Repair a Rust plugin JSON configuration without losing data

A missing comma can stop a Rust plugin from loading its settings. Deleting the plugin's data directory may make the error disappear while losing player progress. Work on the exact configuration file and preserve both the broken and last working versions.

Find the right file

For an Oxide installation, configurable plugins normally create a JSON file under `oxide/config/` matching the plugin name. For example, a configurable `ExamplePlugin.cs` normally uses `ExamplePlugin.json`. Hosted paths can vary. Keep the existing name and extension, and check `oxide/logs/` for the first error associated with that plugin.

Configuration describes behaviour; files under plugin data directories may contain balances, claims, inventories or usage records. Read the plugin author's storage notes before touching either. Do not assume a file containing JSON is disposable configuration.

Preserve a consistent rollback

1. Stop further edits and copy the faulty file to your own computer.
2. Retain the last known working configuration separately, with a date and plugin version.
3. Back up the relevant plugin data and any external database it uses. If consistent copying requires a normal server stop, schedule that first.
4. Record the setting you intended to change. Fix that one edit before trying unrelated adjustments.

Validate syntax locally

Use a local editor with JSON validation. If Python is installed on your own computer, its standard JSON formatter can check a downloaded copy:

```
python3 -m json.tool ExamplePlugin.json
```

This checks JSON syntax, not whether the plugin accepts a setting. Use straight double quotes around keys and strings, retain braces and brackets, and avoid trailing commas or comments. Do not upload configuration containing passwords, webhook URLs or database credentials to a public validation website.

An error line often marks where parsing became impossible, rather than the original mistake. Inspect the preceding line too. Compare with your retained copy instead of repeatedly replacing large sections. Ensure an editor has not silently saved the file with an extra extension.

Apply and watch the result

Upload only the corrected configuration to its original location. Apply it with the documented plugin reload, for example `oxide.reload ExamplePlugin`, from the trusted server console. Read the new console output immediately. A successful JSON check followed by a plugin error may indicate an unsupported value, outdated schema or missing dependency.

If it still fails

Restore the retained working configuration and reload again. Do not overwrite newer player data just to roll back a setting. If an upgrade changed the schema, compare the installed version with the author's current documentation and test migration in a separate environment where possible.

For support, provide the plugin name and version, the first relevant error and a minimal redacted section around the failing setting. Confirm whether restoring the prior configuration fixed it. Never include tokens, database passwords or whole player-data archives in a public post.

Related guides

- Change Stack Size Controller carefully
- Adjust Gather Manager rates

Further reading

- uMod: Oxide plugin configuration
- Python: local JSON validation

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