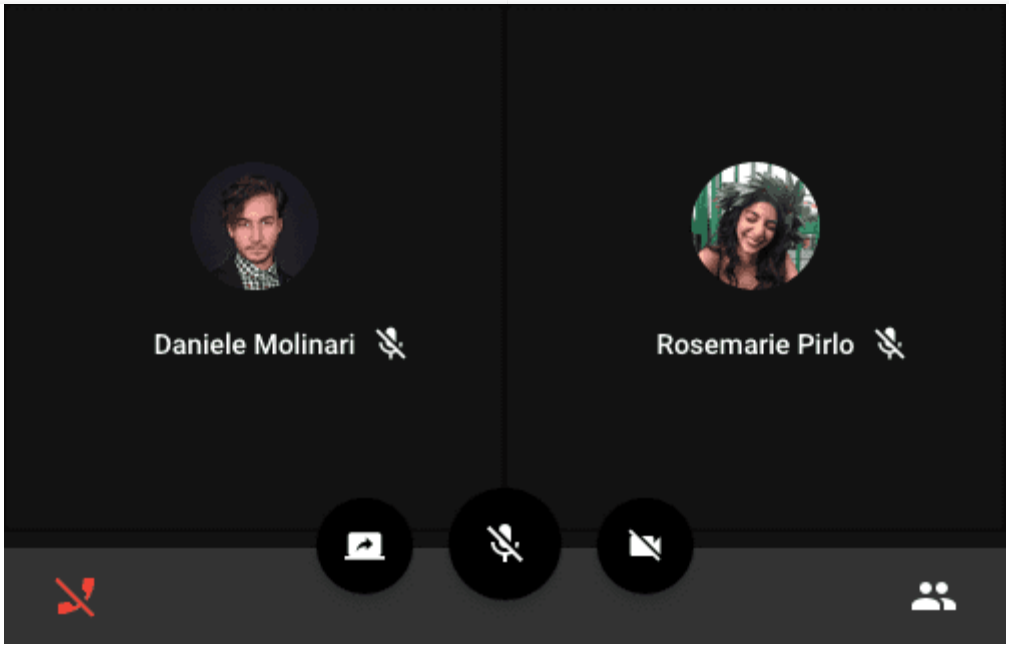


The meeting view.

The meeting page is composed of:

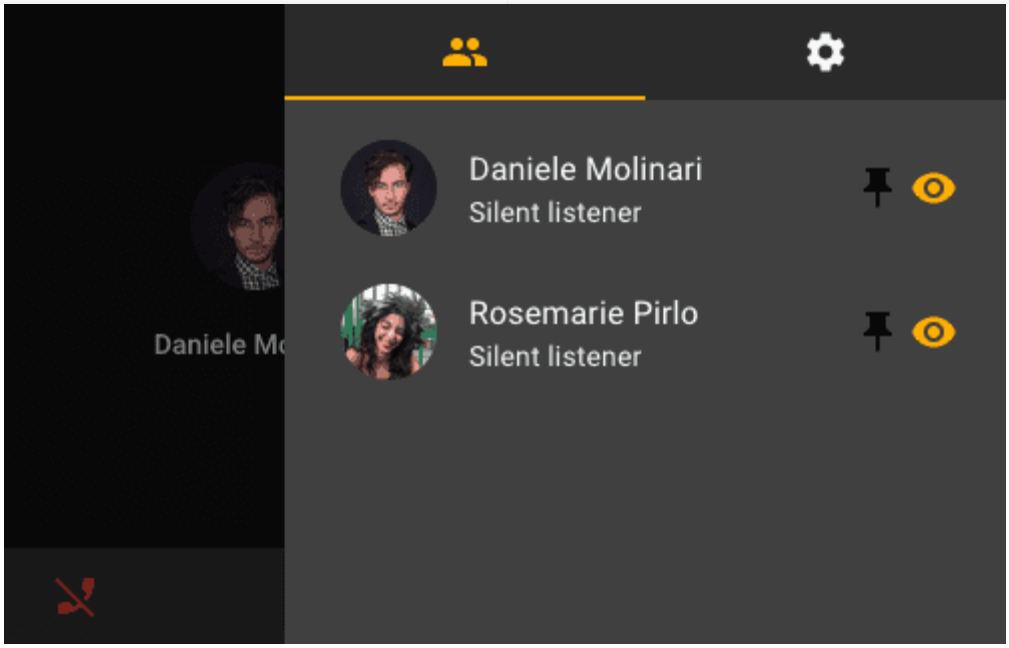
- The main layout, with an interface for each user. By default, the user interface shows the user’s name and an avatar (pulled from Gravatar via the email).

- The meeting bar at the bottom, with controls to manage camera, microphone and screen.



A user interface, with the meeting bar at the bottom.

- The meeting drawer at the right, with a list of connected peers and some useful configuration settings.



The meeting drawer.

Each user has its own interface, by default a gravatar picture (derived from

the user email) and a name. If the user activates video, the avatar disappears and the name becomes visible on hover.

Peers can be hidden (they will disappear from the main screen, and they will stay visible only in the side drawer).

Interfaces can be pinned: in this case, one peer will stand out in a bigger square on the left. This is very useful for screen sharing. In fact, Elderberry automatically enters UI Pinned mode when screen sharing starts.

How do meetings work?

In this section, I will try to de-mystify the magic world of meetings.

The first thing that you need in order to have a meeting is two or more peers. Each peer shall have a name (or an id, or both). All peers must communicate to all others their presence in the meeting, so that a proper UI can be rendered.

Each peer will need a persistent channel to send data to the server. We call this a *producer transport*.

Then, each peer will need to stream its own camera and microphone. There will be one *producer* for the camera and one *producer* for the microphone, both going through the same *producer transport*.

If there is screen sharing, we will have a third *producer* going through the same *producer transport*.

Now, each peer will need a persistent channel to receive data from the server. We call this a *consumer transport*.

Then, each peer will need to consume every stream of every other peer.

There will be one *producer* for the camera and one *producer* for the microphone of each peer, both going through the same *consumer transport*.

If there is screen sharing, we will have a third *consumer* going through the same *consumer transport*.

When the meeting ends, the *producer transport* of the peer and all *consumer transports* attached to it get closed.

That's it, meetings are complex, but they are not magic!

Project structure

The project is composed of a Node.js backend (inside the `backend` folder) and a React frontend (inside the `frontend` folder). Both backend and frontend can run on any machine supporting Node.js v18.x.x.

Then, there is a `launcher` script, for automated installation, which will only run on Ubuntu machines. If you want to use a different OS, you will have to follow the "Manual installation" steps.

Backend

The backend `config.js` file is responsible for parsing your configuration options from the `.env` file, plus it holds some hardcoded configuration options that you should not touch unless you are very experienced with Node.js.

You need to have a `.env` file in order to run the Elderberry backend. A pre-filled one will be generated for you by the automated installer, but you can also manually copy from `.env.example`. This can be done with the command `cp .env.example .env`.

The backend has a single entry point, the `index.js` file and three

subsystems under the `src` folder: HTTP Server, socket.io and Mediasoup.

The HTTP Server subsystem has two jobs:

- Serving the frontend to the user (from the location `../frontend/build`)
- Listen for new socket.io connections

The socket.io subsystem is responsible for two-way communication between frontend and backend. socket.io is the true core of Elderberry. Event names are quite self-explanatory:

- `join` is used to add a user (name, email, uuid) to the meeting
- `leave` is used to remove the current user from the meeting
- `getRouterRtpCapabilities` gets the RTP Capabilities of the backend. Default configuration is OPUS + VP8. Mess with this only if you know what you are doing.
- `createProducerTransport` creates a new producer transport (should be one per user).
- `createConsumerTransport` creates a new consumer transport (should be one per user).
- `connectProducerTransport` connects a producer transport by user id.
- `connectConsumerTransport` connects a consumer transport by user id.
- `produce` starts producing a stream through a producer transport.
- `consume` starts consuming a produce stream through a consumer transport.
- `resume` resumes producing a paused consumer transport. For a variety of reasons, video streams start paused, therefore this is called right after `consume`.
- `closeProducer` removes a producer and all connected streams (also on the consumer side).

The Mediasoup subsystem is responsible for the audio and video streaming transports used by socket.io above.

Frontend

The frontend `src/config.js` file is responsible for parsing your configuration options from the `.env` file, plus it holds some hardcoded configuration options that you should not touch unless you are very experienced with Node.js.

You need to have a `.env` file in order to run the Elderberry frontend. A pre-filled one will be generated for you by the automated installer, but you can also manually copy from `.env.example`. This can be done with the command `cp .env.example .env`.

The folder structure is a standard React + Redux setup. `index.js` is the entry point of the app. `App.js` is where routing happens. The main source code is inside the `src` folder:

- `reducers` contains the redux state of the app.
- `actions` contains async redux actions. IO actions are responsible for communication with the backend. Media actions are responsible for audio, video and screen stream management.
- `assets` contains fonts and images.
- `layouts` contains the React Router layouts. Elderberry is a very simple script, therefore there is only one layout here, the Meeting layout.
- `views` contains the pages of the app. There are two pages: Join and Meeting.
- `theme` contains global settings, such as primary and accent color (button colors, backgrounds etc).
- `common` contains components used through multiple views.
- `utils` contains functions used in a variety of places, such as the logger.

Connection to the server happens in the `App.js` file, via two actions:

`setupSocket` and `setupMedia`. This automatically creates and connects both a producer and a consumer transport for the user.

After joining the meeting, the transports start streaming via calls to the “produce” and “consume” api.

Requirements

Elderberry can run on any OS or machine that supports Node.js v18, with at least 2GB of RAM. However, in order to provide a safe environment for the unexperienced user, we recommend using Ubuntu 22.04 LTS or 20.04 LTS. We provide fully automated installation (via installation script) and full installation instructions for Ubuntu only.

For the unexperienced user, we recommend buying a cheap Ubuntu 22.04 LTS or 20.04 LTS VPS from DigitalOcean or OVH, then following the Automated Installation section. Choose a server next to your physical location (if you are in Europe, France and Germany are ok). We are not affiliated with OVH and DigitalOcean, and we do not gain commission from your VPS purchase. They just happen to work well at the time of writing.

> *Make sure to have at least 2GB of RAM, please double-check. Also, 4GB would be way faster.*
>
> *IMPORTANT: make sure to open at least ports 80, 443 and 10000-12000 through your firewall, both on UDP and TCP.*

Automated installation

Welcome to Elderberry: meeting script with audio and video via getUserMedia!

Whether experienced user or not, it is always a pleasure to run a fully automated installation. Please be aware that the automated installer will work only on Ubuntu. We recommend installing Elderberry on a clean server.

Connect to your server via SSH as root user, upload the Elderberry .zip archive and extract it in a location of your choice. cd into that location. Run `./launcher setup` and follow instructions on screen.

At the end of the installation process, Elderberry will be reachable on port 4000 and the nginx reverse proxy will be configured for ports 443 and 80.

You can stop Elderberry with `./launcher stop`. You can start Elderberry again with `./launcher start`. You can restart Elderberry with `./launcher restart`.

To upgrade to a newer version, overwrite the folder contents and run `./launcher rebuild`.

> *To elevate your shell to root, use `sudo su`.*
>
> *If you get error Command 'node' not found run command `source ~/.profile` then retry.*
>
> *IMPORTANT: make sure to open at least ports 80, 443 and 10000-12000 through your firewall, both on UDP and TCP.*

Manual installation

> *You must have experience in server management to deploy the app manually. Do not attempt this if you do not know what you are doing. Contact us at support@honeyside.it in order to request a paid custom installation service or go back to Automated Installation.*

Requirements

You will need the following software installed in order to properly run Elderberry.

1. **Build Essential**: you will need a C++ compiler and a Python 3 compiler to install certain backend dependencies. In Ubuntu, this is handled using `apt-get install build-essential python3 python3-pip`.
2. **Node.js**: you will need Node.js 18.12.1 or latest Node.js v18.
3. **Reverse Proxy**: you will need a reverse proxy for SSL. We recommend nginx.

The following software is highly recommended:

1. **Yarn**: for dependency management, in place of npm.
2. **pm2**: for process management.

Installation

1. Extract archive contents.
2. Install build essential package or equivalent for your system.
3. Install Python 3 and Python 3 pip.
4. Install Node.js v18. Using nvm (Node Version Manager) may help you manage your server later.
5. cd into backend and yarn or npm install all dependencies.
6. Copy backend/.env.example into backend/.env
7. Edit backend/.env according to your needs.
8. Run backend/index.js.
9. cd into frontend and yarn or npm install all dependencies.
10. Copy frontend/.env.example into frontend/.env
11. Edit frontend/.env according to your needs.
12. Setup nginx with reverse proxy on port 4000 (default) or to the port of your choice. In Ubuntu, this is handled using `apt-get install nginx`.
13. Setup certbot SSL for your domain (required for the meeting to work). In Ubuntu, this is handled using `apt-get install certbot python3-`

`certbot-nginx` (requires Python 3).

14. Don't forget to open port 80, 443 and 10000-12000.

Example nginx configuration (before running certbot):

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name elderberry.example.com;
    location / {
        proxy_pass http://localhost:4000;
    }
}
```

How to run certbot:

```
sudo certbot --nginx -d elderberry.example.com --non-interactive --agree-tos -m your-em
```

A thorough example of installation process is the `launcher` file. You may want to take a look at that file and edit according to your needs.

> *IMPORTANT: make sure to open at least ports 80, 443 and 10000-12000 through your firewall, both on UDP and TCP.*

Rebranding

Rebranding means changing the name from Elderberry to something else. Here are the steps to take:

1. Change the HTML `<title>` and the meta description in `frontend/public/index.html`.
2. Change the `REACT_APP_TITLE` in the frontend `.env` file.
3. Rebuild the frontend by running `yarn build` in the frontend folder.

Support

Please remember you have purchased a very affordable piece of software, and you have not paid for a full-time software development agency. Occasionally we will help with small tweaks, but these requests will be put on a lower priority due to their nature. Support is also 100% optional and we provide it for your connivence, so please be patient, polite and respectful.

Please use the Honeyside Forum for support requests. This way, your questions will be available for all Honeyside users to be read and answered. You will also receive an official answer (or confirmation) from us as soon as possible. If you still prefer to contact us directly, visit our CodeCanyon profile page or ask your question via email at support@honeyside.it. If you choose the Forum, a huge *thank you* from us in advance!

Support for our items includes:

- Responding to questions or problems regarding the item and its features
- Fixing bugs and reported issues
- Providing updates to ensure compatibility with new software versions

Item support does not include:

- Customization and installation services
- Support for third party software and plug-ins

Before seeking support, please...

- Make sure your question is a valid Elderberry Issue and not a customization request.
- Make sure you have read through the documentation and any related video guides before asking support on how to accomplish a task.
- Make sure to take a look at the Honeyside Forum first.
- Make sure you are running Elderberry in a proper Node.js environment. See

the Requirements section above for more.
- If you have customized your Elderberry installation and now have an issue, back-track to make sure you didn't make a mistake. If you have made changes and can't find the issue, please provide us with your changelog.
- Almost 80% of the time we find that the solution to people's issues can be solved with a simple "Google Search". You might want to try that before seeking support. You might be able to fix the issue yourself much quicker than we can respond to your request.
- Make sure to state the name of the item you are having issues with when requesting support via CodeCanyon.

Changelog

Once again, thank you so much for purchasing Elderberry!

```
--------------------------------------------------------------------------------
1.0.4 - September 23rd, 2023
- dependencies upgrade
- minor bug fixing

1.0.3 - May 20th, 2023
- dependencies upgrade

1.0.2 - January 20th, 2023
- dependencies upgrade

1.0.1 - December 20th, 2022
- enhanced documentation
- minor bug fixing

1.0.0 - December 11th, 2022
- first version
--------------------------------------------------------------------------------
```

Copyright and license

Elderberry is sold exclusively through CodeCanyon (Envato marketplace), where the CodeCanyon Standard Licenses apply.

You will need one Regular License for each installation of Elderberry. If you are planning to sell products that will include the Elderberry original or modified source, you will need an Extended License.

Tags #documentation #elderberry

Share 

---

0 Comments

Honeyside

VAT ID IT08510780722 - REA BA-631501 - PEC honeyside@pec.it

Previous Article

Clover 2.8.3 - Documentation

Next Article

Nginx reverse proxy setup on Ubuntu servers