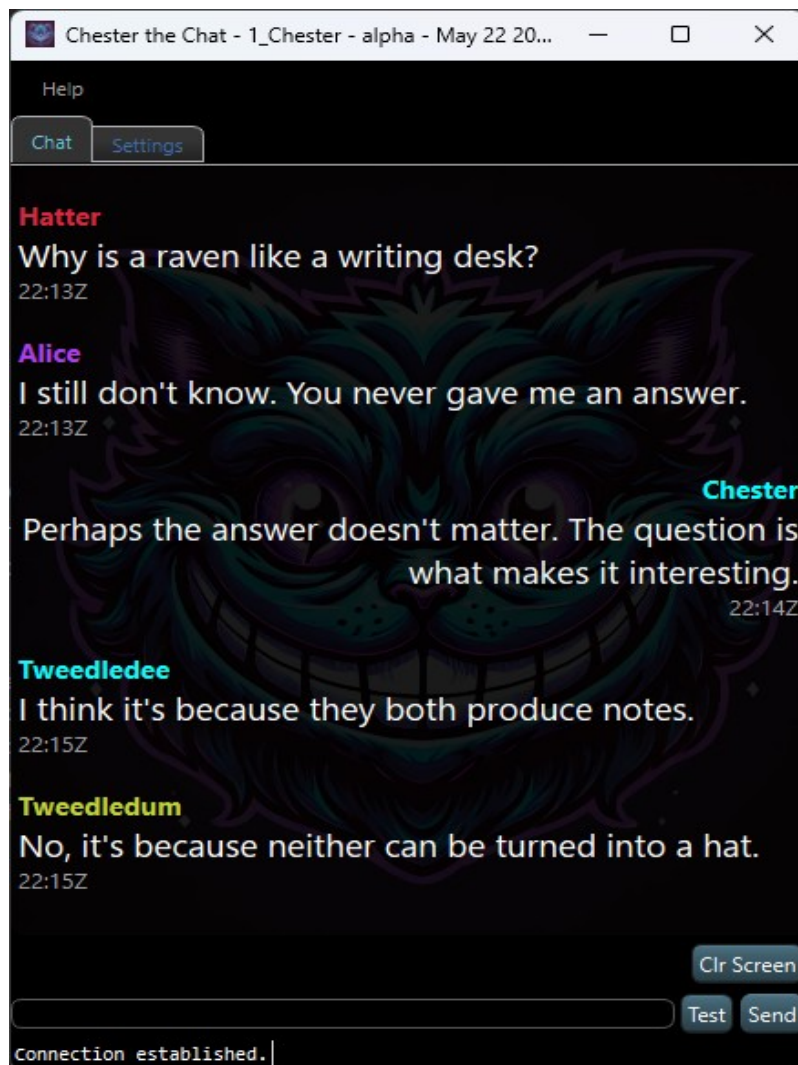


# Chester The Chat

## User Manual

Updated: 05/22/25



## Table of Contents

Overview.....	3
Building Chester From Source.....	4
TL;DR Developer Build (Qt Creator).....	4
.....	4
Installation.....	5
TL;DR Installation (for End Users).....	5
Running the Application.....	6
TL;DR Running Steps.....	6
Chat Features.....	7
TL;DR Chat Features.....	7
.....	7
Networking Settings.....	8
TL;DR Networking Settings.....	8
Appearance & Stylesheets.....	9
TL;DR Appearance Features.....	9
Multiple Instance Support.....	10
TL;DR Multi-Instance Support.....	10
Message History & Persistence.....	11
TL;DR Message History.....	11
Troubleshooting.....	12
? "I launched the app, but it says the port is already in use.".....	12
? "My custom stylesheet isn't showing up in the list.".....	12
? "I applied a style, but nothing changed visually.".....	12
? "One of my instances has weird settings or doesn't seem to start clean.".....	12
? "Messages aren't being saved between sessions.".....	12
? "Some messages don't show up in the chat window.".....	13
? "I'm getting unexpected echo messages from myself.".....	13
TL;DR Troubleshooting Quick Fixes.....	13
Screenshots.....	14
Support.....	14

## Overview

**Chester The Chat** is a lightweight, Qt-based UDP messaging tool built with developers and engineers in mind. Whether you're testing multicast behavior, running multiple chat clients on the same machine, or simply exploring how messages flow in a networked environment, Chester makes the process approachable and visual.

With support for both unicast and multicast communication, Chester enables you to simulate multiple users from a single desktop — each instance maintaining its own identity, settings, and chat history. Behind the scenes, a compact SQLite database preserves your message log across sessions, so nothing is lost when you close and reopen.

Beyond functionality, Chester also offers personality. You can customize the look and feel of your chat window using stylesheets that can be applied live. Whether you're debugging a networking issue or staging a playful Wonderland-themed demo, Chester gives you the tools — and the charm — to do it with style.

## Building Chester From Source

Chester The Chat includes a ready-to-use `.pro` file, making it easy to build directly inside **Qt Creator**.

To build the application:

1. Open `ChesterTheChat.pro` in **Qt Creator**.
2. Select your preferred build kit (e.g., MSVC or MinGW).
3. Click **Build** → **Run qmake**, then **Build Project**.
4. After building, navigate to the **Build Directory** manually. The executable is usually found under:






```
php-template  
Copy code  
<project-dir>/build-ChesterTheChat-<kit>-<mode>/release/
```

To make the application portable, run `windeployqt` on the built `.exe` file. This tool, included with Qt, copies all required Qt DLLs and plugins to the executable's folder.

For optional styling, you can place `.qss` files into a folder named `../QStyleSheets` relative to the executable. Chester will detect and allow you to apply them at runtime.

---

### TL;DR Developer Build (Qt Creator)

-  Open `ChesterTheChat.pro` in **Qt Creator**
-  Build the project with your chosen kit
-  Find the `.exe` in the `release/` subfolder of your build directory
-  Run `windeployqt path\to\ChesterTheChat.exe`
-  Drop `.qss` files into `../QStyleSheets`

## Installation




Installing Chester is quick and straightforward, thanks to its bundled Windows installer.

1. **Download the latest installer**, typically named something like:  
Chester-2025-May-19\_0603.exe  
(The date and time in the filename indicate the build version.)
2. **Double-click the installer** to launch the setup wizard.
3. Follow the prompts:
  - Choose your installation location
  - The installer includes all required Qt libraries — no additional setup is needed
4. Click **Finish** once the installation completes. You can then launch Chester from the Start Menu or desktop icon.

Chester does not require administrator privileges unless you're installing into a protected directory. All user-specific data (like settings and chat history) are stored alongside the executable.

---

### TL;DR Installation (for End Users)

-  Download a file like Chester-2025-May-19\_0603.exe
-  Run the installer and follow the steps
-  Launch Chester from the Start Menu or shortcut

## Running the Application

Starting a Chester chat session is as easy as opening the door to Wonderland.






When you launch the application, you're prompted to enter a username — this will be the name other users see when your messages appear. Each Chester instance runs independently, so you can launch multiple windows and give each a unique name (like *Alice*, *Hatter*, or *Cheshire*) to simulate a group conversation.

Before you start chatting, take a moment to configure the network settings. Choose your local interface, set the desired ports, and decide whether to use unicast or multicast. Once everything is in place, click **Connect** to join the chat.

To send a message, type into the input field at the bottom of the window and press **Enter** or click the **Send** button. Your message will appear in the chat history and be stored automatically for future sessions.

---

### TL;DR Running Steps

-  Launch `ChesterTheChat.exe`
-  Enter a username (each window = a unique user)
-  Set up local IP, ports, and multicast preferences
-  Click **Connect**
-  Type a message and hit **Enter** or click **Send**

## Chat Features

Chester The Chat brings lightweight messaging to life with a toolkit tailored for testing, play, and productivity. Messages are sent in real time over UDP, offering a fast and responsive communication experience without the overhead of server infrastructure.

Each user is assigned a consistent, hard-coded color based on their username, making it easy to track who said what — even in a multi-instance setup. Whether you're running one Chester or five, your messages will always be clearly distinguished.











Chester adapts to your environment, supporting both light and dark themes automatically. Messages include timestamps (in UTC) for precise tracking, and you can send anything from a single word to a multi-paragraph musing thanks to full multiline support.

To keep things smooth, chat history is scrollable and loaded in pages, ensuring performance doesn't degrade over time. Behind the scenes, everything is stored in an SQLite database, so your conversations persist between sessions — even across restarts.

For testing and demonstration purposes, you can inject a sample message with a single click. And when it's time to start fresh, a simple button lets you clear the entire message database instantly.

---

### TL;DR Chat Features

-  Real-time UDP messaging (unicast + multicast)
-  Consistent per-user color assignment
-  Light/dark theme support
-  Self-message echo suppression
-  Multiline message input
-  UTC timestamps for every message
-  Scrollable, paged history view
-  SQLite storage (per instance)
-  Test message injection button
-  One-click database clear





## Networking Settings

Chester gives you full control over the underlying UDP communication, making it ideal for network simulation, diagnostics, or local-only testing.

You can start by selecting the **local interface IP**. This determines which network adapter Chester will bind to for incoming traffic. If you're unsure or just testing locally, the "ANY" option allows Chester to listen on all available interfaces.

Next, configure the **local and remote UDP ports**. These define where messages are received and sent. You can even run multiple instances with different configurations to simulate full chat networks.






If you're working with multicast traffic, Chester supports it out of the box. By enabling the **multicast toggle**, messages are sent to a multicast group address instead of a specific peer. You can then have multiple instances on the same machine or network segment receive those messages simultaneously.

The **loopback** option is specific to multicast scenarios: when enabled, it allows other local instances of Chester on the same machine to receive multicast messages sent from that machine. Disabling it prevents local listeners from receiving those packets, simulating a more realistic remote-only environment.

Lastly, **TTL (Time To Live)** defines how far multicast packets can travel — a higher number allows traversal across more network segments, while a low value (e.g., 1) keeps the traffic strictly local.

---

### TL;DR Networking Settings

-  **Interface IP** — Bind to a specific adapter or use "ANY"
-  **Ports** — Define local (receive) and remote (send) UDP ports
-  **Multicast** — Enable multicast mode for group messaging
-  **Loopback** — Allow local listeners to receive your multicast packets
-  **TTL** — Set how far multicast packets can travel across the network

## Appearance & Stylesheets

Chester doesn't just function well — it looks good doing it.

The application supports live theming through Qt Style Sheets (`.qss`), allowing you to completely change the look and feel without restarting. Whether you're aiming for high contrast, a minimalist vibe, or full Wonderland flair, just drop your custom `.qss` files into the `../QStyleSheets` folder, and Chester will automatically detect and list them for selection.





Once applied, the style is instantly reflected across the UI. These themes aren't just cosmetic — they can help emphasize different chat roles, reduce eye strain, or match your development environment.

Chester also supports optional **background images**, which can be toggled on or off. This adds a layer of personality and visual depth to your chat environment — perfect for demos or just making your workspace a bit more fun.

Best of all, each instance remembers its appearance settings independently. That means you can run one window in dark mode with a background image, and another in light mode with a completely different stylesheet — a handy touch when simulating conversations between multiple characters or users.

---

### TL;DR Appearance Features

-  Load `.qss` stylesheets at runtime (no restart needed)
-  Optional **background image** support
-  Style settings are **saved per instance**
-  Drop `.qss` files into `../QStyleSheets`

## Multiple Instance Support

Chester was designed with local testing and simulation in mind, and that's where its multiple instance support really shines.





Each time you launch Chester, it assigns a **unique instance ID** to that running copy. This ID is used to isolate all related settings, logs, and message history. From the interface layout to the chat database, every instance runs in its own self-contained environment.

This makes it easy to simulate a multi-user conversation entirely on a single machine — you can run one window as *Alice*, another as *Hatter*, and a third as *Chester*, all chatting back and forth over UDP without ever leaving your desk.

The instance IDs are tracked internally in a shared file (`instance_ids.txt`) to prevent conflicts. You can safely run as many concurrent instances as your system can handle, each storing its own configuration and messages independently of the others.

---

### TL;DR Multi-Instance Support

-  Each instance gets a **unique ID**
-  Config and database files are **isolated per instance**
-  Ideal for **same-machine multi-user testing**
-  Instance tracking via `instance_ids.txt`

## Message History & Persistence

Chester doesn't just display your messages — it remembers them.

Every message you send or receive is automatically saved to a local SQLite database. This means you can close the application and return later to find your conversation exactly where you left it. There's no need to manually export or save — persistence is built in.






To keep things organized, each instance of Chester maintains its own dedicated database file. Whether you're simulating two users or twenty, each window has a clean, isolated history that matches its unique configuration.

Under the hood, the message schema is designed with versioning in mind. While the current release uses a simple format, the groundwork is already in place to support future schema upgrades without losing data — so if you decide to extend the application later with things like message reactions, file attachments, or custom metadata, you'll be ready.

For convenience and testing, there's also a one-click option to delete the current message database. This allows you to start with a clean slate whenever needed.

---

### TL;DR Message History

-  Messages are saved automatically in **SQLite**
-  Each instance uses its **own database file**
-  Chats persist across restarts
-  Schema versioning supports future upgrades
-  One-click **clear history** option

## Troubleshooting

Even in Wonderland, things don't always go according to plan. Here are some common problems you might run into while using Chester — and how to fix them.

---

### ? "I launched the app, but it says the port is already in use."

**A:** This usually means another instance of Chester or another application is already using the same UDP port. Try changing the **Local Port** and **Remote Port** settings in the UI. Make sure each instance has a unique **Local Port** unless you're intentionally testing multicast.

---

### ? "My custom stylesheet isn't showing up in the list."

**A:** Chester looks for .qss files in a folder named `./QStyleSheets` relative to the application binary. Make sure:

- The .qss file has a .qss extension
  - The folder name is exact (case-sensitive on some systems)
  - The stylesheet is valid and doesn't contain syntax errors
- 

### ? "I applied a style, but nothing changed visually."

**A:** This may happen if the .qss file doesn't actually define styles for the elements you're expecting. Try a known-working .qss first. If you edited the file while the app was open, reselect it from the dropdown to reapply.

---

### ? "One of my instances has weird settings or doesn't seem to start clean."

**A:** You may have a corrupted or orphaned instance entry. To reset all instance tracking:

1. Close all running instances of Chester
  2. Delete the file named `instance_ids.txt` in the app directory
  3. Relaunch — each new window will now start fresh
-

### ? "Messages aren't being saved between sessions."

**A:** Check whether Chester is allowed to write to its database file. This could be a **permissions issue** if the app is running in a protected folder (like **Program Files**). Make sure Chester is running from a writable directory.

Also, confirm that the database file wasn't deleted or locked by another process. Each instance uses its own file — if something blocks access, message logging will silently fail.

---

### ? "Some messages don't show up in the chat window."

**A:** If you're using multicast, make sure:

- All instances are using the **same multicast group and port**
- The **loopback** option is enabled to allow local multicast delivery
- Your firewall or OS isn't blocking multicast packets

If you're using unicast, double-check the **Remote Address** and **Remote Port** fields.






---

### ? "I'm getting unexpected echo messages from myself."

**A:** Make sure **loopback suppression** is working by confirming that:

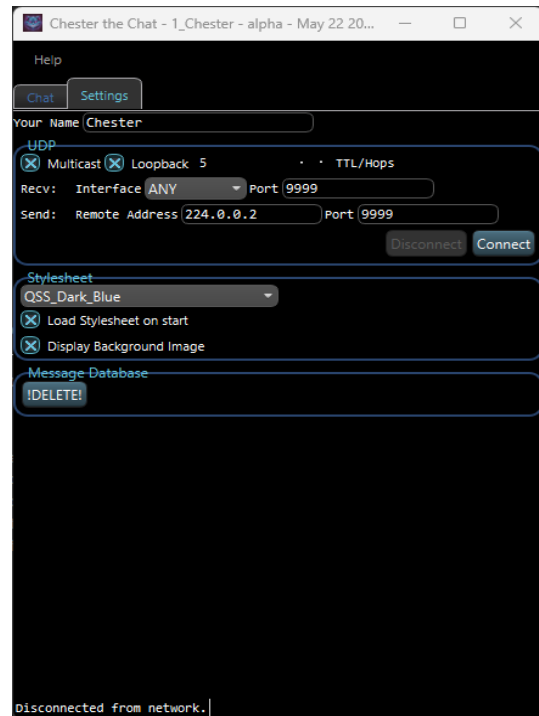
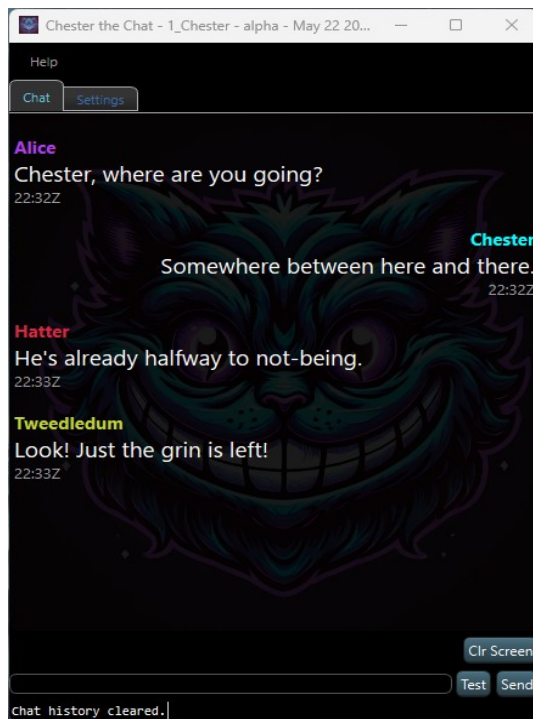
- Your messages are not being interpreted as received packets
  - The timestamps of sent messages are tracked accurately
- If needed, adjust the suppression time window or double-check that each instance has a unique **user name**.
- 

## TL;DR Troubleshooting Quick Fixes

-  **Port in use?** Try different local/remote ports.
-  **Style not applying?** Check file path and syntax.
-  **Stray instances?** Delete `instance_ids.txt`.
-  **Messages missing?** Verify DB file and folder permissions.
-  **Multicast issues?** Check loopback, TTL, and group config.



## Screenshots



## Support

Report issues or request features on GitHub. Include your platform and version information.