Trace FiveM script errors and measure resource performance

A red error line, hitch warning and low client FPS describe different problems. Start with the earliest relevant error and the action that triggers it. Repeated later errors can be consequences of one failed dependency, so collecting a short startup sequence is more useful than restarting resources until the console becomes quiet.

Capture a useful reproduction

Record the resource name, installed release, FXServer artifact, time and player action. Check both the server console and the affected player's F8 console. A server-side SQL exception and a missing client texture need different fixes. Keep the first error and its stack trace together; the final line alone often omits the function or file that failed.

Before editing or updating a resource, back up its files and configuration. For persistent frameworks, export the matching database too. Use a test copy where possible. Do not hide an error by removing permission checks, suppressing every exception or replacing player tables with a fresh schema.

Classify the error before changing code

- **Missing export or dependency:** verify the referenced resource exists, started successfully and matches the expected API version.
- **Nil value:** inspect the named variable's setup and prerequisite state; a missing player or configuration value can trigger it.
- **SQL error:** distinguish connection failure from a missing table, incompatible query or schema version.

- **Client-only error:** reproduce with another client and check asset paths, case and the author's supported game build.

Change one thing at a time and repeat the same test. A resource restart can disrupt dependent scripts, so schedule maintenance for core changes rather than restarting the whole framework under connected players.

Measure on the correct side

On a test client's F8 console, `resmon true` can show per-resource client timing and memory; use `resmon false` afterwards. Its availability depends on branch and developer-mode policy. Legacy instructions using a client launch argument do not automatically apply to Enhanced, whose documentation describes a different developer-mode setup. Do not enable development access on a production server merely to bypass a denied command.

Client resmon does not measure the server's entire CPU load or database latency. Compare readings under the same player location and workload. A resource that is cheap while idle may become expensive in a busy shop or during an event.

Take a bounded profile

The Cfx profiler works in the appropriate server or client console. For a short authorised capture while reproducing the issue:

```
profiler record 500
profiler status
profiler saveJSON help-capture.json
```

Wait for recording to finish before saving. The count is frames, not seconds, so duration varies with workload. The output location belongs to the process that recorded it; a client capture is not saved in your hosted server's files. Ask support if server-side output is outside your available file access.

Review the capture for costly functions and compare before and after a single change. Share profiles privately because paths and script details can be sensitive. Keep raw timing evidence when asking an author for optimisation help; adding RAM does not automatically repair a blocking loop or an inefficient database query.

Related guides

- Inspect resource dependencies
- Distinguish database failure types

Further reading

- Cfx profiler guide
- Cfx scripting performance reference
- Cfx client console reference
- Enhanced developer-mode differences

Looking for another game? Browse game server setup and troubleshooting guides.

Revision #2
Created 2026-11-09 09:21:20 UTC by CloudExa Hosting Admin
Updated 2026-11-09 09:21:21 UTC by CloudExa Hosting Admin