Install a FiveM resource and check its manifest dependencies

A downloaded resource must have its manifest at the correct directory level before FXServer can discover it. Installing a script also means meeting its dependencies, selecting the correct framework variant and preserving any existing configuration. A resource appearing in the console is only the first check; its actual game feature must work too.

Check the download before uploading

Use the author's release or authorised purchase download. A repository source archive can lack built web-interface files included in a release. Read the version requirements and configuration instructions. Before replacing an existing resource, back up its folder and configuration; if it stores player data, export the matching database as well. Keep that rollback pair together.

Extract into a resource directory beneath your server's `resources` folder. For example, `resources/[local]/example_resource/fxmanifest.lua` is sensible. An accidental extra level such as `example_resource/example_resource/fxmanifest.lua` can prevent discovery. Bracketed folders group resources; the resource name remains `example_resource`. Keep author-required names and letter case, particularly on Linux.

Read the manifest

Modern resources normally use `fxmanifest.lua`. The following is an illustrative manifest for a resource you are writing, not a replacement for a purchased script's manifest:

```
fx_version 'cerulean'
game 'gta5'
```

```
client_script 'client.lua'
server_script 'server.lua'
dependency 'example_library'
```

The declared script paths must exist. Dependencies can include another resource and runtime requirements, so read the author's notes rather than deleting a dependency line to force startup. Client and server files run in different contexts; database credentials belong in server configuration, never a client script or shared web asset. Do not modify protected escrow files.

Start and test deliberately

Install the documented dependency release first. In the FXServer console, with a maintenance window if players are online, discover newly added resources and then start the required components:

```
refresh
ensure example_library
ensure example_resource
```

Replace those example names. `refresh` rescans resources; it does not install missing libraries. `ensure` starts a stopped resource or restarts one already running, so do not use it casually on a live core framework. Observe the first error rather than repeatedly issuing the command. Missing exports often mean a dependency did not start, was renamed, or exposes a different API version.

Make the result survive restart

Add the confirmed order to the active configuration, then perform a planned restart. Join using a test account and exercise the feature, including any save-and-rejoin behaviour. A clean startup with no visible feature may point to framework configuration or permissions rather than networking. Add one new resource at a time so the last change remains easy to identify.

If installation fails, restore the previous folder and configuration during maintenance. Do not import a fresh schema into a live database or remove player tables as a troubleshooting shortcut. Share the resource name, release, first error and dependency versions with support, redacting secrets and player identifiers.

Related guides

- Order resources in server.cfg
- Trace resource errors and performance

Further reading

- Cfx resource manifest reference
- Cfx introduction to resources
- Cfx server commands

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