Mozout

Generated on 15/11/2026 00:39

KNOWLEDGE BASE · PUBLIC DOCUMENT

WebSockets and Socket.IO: what to expect on cPanel and on a VPS

Node.js & JavaScript

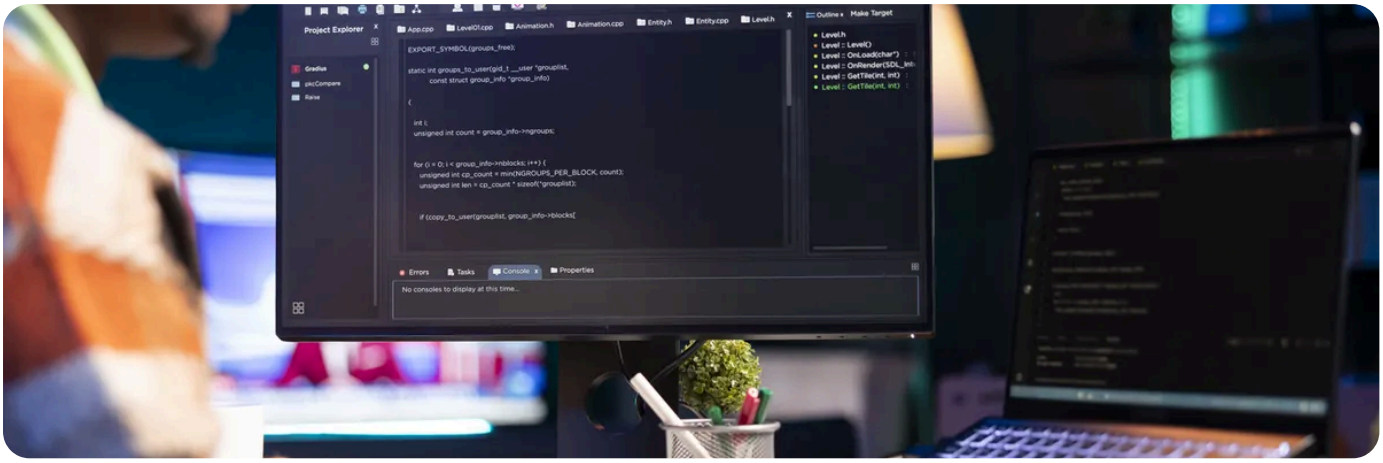
CODE	VERSION	COUNTRY · LANGUAGE
RLX-INT-KB-3402	v2026.10.18	International · English

MozOut International
RLX Holdings (Pty) Ltd. · Group RLX Holdings
Tax no. (SARS) 943 907 6234 · Registration no. (CIPC) 2025/667953/07

https://www.my.mozout.com/knowledgebase/3402

WebSockets and Socket.IO: what to expect on cPanel and on a VPS

Node.js & JavaScript Created on **October 10, 2026** Updated on **October 18, 2026** 4 min read



A chat, a dashboard that updates by itself, live notifications: all of these need a connection that stays **open** between the browser and the server, instead of the usual request and response. That connection is called a WebSocket. **The short answer:** on shared hosting, do not count on it. Open connections tie up account resources for as long as anyone is on the site, and the account has process limits. On a **VPS** it works well, with the proxy set up properly.

What differs from a normal connection

A normal HTTP request lasts milliseconds and frees the server. A WebSocket lasts minutes or hours: each connected visitor is a connection using memory and a “seat” in the process. Ten visitors are ten connections open at once, even if nobody types anything. Hence the difference between where it fits and where it does not.

	In cPanel (shared)	On a VPS
Open connections	They use limited account resources; a resident websocket server runs into the process limit.	Only the VPS’s memory and processor limit them.
Who handles the proxy	Passenger. We cannot promise long connections pass through reliably.	You, with nginx: you must pass the *upgrade* headers.
Socket.IO	It may start with *polling* (repeated requests) when the WebSocket does not get through. It works, but it is one request after another.	It uses the real WebSocket.
Recommendation	Try it with a few users; for real-time at any scale, move.	The right place.

On a VPS with nginx, step by step

1. **Make the application listen only on the server itself** (`127.0.0.1`) and keep it running, with PM2 or systemd: **keeping the app running with PM2**.

2 **In the nginx block**, add the lines that let the switch from HTTP to WebSocket through:

```
location / {
    proxy_pass http://127.0.0.1:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_read_timeout 3600s;
}
```

`proxy_read_timeout` matters: by default nginx closes a connection that has had no data for 60 seconds, and idle websockets drop. The value above is an example.

3 **In the browser, use** `wss://` (not `ws://`) when the site is HTTPS, on the same domain. An HTTPS site that opens `ws://` is blocked by the browser.

4 **Reload nginx** after validating the configuration: `sudo nginx -t` and `sudo systemctl reload nginx`.

5 **Confirm in the browser:** in the developer tools, “Network” tab, filter by “WS”. A connection with status “101” is a working WebSocket.

! **If Socket.IO “works” but is slow**

, it may be falling back to *polling* without anyone noticing: the “Network” tab shows repeated requests instead of one `101` connection. It is a sign the WebSocket is not getting through, nearly always because the *upgrade* lines are missing in the proxy.

Several instances of the application?

With more than one process behind the proxy, Socket.IO clients need to stay on the same process or use a shared adapter: the project’s documentation explains. Start with one process. On an unmanaged VPS the maintenance is yours: **how far our support goes**.

RECOMMENDED

Need real-time connections? A VPS gives you control of the proxy and the processes.

See the VPS servers →

Where to see it in your account

Open **Resource Usage** in cPanel: it shows the processes, memory and CPU of your account against its limits, and marks when a limit was reached. When the process limit is hit, new requests are refused and visitors may see error 508; when memory runs out, things fail instead. A resident websocket server keeps a process and its memory in use for as long as someone is connected, so it reaches that ceiling quickly. For real-time connections the path is a VPS, which gives you full SSH access and control of nginx, or a Cloud server. VPS1 (1 vCPU, 2 GB of RAM, 40 GB NVMe) or Cloud1 (1 vCPU, 2 GB, 25 GB) is a sensible place to start, and you can move up to VPS6 or Cloud6 (32 vCPU, 64 GB) as the connections grow.

SEE ALSO

- **Nginx as a reverse proxy in front of a container**
- **Keeping a Node.js app running on a VPS with PM2**
- **Ports and firewall on a VPS: why your app is not reachable**

Mozout

MozOut International

RLX Holdings (Pty) Ltd. · Group RLX Holdings

TAX NO. (SARS)	943 907 6234
REGISTRATION NO. (CIPC)	2025/667953/07
REGISTERED OFFICE	1111 Olive Street Saint Louis 63101, Missouri, United States
CONTACT	info@mozout.com · +27 78 823 3325 · www.mozout.com
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

Online version of this document

RLX-INT-KB-3402 · v2026.10.18 · Generated on 15/11/2026 00:39