Guide contents

Choose a mod type

Build a first content level

Add Lua behavior

Tools, assets, and levels

Test and publish

A complete first project: a playable local scene

Sources / References

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

Offline copy from ModDocs. Source links remain clickable.

Teardown modding

Teardown includes a Mod Manager and level editor, with Lua scripting for interactive behavior. The game's own modding documentation covers the editor, mod types, assets, scripting, and publishing.

Quick start status: Complete as a written walkthrough; in-game verification remains unrecorded. **Overall documentation:** In progress. This page is the beginner route. The separate editor chapter covers scene hierarchy, tags, layers, prefabs, and spawnable assets. Source pages were checked on **2026-10-06**; this guide page was updated the same day. The official Lua API now labels itself **2.1.0** and includes multiplayer-aware `server.*` and `client.*` callbacks. The older callback discussion below describes the original single-player route; verify its context before applying it to a 2.x multiplayer mod. This guide has not been independently tested in the game.

> **Quick start:** open Teardown's Mod Manager, make a local content mod, edit one simple level, press **F5** to test it, and return to the editor with **F4**. Add Lua only after the level opens and behaves as expected.

Choose a mod type

Goal	Mod type	Key file or tool
Add a playable sandbox or mission	Content mod	`main.xml` level, edited with the built-in editor
Change behavior across other gameplay	Global mod	Lua script and the game's mod manager
Build a custom tool or weapon	Lua plus model/assets	Tool registration and input handling
Add or reshape voxel art	Voxel assets or vox-script	Editor and voxel tooling

These types have different scopes. A content mod is launched as its own experience; a global mod can affect the campaign or other content while enabled. Pick the least broad type that fits the feature. The official mod-type explanation distinguishes them and lists the required content files.

Build a first content level

From the Mod Manager's Local files pane, create a new content mod and open it in the editor. Add a small object, save, and run the level with F5. Return with F4, make one change, and test again. The official editor walkthrough explains the scene hierarchy, groups, layers, properties, and object placement.

Content mods require a `main.xml` that identifies the initial level. A multi-level mod can keep additional level files and their related assets alongside it. Keep the first level simple enough to load and exit reliably. Test destructible objects, collisions, spawn position, and whether scripts find the objects you expect. Copy a built-in mod to a local file if you want to study it; the official docs warn against trying to edit the built-in copy directly.

Add Lua behavior

Teardown's Lua scripts can react through documented callbacks such as `init`, `tick`, `update`, and `draw`. Use `init` for startup setup, `tick` for input, and draw only when

you need rendering. The scripting reference explains callback order and delta time. A minimal test is an `init` that records state and a `tick` that reads it; then replace the test with a feature you can see in the level.

Objects in the scene hierarchy can carry tags. Scripts can look up matching objects, but a search may be scoped to objects below the script unless the API's global option is used. When a script cannot find an object, inspect its tag spelling and its place in the hierarchy before changing game logic. Use the Lua API reference for function signatures and return types.

Tools, assets, and levels

A custom player tool needs its model, registration, enablement, and input behavior. The official docs include a tool example and point to a built-in Laser Gun mod for a fuller implementation. Avoid copying only a registration line and expecting a complete tool. For a level, keep voxel assets, XML, and Lua paths together in the mod folder. The `MOD/` path prefix refers to the mod's own files; `LEVEL/` is scoped to the active level in the documented structure.

Vox-scripts generate geometry at build time rather than being normal frame-by-frame gameplay scripts. Use the separate vox-script reference when creating geometry programmatically. Ordinary images and sound have their own format rules; check the official modding page before packing them.

Test and publish

1. Run the mod from the local Mod Manager and check that it works after restarting the game.
2. For a global mod, test its effect in a normal campaign or another content mod and check that disabling it restores behavior.
3. For a content mod, test loading, playability, leaving the level, and any mission progression.
4. Fill in `info.txt` and add a `preview.jpg` that meets the current size rule, then use the Mod Manager's publish action. The official publication steps describe this path.

A complete first project: a playable local scene

From **Play** → **Mods**, create a local **content mod**, open its editor, and save the generated level as `main.xml`. In the scene, place a safe player start and one visible object on a stable floor; keep the scene small enough to inspect from the spawn point. Press **F5** to enter play mode, interact with the object, then **F4** to return and make a change. The official quick start and editor tutorials cover these exact controls and the scene hierarchy.

Close and reopen the game, select the local mod from Mod Manager, and verify it still loads without the editor's unsaved state. If the player falls or starts outside the scene, inspect the spawn and collision. If an object is visible but a script cannot find it, check its tag and hierarchy scope using the Lua API. Keep the first release script-free unless behavior is needed. Add the required metadata in `info.txt` and a `preview.jpg`, then test the mod again before using the manager's publish action. A later multi-level project needs explicit level transitions; the official docs show how `StartLevel` refers to another XML file in the mod.

The playable local scene, metadata and publish path form a complete first content mod. This walkthrough is checked against the linked official docs; it has not been run in a local Teardown installation here. A custom tool, multi-step mission or voxel-asset project needs its own feature-specific workflow from the official docs and Lua API.

Sources / References

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

- modding documentation
- Lua API
- vox-script reference