Guide contents

Pick the loader you actually have

Build a minimal Oxide plugin

Add a safe gameplay feature

Sources / References

Documentation: In progress **Quick start: In progress**

Offline copy from ModDocs. Source links remain clickable.

Rust server plugins

This page covers **Rust dedicated-server plugins**, not client-side replacements for the Rust game. The uMod site documents two related tracks, **Oxide** and newer **uMod Agent** workflows. They use different plugin folders and some docs explicitly say "uMod, not Oxide," so identify what your server actually runs before copying file paths or APIs. See the Rust installation guide, Oxide getting started, and uMod plugin management.

> **Quick start:** on a private test server, confirm the installed loader first. Add a minimal C# plugin that logs one line on initialization, watch the server console for that line, then unload or remove it and verify it disappears. Keep your production server and player data out of the first test.

Pick the loader you actually have

Server setup	Plugin location in source docs	Useful verification
Oxide	`oxide/plugins`	`oxide.version` command and load message
uMod Agent	`umod/plugins`	Agent plugin list or server load output

The Oxide installation page says `.cs` files in `oxide/plugins` are loaded when Oxide is installed. The uMod Agent plugin guide instead describes `umod/plugins` and its Agent commands. Do not copy a file into both directories to "make sure." Record your Rust server build, platform version and plugin version, because Rust updates can invalidate server patches.

Build a minimal Oxide plugin

For an **Oxide** server, the documented minimal plugin structure is a C# file in `Oxide.Plugins` with `[Info]` and a plugin class. A loading-only example is:

```C#
namespace Oxide.Plugins
{
    [Info("ModDocsHello", "YourName", "1.0.0")]
    [Description("Checks plugin loading on a test server")]
    class ModDocsHello : CovalencePlugin
    {
        private void Init()
        {
            Puts("ModDocsHello loaded");
        }
    }
}
```

Save this as `ModDocsHello.cs` in `oxide/plugins` on a test server. Watch its console for compilation or loading errors and the unique line. If nothing appears, verify `oxide.version` and check the server log before changing the C# body. Remove the file and confirm the plugin unloads. The getting-started reference documents the namespace, metadata and lifecycle example; this guide has not compiled it against a local Rust server.

For a newer **uMod Agent** installation, follow its project and plugin scaffolding commands rather than assuming the Oxide file will be used as-is. The Agent can create a solution, add server assembly references and create a plugin template. Test the generated skeleton unchanged before adding logic.

Add a safe gameplay feature

After loading works, choose a documented Rust hook or command. The Rust hook list includes initialization and server events; the uMod hook guide explains that return values can change or cancel default behavior. Prefer an observer that logs one event before trying a cancellation hook. A hook that fires on every tick should never write a line for every tick on a live server. If a command can be run by players, validate permission in the callback rather than assuming server-side placement makes it admin-only.

Test with only the necessary plugins, restart the server, and confirm both the positive behavior and the behavior after disabling the plugin. Back up configuration and save data before any feature that changes persistence. For release, state whether the plugin targets Oxide or uMod Agent, the Rust server build, required dependencies and how to remove it.

This guide is **unfinished**: it provides a minimal test-plugin path and the loader distinction, but not a tested gameplay command, permission model, configuration migration or production-server upgrade procedure.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- Rust installation guide
- Oxide getting started
- uMod plugin management
- Oxide installation page
- minimal plugin structure
- Rust hook list
- uMod hook guide