Web logging, references and debugging

How to go from "a user saw an error" to the cause, and how to turn logging up for one user or one site without a restart. Applies to iqxWEB (the web server), WebHub and the browser client. The code table is on **Web error codes**.

What the user has

Every error toast shows a friendly message and a smaller line `CODE · ref XXXXXXXXXX`. Ask for both. The **ref** is a 10-character reference (no ambiguous characters, easy to read over the phone) that identifies one request across the browser, the web server and whichever WebHub node handled it — all three log the same ref.

Finding a ref

1. **Sentry** — Issues and Logs both accept `request_id:<ref>`. A 5xx is an Issue with the stack, tagged `code` and `request_id`; a 4xx is a log line only (user faults are not bugs). `code:<code>` shows how often something happens; filter by `site` for one install.
2. **Server stdout** — `docker logs <container>` on container sites, the service `logs/` folder on Windows sites. Lines are single-line and start with the ref in brackets. Search both WebHub nodes and the web server.
3. The line for the ref carries `detail` — the precise cause (`user-not-found`, `state expired`, `hub 404` …). It is never shown to the user for 4xx, so nothing in the toast leaks it.

Turning logging up — Owner → Security → Diagnostics → Logging

Nothing here needs a restart or a box login. Changes are stored in WebHub's database and picked up by every WebHub node and the web server within 15 seconds; each change is itself logged (and shipped to Sentry) with the login ID that made it.

- **Debug one user** — add their login ID or email with an expiry (default 4 h, max 7 days) and a reason. From then on every request they make logs at `debug` on the servers *and* ships those lines to Sentry Logs, their error responses include the cause for 5xx, and their browser switches itself into debug (its console logs and the toast detail line appear). It switches itself off at the expiry. This is the tool for "it only happens to Jane".
- **Log level** — the whole server. `debug` is very verbose on stdout but ships nothing extra to Sentry. Use it on a site you have box access to, briefly.
- **Sentry Logs on/off** — a site can be opted out of shipping log lines (error events still arrive). Off is for sites with data-residency concerns or that are burning the quota.
- **Sentry minimum level** — default `warn`. `info` for a short time shows everything the servers say about sign-ins; it counts against the 5 GB/month Team allowance, so turn it back.

"Default" in each control means the value from that server's environment file; choosing it removes the override. Users listed in the environment (`DEBUG_USERS`) show greyed and can only be removed there.

Debugging in the browser without the server

In dev tools: `localStorage["iqx:debug"] = "1"` then reload. The client logs to the console and shows the detail line in toasts *if the server sent one* — the client flag alone never changes what the server sends. Remove the key to switch off.

What ships to Sentry, and why it stays inside the quota

All log lines go to stdout, always. Only three kinds reach Sentry Logs: anything at `warn` or above; a handful of lifecycle lines (server started, cluster role changed, logging settings changed, DB reconnected); and the `debug` lines of users being debugged. The access log never ships. Each error code is limited to 60 lines per minute per node, after which one summary line says how many were suppressed — an afternoon of a provider being down costs megabytes, not gigabytes. Dev processes ship nothing.

Environment (defaults only)

`LOG_LEVEL`	`info` (debug when `DEV=true`)	yes
`DEBUG_USERS`	empty	additive — the UI list is added to it
`SENTRY_LOGS`	`true`	yes
`SENTRY_LOGS_LEVEL`	`warn`	yes
`LOG_PRETTY`	`true`	no (dev/box only)
`SENTRY_SEND_IN_DEV`	`false`	no
`ERROR_BODY_JSON`	`true`	no — `false` restores the pre-2026 string bodies for an old client

For developers

- Never `rep.status ( n ).send ( "text" )`. Use `sendError ( req, rep, "CODE", { detail, legacyBody } )` or `throw fail ( "CODE", legacyBody, { detail } )`.
- `detail` is for us, names *why*, and never contains a password, token or secret.
- A `catch` that does not know the cause goes through `toAppError ( e )` (a 5xx, Sentry issue) — it is never given a specific user-fault code. That is how "magic link expired" once became the label for six unrelated failures.
- User-fault codes (wrong password, expired link) are in EXPECTED_CODES: `info`, never Sentry unless the user is being debugged.
- `console.*` is banned by ESLint on all three codebases. Servers: `req.log` or `moduleLogger ( "name" )`. Client: `inject ( LoggerService )` or the module-level `log`.
- New code needs a new entry in `@iqx-limited/web-errors` (status + public message), a publish, a bump in all three repos, and a row on **Web error codes**.

From:
https://iqxusers.co.uk/iqxhelp/ - **IQX**

Permanent link:
https://iqxusers.co.uk/iqxhelp/doku.php?id=web:debugging

Last update: **2026/09/21 20:34**