Page contents

- Asset, map and location are different things
- Edit a portion of an image
- Read a map before editing it
- Add a new persistent location deliberately
- Diagnose a visual or world patch
- Sources and provenance

Offline copy of Content Patcher: images, maps and locations .

Stardew Valley Content Patcher: images, maps and locations

Documentation status: In progress. **Source checked:** 2026-10-12. **Target:** Stardew Valley 1.6 and the Content Patcher `2.9.0` format shown by its maintainer. **In-game validation:** not recorded. This is an authoring reference for visual and world changes, not a downloadable art/map asset pack. Start with pack structure and patch behavior if `Target`, `FromFile`, and load order are new to you.

Asset, map and location are different things

An **image asset** is a texture loaded by the game, such as a portrait or sprite sheet. A **map asset** describes tiles, layers, tilesheets and properties for an area. A **location** is persistent game state that uses a map and can contain players, objects, crops and NPCs. Editing `Maps/Town` changes a map asset; adding a new location needs more than a new map file. The distinction matters for saves: the map can reload, while the location's contents are saved.

Use `EditImage` for a portion of an existing image, `EditMap` for a portion or property of an existing map, and `CustomLocations` for a newly persistent area. `Load` replaces a whole asset and is easier to conflict with another pack. All paths in `FromFile` refer to files **inside your pack**; `Target` is a game content asset name, not an absolute disk path.

Edit a portion of an image

An `EditImage` patch can copy art from a pack PNG into a target asset. The key geometry is a **source rectangle** (`FromArea`) and a **destination rectangle** (`ToArea`). Their X and Y values use pixels from the upper-left corner. If omitted, `FromArea` is the whole source image and `ToArea` begins at the target's upper-left corner with the source size. For a sprite sheet, omitting `ToArea` can overwrite the wrong sprites. Confirm the source dimensions and exact target rectangle before shipping. The maintainer's EditImage reference documents overlay, transparency, extension and patch-order details.

```
{
  "Format": "2.9.0",
  "Changes": [
    {
      "Action": "EditImage",
      "Target": "Maps/springobjects",
      "FromFile": "assets/my-sprite.png",
      "FromArea": { "X": 0, "Y": 0, "Width": 16, "Height": 16 },
      "ToArea": { "X": 160, "Y": 80, "Width": 16, "Height": 16 }
    }
  ]
}
```

This is a **schema example**, not a complete mod: `assets/my-sprite.png` must be original art supplied by its author, and the `ToArea` must be checked against the actual target layout for the desired sprite. Do not infer that a sprite's visible game size equals its sheet rectangle. Export the target before and after the patch with Content Patcher's `patch export`, inspect the PNG at native pixel size, then verify it in a disposable save. Check both an unmodified install and the other texture packs you intend to support.

Read a map before editing it

The game's map reference describes the standard `Back`, `Buildings`, `Paths`, `Front`, and `AlwaysFront` layers. The tile grid uses **tile coordinates**, generally 16 × 16 source pixels per tile. Tile properties can change collision or trigger actions; a blank-looking

tile may still matter for movement. `.tmx` is a text-based Tiled map format; `.tbin` is an older binary format. Include any custom tilesheets your map refers to, and use the relative paths expected by the map and Content Patcher.

`EditMap` has three main modes of work:

Change	Relevant fields	Main risk
Paste part of a source map	`FromFile`, `FromArea`, `ToArea`, `PatchMode`	Erasing tiles on an unintended layer
Change map-level metadata/warps	`MapProperties`, `AddWarps`, `AddNpcWarps`	Broken destination or one-way travel
Change a few tiles or properties	`MapTiles`	Incorrect layer, index, collision or action

For an overlay, `FromArea` and `ToArea` are measured in **tiles** rather than image pixels. `PatchMode: "Overlay"` replaces matching source tiles; `ReplaceByLayer` (the default) replaces all tiles on source layers in the area; `Replace` replaces all layers in the area. Pick the mode by what empty source tiles should do. The EditMap action reference defines the exact merge behavior. Multiple map patches apply in order, so inspect the combined result when another pack touches the same region.

`MapTiles` is for a small, precise edit. Each entry names a `Layer` and a `Position`; adding a tile also needs its tilesheet and index. `SetProperties` changes properties of an existing tile, and `Remove` deletes the tile from that layer. A tile property without a tile on that layer has no effect. Test movement and collision, NPC routes, triggered actions and warps from both directions. Map edits can be visually correct while trapping a player or breaking an NPC schedule.

Add a new persistent location deliberately

The maintainer's custom-location reference puts `CustomLocations` at the top level of `content.json`, beside `Changes`. Each location has a globally unique `Name` beginning with the pack's manifest ID and a `FromMapFile` path inside the pack. The feature creates the persistent location; an `EditMap` warp or another access mechanism is still needed to reach it. Use the documented map/property conventions for entry and exit points.

Do **not** make a saved location conditional. Removing it can permanently discard objects and other player changes stored there. Renaming it has the same risk; the documented `MigrateLegacyNames` mechanism exists for controlled migrations from older IDs. Make a backup of the save, test an upgrade from the previous version, enter and leave the location, place an object, save, exit and reload. Test removal only on a disposable copy of the save and document what the player will lose or retain.

Diagnose a visual or world patch

1. Check SMAPI's pack-loaded line and JSON parse errors. Missing `FromFile` and incorrect folder depth are common early failures.
2. Run `patch summary` for the pack and target asset. A loaded but inactive condition is not an image/map edit.
3. Run `patch export` for the target. An exported PNG/map confirms the combined loaded asset, not that your patch was the only contributor.
4. Compare dimensions, pixel versus tile coordinates, map layers and tilesheet paths. Read any Content Patcher warnings before testing in game.
5. Visit the affected place in a disposable save, including seasons or conditions used by the patch. For warps, verify exit and return routes; for maps, verify passability and NPC movement.
6. Repeat with another likely image/map pack. Avoid forcing `Priority` unless the dependency and desired layering are explicit.

This chapter covers the main authoring model. It does not supply a tested `.tmx`, a full tile-property catalog, every image blend mode, custom location data migration, or a complete new-location project. The overall Stardew guide remains **In progress** until those and the SMAPI reference gaps are addressed and tested.

Sources and provenance

- Content Patcher EditImage, EditMap, and custom locations — maintainer references; read 2026-10-12.
- Stardew Valley map guide — game modding wiki; read 2026-10-12; layers, coordinates, formats and map properties.

- Content Patcher troubleshooting — maintainer reference; read 2026-10-12; patch diagnostics. This is original ModDocs prose and includes no redistributed game art or map files.