Config Language in Drupal

Hidden inconsistencies and Open Source solutions

Slides by Gábor Hojtsy - August 6, 2026.

Drupal multilingual is market leading

Drupal Core's key goal is to provide a rock solid data layer

Four modules with distinct goals

- **Content translation**
  - UI for content entities (nodes, user profiles, etc)
  - Tracks extra metadata
  - Language selector on every node (if configured)
- **Config translation**
  - UI for config such as field labels, menu names, view titles etc. translation
  - Translate into any installed language
- **Language**
  - Manages languages as config entities
  - Config overrides (storage for config translation)
  - Language negotiation (URL prefixes, etc.)
- **Interface translation (locale)**
  - Translates interface strings
  - Syncs with localize.drupal.org translations
  - Coordinates with config translation storage

Every piece of content and config has language tracking on it.

For *content*, language is visible and prominent (listings, edit forms, translate tab).

Language of config is treated internal

- All config listings lack language information and most config forms lack display or editability of language of the config.
- Three core config entity types expose a language selector directly: `menu`, `date_format`, `taxonomy_vocabulary`
- Views has one — but buried in the edit-details subform.
- Content types, image styles, text formats, roles, and most contrib config has no indicator, no selector and no way to change it after creation.

Why the source language matters

- The base configuration UI (such as editing a content type or webform) uses that source language to edit the config.
- Interface translation integration makes assumptions based on that language.
- Historic idea was that sites may want config in dedicated language in some cases (eg. a view only visible on the Spanish part of the site only in Spanish). But the UI did not follow to expose this in listing or editing and Drupal's behavior ended up in confusing mixed language setups as a result.

"Bad" `langcode` on config could lead to

- Not possible to translate in **config translation UI** because different grouped configs use different language or the language used is not the actual language the text is in.
- Not possible to manage **config language overrides** for them, especially not for locale managed config as it has the wrong assumption about the config's language then.
- Not possible to use **Drupal Canvas** at all as it relies on strict consistency.

A site can silently accumulate config in a mixture of languages.

Let's start from the beginning

No language or translation modules enabled. An English-only site.

All happy in manually created config land

Any config manually created will be English. This is great, but...

Module installs don't validate anything

Drupal installs config exactly as shipped in module YAML files.
Nothing rewrites it. Nothing validates it. Nothing shows it.

Module config shipped langcode	What you get
`en`	🟢 `en`
(missing key)	🟢 `en`
`xx` (valid on the site)	🟡 `xx` kept
`zz` (unknown on the site)	🔴 `zz` kept
(empty string)	🔴 empty kept

🔴 Invalid and empty langcodes silently persist — the module installer never validates them.

This is mitigated by `langcode` typically being `en` or missing in contributed modules at least.

Recipes attempt to validate, keep anyway

Recipes import their `config/` YAML files **verbatim**.
Whatever `langcode` is in the file ends up in the database.
There is no mechanism for the site to say "use our language."

Shipped `langcode` in recipe config	Result (not `FullyValidatable`)	Result (`FullyValidatable`)
`en`	🔴 `en`	same
(missing key)	🟢 site default	same
`xx` (valid on the site)	🟡 `xx` kept	same
`zz` (unknown on the site)	🔴 `zz` kept	same but also fires an exception
(empty)	🔴 empty kept	same but also fires exception

Validation is being introduced with recipes, module install does not do it (yet).

Now enable Language module

Language module: new complexity

With Language module enabled, config can have its `langcode` set by:

Form	Has selector	Langcode source
Most config entities (such as content type) when created	🟢 No	🔴 Request language
Menu add/edit Date format add/edit Vocabulary add/edit	🟡 Yes	🟡 Request language default
Views add (wizard)	🟢 No	🟢 Site default language
Views edit-details	🟡 Yes	🟡 Edits already saved value

On a multilingual site an admin visiting `/de/admin/…` silently creates config with `langcode: de`

No warning. No indicator.
No way to see or fix this through the UI.

What about the site default language?

Common belief: *"I set the site default to German, so my config will be in German."*

Without Locale module: the site default does nothing for config.

- Does not rewrite any existing config langcodes
- Does not affect most newly created config: that follows the request language (other than Views!)
- Does not affect module/theme installs: those install as shipped (same as without Language module)

The site default and the config langcodes diverge further.

Config actions add more complication

- Language module being enabled changes nothing about config in recipes. Config files in a recipe's `config/` directory are still imported verbatim.
- But config actions will follow the request language.
- If the request language when the recipe runs is `de`, config actions create entities with `langcode: de`. While the recipe imported config may remain empty or `en`.
- Same recipe could unintentionally install config in multiple languages. (Such as with Project Browser).

Enable Interface Translation

Now config language gets rewritten — but has many gaps

Locale's rewrite mechanism

When locale is enabled and a **module** is installed via the UI:

1. Locale's batch (`updateDefaultConfigLangcodes()`) runs after install
2. Finds all config tracked in locale's **install storage** (from `config/install/` of all installed modules)
3. For each one, if the active `langcode` is `en` **or empty/missing** → rewrite to site default
4. Then merge available translations for the new language

This is the only time locale rewrites config langcodes.

Rewrite on install is full of holes though

Config shipped with module	Does locale rewrite it?
`en`	🟢 Yes — rewritten to site default
(missing)	🟢 Yes — rewritten to site default
(empty)	🟢 Yes — rewritten to site default
`xx` (valid on the site)	🟡 No — locale only rewrites `en`
`zz` (unknown on site)	🔴 No — silently persisted as invalid

Additionally UI-created config is **permanently invisible** to locale's rewrite batch. An admin who creates a content type while site default is `en` and later changes to `de` will find that type's langcode stays `en` forever.

Locale does not rewrite non-`en` codes

- Once a config object has been rewritten away from `en` (or empty/missing), locale never touches it again.
- If the site default is changed to `fr`, later newly installed module config is rewritten to `fr`. Old config (previously installed and manual) stays as-is.
- Later if switching back to English site default, locale will also not do a rewrite then (or afterwards).

Non-translatable config at install time

🚀 `Fixed in Drupal 11.5/12.0` → Core issue #3600904

- **If the installed config did not have translatable elements at install time, locale will skip it and not assign a language**
- **But such config can easily get translatable pieces later from plugins or third party settings**

Changing site default with locale: still does nothing on its own

With locale enabled, changing the site default language does not immediately rewrite any active config.

The rewrite only happens as a side effect of installing a module. If no module is installed after the site default changes, config langcodes are completely unaffected.

This is of course not intentional.

Rewrite does not happen on theme install

🚀 `Fixed in Drupal 11.5/12.0` → Core issue #3612163

Standard theme install goes through a link, not a form. The link happily redirects when done and does not execute the batch that locale sets to rewrite the installed config.

Even `en` config is not rewritten to the site default.

All langcodes stay exactly as shipped, despite locale being enabled and the site default being set.

Only experimental themes had this working as they have a confirm form.

Wait a minute....

We did not even enable Config Translation

Config Translation is just a UI module

You can use alternate UIs or just the API

Config Translation doesn't need Locale — but requires it anyway

Config Translation module requires Locale as a dependency, but the only thing it actually uses from Locale is `LocaleConfigManager::hasTranslation()` — a one-line function that calls the configurable language manager, which itself has no dependency on Locale at all.

This means **you cannot use Config Translation without also enabling Interface Translation**, even on headless sites that may not need UI translation at all.

Core issue #3614335 — not yet resolved.

Summary of problems identified

- UI forms and config actions follow the request language
  - Except Views which save as site default + 3 others have selectors
- Site default change
  - Does not change config immediately and
  - May never change most existing config (if not still `en`)
- Locale only rewrites `en` and empty/missing, never other known or unknown languages
- Locale never rewrites UI-created config
- Locale skips config with no translatable elements 🚀 `Fixed in Drupal 11.5/12.0`
- Theme install: locale batch never runs 🚀 `Fixed in Drupal 11.5/12.0`
- Recipes import langcodes as-is
- Empty/unknown langcode in `FullyValidatable` entities are saved AND fail
- Config Translation module requires Locale even though it doesn't need it

All of this accumulates quietly. Most config have no UI exposing it.

Your site may be full of multilingual config you did not want.

Language Audit:
making it visible

A new read-only diagnostic module that surfaces this information at `admin/config/regional/audit` without modifying anything.

Also covers content language in case you need that too.

⚠ node_type	hu: 1 en: 4	0	-
🔴 page_region	en: 2	0	-
🔴 pathauto_pattern	en: 4	0	-
🔴 pattern	en: 1	0	-
🔴 responsive_image_style	en: 29	0	-
🔴 search_api_index	en: 1	0	-
🔴 search_api_server	en: 1	0	-
✅ simple_sitemap	hu: 2	4	-
✅ simple_sitemap_type	hu: 2	4	-
✅ symfony_mailer_lite_transport	hu: 1	1	-
🔴 taxonomy_vocabulary	en: 1	0	-
⚠ user_role	hu: 3 en: 1	2	-
⚠ view	hu: 15 en: 6	653	-
🔴 webform	en: 2	0	-
🔴 workflow	en: 1	0	-
TOTAL	hu: 188 en: 422	828	-

Simple Configs

Config name	Alapértelmezés szerinti nyelv	Total translatable strings	Fordítások
🔴 autosave_form.messages	en: 1	0	-
✅ autosave_form.settings	hu: 1	1	-
✅ captcha.settings	hu: 1	3	-
✅ easy_breadcrumb.settings	hu: 1	10	-
✅ klaro.settings	hu: 1	2	-
✅ klaro.texts	hu: 1	27	-
✅ menu_link_attributes.config	hu: 1	5	-

How to use it to debug your environment

1. Install Language Audit and visit `admin/config/regional/audit` to review your current status
2. Drill into any entity type to see which specific entities are in unexpected languages

See how an action causes problems

1. Take a config language snapshot using the form on the page
2. Install a module, apply a recipe, create some config or change the site default language
3. Return to the audit page — the diff shows exactly which config langcodes changed or were newly installed

The module can be used to visualize all the problems documented in these slides.

Audit. Experiment.

```
composer require drupal/language_audit
```

```
admin/config/regional/audit
```

But that will not fix it, just let you know about your problems...

Enter: Config Language Lock

One explicit language. Enforced everywhere.

A very simple UI with copious tests

Install the module. Until a lock is configured, it has no side effects.

Configure the **lock language** at `admin/config/regional/config-language-lock`, then locale rewriting is turned off and these six enforcement mechanisms are added:

Point	Mechanism
Any config entity save	`hook_entity_presave` overwrites `langcode`
Module install	`hook_modules_installed` queues lock rewrite batch
Theme install	`hook_themes_installed` queues lock rewrite batch
Recipe apply	`RecipeAppliedEvent` subscriber normalizes recipe config
Config language selectors	Removed so the locked language is universally used
Lock settings form	Batch rewrites all existing active configuration

Presave enforcement for config entities

`hook_entity_presave` fires on **every config entity save** — each hook decides itself whether to respect `isSyncing()`.

- UI form save and config actions
- Recipe's own `config/` entities
- Regular module / theme install
- Config sync (`drush cim`)
- Any other programmatic save

Language selectors hidden on forms

`hook_form_alter` hides language selectors so admins don't see a control that the lock will override anyway.

For menu, date format, vocabulary add / edit and **contrib entity forms using a standard `$form['langcode']` with an `EntityFormInterface`** have their language selector hidden.

Views edit-details has its `$form['details']['langcode']` hidden with an explicit form ID check.

For contrib entity forms using other form structures, if they have a selector, that is **not hidden**.

The presave enforcement fires regardless of whether the selector was visible, hidden, or absent, but we hide the selectors to simplify the UIs and not mislead the user.

Extension install: full rewrite, not just `en`

When a module or theme is installed with a lock active:

- Locale's own install hooks are removed (`#[RemoveHook]`)
- The lock queues its own batch: `updateConfigForLockedLanguageSwitch()`
- That batch rewrites **every config object** with a `langcode` key
  → to the lock language, regardless of current value

Newly shipped langcode	No lock + locale	Lock = `xx`
`en`	🟢 `de` (site default)	🟢 `xx`
(empty)	🟢 `de` (normalized)	🟢 `xx`
`xx` (known on site)	🔴 `xx`	🟢 `xx`
`zz` (unknown on site)	🔴 `zz`	🟢 `xx`
No translatable elements	🔴 `en`	🟢 `xx`

Recipes: response when applied

`hook_entity_presave` — fires during recipe application too for config entities only

`RecipeAppliedEvent` **subscriber** — fires after the entire recipe completes:

- Reads `$event->recipe->config->getConfigStorage()->listAll()` — scoped to **that recipe's own `config/` directory only**
- Calls `updateConfigForLockedLanguageSwitch()` — a raw config storage write that covers **both simple config and config entities**

> *Same recipe → same langcodes assuming the same lock language → regardless of environment or request language.*

Independent from site default by default

Can optionally follow site default language

When enabled on the settings page changing the site default on `admin/config/regional/language` automatically:

1. Updates `locked_langcode` to the new site default
2. Runs the full rewrite batch immediately

The language selector on the lock settings page is disabled (site default controls it)

Some people expect this to happen, while others may expect config stays independent.

Settings page: mini language audit

`admin/config/regional/config-language-lock` shows how many config objects are currently in each langcode.

Language	Count	
en	142	
de	3	
xx	1	
(empty)	2	⚠

Useful for quick checks to see if a cleanup is needed. Added in case the enforcement methods are not 100% effective. You can still review if there are gaps and manually fix them.

After setting a lock, the batch runs and the table shows one language only.

Language deletion protection

The locked language is prevented from being deleted via the UI.

- `hook_entity_access` returns forbidden for `delete` on the locked language entity.
- The delete link disappears from the language overview table.
- The locked language is annotated with "(Configuration language)" in the list.

If the locked language is somehow removed (e.g. via drush): `getLockedLangcode()` validates it against installed languages. Returns `NULL` → lock is treated as inactive — no rewrite to a nonexistent language.

Summary of fixes in Config Language Lock

Problem	Without lock	With lock
UI forms and config actions follow request language	🔴 Silent	🟢 Presave enforces lock
Some config entities have UI language selectors	🟡 Selector	🟢 No language selector
Changing site default doesn't rewrite config	🔴 Silent	🟢 Lock switch batch (or follow-site-default)
Module install: non-`en` config not rewritten	🔴 Silent	🟢 Lock batch rewrites all langcodes
UI-created config never rewritten by locale	🔴 Silent	🟢 Lock applies to all active config
Config with no translatable elements stays `en`	🔴 Silent	🟢 Lock rewrites regardless
Theme install: locale batch never runs	🔴 Silent	🟢 Lock batch runs via batch continuation
Recipes import langcodes as-is	🔴 Silent or 500	🟢 Presave + subscriber normalize
Bad langcode causes 500 in `FullyValidatable` entities	🔴 500 error	🟢 Presave normalizes before validation

Side effect: intentionally creating or installing base config in multiple languages is not possible

This is by design though for those using the module

Install. Configure. Lock.

```
composer require drupal/config_language_lock
```

```
admin/config/regional/config-language-lock
```

Core issue: Introduce a dedicated "Configuration default language" different from "Site default language"

#3337864

Modules to try out

- drupal.org/project/language_audit — read-only dashboard to review site status
- drupal.org/project/config_language_lock — enforces one lock language on all config entity saves, module/theme installs, recipe applies and its own form

Core issues (already fixed in Drupal 11.5 / 12.0)

- #3612163 — Theme install: standard UI redirect skips locale's rewrite batch entirely
- #3600904 — Locale skips config with no translatable elements at install time

Core issues (open)

- #3472317 — Recipes import `langcode` verbatim; bad values are saved then cause 500s on `FullyValidatable` entities
- #3614335 — Config Translation requires Locale even though the only thing it uses is a one-line language manager call