Page contents

- One package, several execution stages
- Prototype versus runtime object
- Settings and expansion features
- Verify and preserve a content release
- Reference coverage still missing
- Sources and provenance

Offline copy of Prototype stage and mod package .

Factorio: prototype stage and mod package

Documentation status: In progress. **Source checked:** 2026-10-06 against Wube's 2.1.20 API documentation. **In-game validation:** not recorded. The quick start contains a complete runtime greeting mod; this chapter explains the package and the earlier startup stages that define game content.

One package, several execution stages

Factorio loads either a mod folder or a ZIP containing a mod folder. `info.json` is the only mandatory file. It gives the internal `name`, `version`, `title`, `author`, target `factorio_version`, and optional dependencies and descriptive metadata. A loose folder may be named `name` or `name_version`; the ZIP filename must be `name_version.zip`. Check the exact rules in Mod Structure when shipping a release.

`factorio_version` is a **major compatibility branch**, such as `2.1`, and is distinct from the mod's three-part `version`. It marks one major branch, not an open ended range. A 2.0 package should be reviewed against 2.1 APIs, then explicitly changed to 2.1 if compatible. The optional `dependencies` field can require another mod, declare a conflict, or express optional/recommended load relationships. A required dependency also affects load order. If your code reads a prototype supplied by another mod,

declare the relationship rather than relying on alphabetical names. The manifest field reference gives exact prefix and version syntax.

File or directory	Stage	Purpose
`settings.lua`, `settings-updates.lua`, `settings-final-fixes.lua`	Settings	Define startup, map or player setting prototypes.
`data.lua`, `data-updates.lua`, `data-final-fixes.lua`	Prototype	Define or modify items, recipes, entities and other prototypes.
`control.lua`	Save runtime	Register event handlers, commands and interfaces.
`locale/<language>/*.cfg`	Localization	Name and describe content for players.
`migrations/`	Save upgrade	Apply named JSON or Lua migrations when versions change.

The game executes each settings round across all mods before the next round, then each prototype round across all mods. It does **not** finish all three data files from mod A before starting mod B. The data lifecycle describes this order. `data-updates.lua` and `data-final-fixes.lua` are for deliberate later changes; moving code there can hide a missing dependency without making compatibility reliable.

Prototype versus runtime object

A prototype is a startup definition. A placed entity, item stack, or player in an active save is a runtime object. During the prototype stage, `data` and `data.raw` are available, and `data:extend` registers prototype tables. The `game` object and runtime event API are unavailable. Once the prototype stage finishes, its Lua state is discarded; the resulting prototypes are constructed, but local Lua variables do not cross into `control.lua`.

Use the prototype reference hierarchy to find the exact type, its inherited fields and required values. An `ItemPrototype` needs a different shape from a recipe or assembling machine. The ItemPrototype reference is one example of the hierarchy.

Source examples in the current game files are useful, but copy only the fields your new type actually needs and verify every asset path and name. A successfully parsed Lua table is not evidence that an item is obtainable in play: it may still need a recipe, technology, entity placement rule or locale entry.

For an edit to an existing prototype, inspect `data.raw` for the type and name before assigning fields. A mod can be installed without the optional target mod, or the target can change between major Factorio branches. Guard optional integrations and test with the target present and absent. Record which file provides each prototype and why your edit belongs in that stage.

Settings and expansion features

Settings prototypes are defined in the earlier settings stage. Startup values are then available to the prototype stage. Runtime changes to a player's or map's setting do not retroactively rerun startup prototype definitions. Choose the setting type based on when a change should take effect, and describe that behavior to players. See the data lifecycle and settings documentation for the schema and timing.

The 2.1 manifest also exposes feature flags for Space Age capabilities such as quality, elevated rails and space travel. These are not generic compatibility labels; they gate specific expansion prototypes/properties and can require ownership of that expansion. Keep an ordinary base-game mod on the base path unless its defined content uses a gated feature. Check each flag against the 2.1 manifest reference before publishing.

Verify and preserve a content release

Test the unzipped folder first, then remove it and test the ZIP that users will install. Enable the mod, restart for startup changes, read the first prototype error, and open a new test map to confirm the item/recipe/entity is visible, localized and obtainable. Also load an existing save to see whether prototype renames or removals need a migration. Keep a copy of the old save and mod ZIP for that test. Store `info.json`, Lua source, locale, original art and release notes together; link the exact API branch and record the game build used for testing.

Reference coverage still missing

The official prototype reference covers many families: items, recipes, technologies, fluids, equipment, entities, graphics, sounds, logistics, GUI styles and expansion-specific types. This chapter defines the **navigation model** for that reference; it does not enumerate the fields of every family. Each substantial family needs a dedicated versioned ModDocs reference and tested example before Factorio can be marked comprehensive.

Sources and provenance

- Wube Mod Structure, Data lifecycle, prototype hierarchy and ItemPrototype — official 2.1.20 API; checked 2026-10-06.
- Factorio API documentation license — Wube's documentation license permits redistribution with its notice. This ModDocs page uses original prose and does not bundle Wube's API files or game assets.