Page contents

- The three different inputs to a patch
- Add a player switch to the Parsnip example
- State changes and update frequency
- Preserve text with translations
- Release checklist and remaining coverage
- Sources and provenance

Offline copy of Content Patcher: tokens, config and translations .

Stardew Valley Content Patcher: tokens, config and translations

Documentation status: In progress. **Source checked:** 2026-10-12. **Target:** Content Patcher's documented `2.9.0` pack format for Stardew Valley 1.6. **In-game validation:** not recorded. Begin with the two-file Parsnip pack and the action and asset chapter. This page adds controlled variation without changing installed game files.

The three different inputs to a patch

A **placeholder** such as `{{Season}}` supplies a value inside a field that supports tokens. A **condition** in `When` decides whether a patch is active. A **config token** is an author-defined value chosen by the player. They can be combined, but each should be tested separately first. A patch using a token that is unavailable at its update point can become inactive; a successful pack load alone does not prove that a conditional patch ran.

Need	Use	Example
Choose a file from game state	Token placeholder	`assets/{{Season}}_house.png`

Need	Use	Example
Limit a patch to a game state	`When`	`"Season": "Spring, Summer"`
Let a player opt into a change	`ConfigSchema` and `When`	`"EnablePriceChange": "true"`
Translate player-facing text	`i18n` files and token	`{{i18n: some-key}}`

The maintainer's token reference groups built-in tokens by date/weather, player, relationships, world, string and number operations. It also documents config, dynamic, local and mod-provided tokens. Check a token's availability and update rate before using it in a new patch. `When` compares each condition with the token's possible values: alternatives within one value list are allowed, while all keys in the same `When` object must match.

Add a player switch to the Parsnip example

Keep the original `manifest.json` from the quick start. Replace its `content.json` with the following **complete JSON** to make the price edit optional. This modifies an existing `Data/Objects` field only when the setting is true; it does not create an item.

```
{
  "Format": "2.9.0",
  "ConfigSchema": {
    "EnablePriceChange": {
      "AllowValues": "true, false",
      "Default": "true"
    }
  },
  "Changes": [
    {
      "Action": "EditData",
      "Target": "Data/Objects",
      "When": {
        "EnablePriceChange": "true"
      },
      "Fields": {
        "24": {
```

```
            "Price": 80
          }
        }
      }
    ]
  }
```

`ConfigSchema` is beside `Changes`, not inside a patch. Content Patcher creates a `config.json` for the player; do not prepackage one with a developer's personal choices. The boolean choice can also appear in the Generic Mod Config Menu when that optional mod is installed. The patch's `When` reads the setting by its declared name. For a multi-choice option, `AllowValues` limits the accepted values and `Default` supplies the initial one. See the config schema reference for blank and multiple-value behavior.

Test both sides: launch with the default, run `patch summary` and `patch export "Data/Objects"`, then set `EnablePriceChange` to `false`, restart or reload the pack as described by Content Patcher, and inspect the exported asset again. The price should return to the other active mods' result, which might differ from the unmodded game. Re-enable it and confirm the change returns. If the false state still shows 80, inspect other patches on `Data/Objects` with `patch summary asset Data/Objects` before attributing the value to this pack.

State changes and update frequency

Built-in `Season`, `Year`, `Weather`, `Time`, `LocationName`, relationship and progression tokens have different meanings and availability. Some values change at day start; others can change within a day. A patch's `Update` setting controls when Content Patcher reevaluates token-driven fields, but a newly patched asset may still be cached or displayed by the game until the relevant content is reloaded. Choose the least frequent update that gives the behavior you need. Test a transition in both directions: for a season-dependent image, view it before and after a season change, reload the save, and inspect the exported image. For time or location behavior, read the maintainer's update-rate rules before selecting a faster update.

For complex logic, introduce a named dynamic token or patch-local token to avoid repeating the same expression in multiple places. Use a mod-provided token only after declaring and testing the dependency that provides it. `patch parse` helps inspect token expansion; `patch summary` shows whether a condition is currently

satisfied. An unavailable token is a different failure from a valid condition evaluating false.

Preserve text with translations

Put default player-facing strings in `i18n/default.json` and locale-specific strings in separate files. Content Patcher can substitute `{{i18n: key}}` where its token syntax is supported. Keep stable keys as a pack evolves; rename a key only after checking every reference. Test default fallback, at least one translated locale if provided, and missing-key behavior. The Content Patcher translation guide defines its exact token and file layout. Translation files are original text supplied by the mod author, not copies of the game's localized assets.

Release checklist and remaining coverage

Ship `manifest.json`, `content.json`, and every referenced asset and `i18n` file inside one pack folder. Exclude the generated `config.json` so updates do not overwrite a player's choices. Validate the JSON, check `patch summary` for warnings, export affected assets, test both config choices in a disposable save, and repeat with likely conflicting packs. Record exact game, SMAPI and Content Patcher versions. A source-reviewed page is **not** a claim of an in-game test.

This chapter covers the common token/config/translation workflow. The full token catalog, custom queries, randomization, custom locations, migration history and framework extension API remain to be documented as versioned references.

Sources and provenance

- Content Patcher author guide — maintainer documentation; read 2026-10-12; pack schema and feature hierarchy.
- Tokens and conditions — maintainer documentation; read 2026-10-12; token types, conditions and availability.
- Player config — maintainer documentation; read 2026-10-12; `ConfigSchema` and generated settings.

- Translations — maintainer documentation; read 2026-10-12. This page is original ModDocs prose and redistributes no game assets.