Shared Translation Architecture

Shared Translation Architecture

Purpose

Define the shared translation catalogue used across all client applications in the suite.

This repository exists to:

- Ensure consistent language across surface-detail, open-invitation, and personal-interface.
- Prevent translation drift.
- Provide stable localisation keys aligned with backend error codes.
- Enable multi-language support from day one.
- Treat translation keys as part of the platform contract.

shared-i18n is the single canonical source of truth for:

- Shared translation keys
- Canonical key set (en.json)
- Locale catalogues
- Versioning rules

Scope

In scope:

- Generic system messages
- Authentication messages
- Shared error states
- Common UI actions and labels

- Cross-application terminology
- Validation message templates

Out of scope:

- Feature-specific UI copy
- Marketing content
- Page-specific long-form text
- Email templates
- Guest communications

App-specific strings must remain inside each application.

Repository Structure

```
shared-i18n/
  package.json
  src/
    locales/
      en.json        (canonical key set)
      de-DE.json
```

Additional locales may be added over time.

Canonical Locale

- en.json is the canonical key set.
- All other locale files must contain exactly the same keys.
- No key may exist in any locale file if absent from en.json.

CI must fail if:

- A key exists in canonical but is missing in another locale.
- A key exists in another locale but not in canonical.

- Placeholder parameters differ across locales.

Key Naming Rules

Keys must:

- Use dot notation.
- Be stable and versioned carefully.
- Not contain dynamic content.
- Be grouped under clear namespaces.

Approved namespaces:

- errors.generic.*
- errors.validation.*
- auth.*
- common.*
- navigation.*

Example keys:

- errors.generic.not_found
- errors.generic.request_error
- errors.generic.service_unavailable
- auth.signed_out
- common.loading
- common.cancel

Keys must not embed variable values.

Dynamic Parameters

Dynamic values must be supplied via placeholders.

Example:

Key:

```
errors.validation.min_length
```

Value:

```
"Must be at least {{min}} characters."
```

All locales must use identical placeholder names.

Language Requirements

From day one:

- At least two languages must be fully implemented.
- No locale may be partially complete.
- Missing translations must fail CI.

Initial supported languages:

- en
- de-DE

Generated Type Contract

- The TranslationKey type is generated from the canonical key set.
- translationKeys array is generated for tooling and audits.
- No manual maintenance of key unions.
- Consumers must import types from shared-i18n.

The generated type contract must be derived automatically from the canonical key set. Manual maintenance of key unions is prohibited.

The generated outputs represent the shared key contract for client compile time validation and tooling.

Canonical Shared Key Set

The canonical shared key set is defined exclusively by the keys present in `src/locales/en.json`.

Documentation must not enumerate keys manually. The source file is authoritative.

Versioning

shared-i18n must:

- Be versioned.
- Follow semantic versioning.
- Publish tagged releases.

Breaking changes include:

- Removing keys.
- Renaming keys.
- Changing placeholder structure.

Clients must explicitly upgrade versions.

Additional rules:

- No key removal without a major version bump.
- Renaming a key counts as removal plus addition and requires a major version bump unless the old key remains present as an alias.
- Additive keys are allowed in minor versions.
- Patch versions are limited to packaging or documentation changes only.

Client Integration Contract

Each client application must:

1. Import shared-i18n as a dependency.
2. Import TranslationKey from shared-i18n.

3. Load shared translations during application bootstrap.
4. Merge shared translations with app-specific translations at runtime.
5. Not override shared keys unless explicitly approved.

Clients in scope:

- surface-detail
- open-invitation
- personal-interface

Clients must not:

- Duplicate shared keys locally.
- Modify shared translations directly.
- Maintain independent copies of shared key unions.

Ownership Boundaries

shared-i18n provides:

- The canonical shared key set (en.json)
- Locale catalogues for all supported languages
- Generated type contract (TranslationKey and translationKeys)
- Versioning and change control rules
- CI enforcement of catalogue integrity

Each client application remains the runtime owner of:

- Locale resolution
- Translation function
- Interpolation and formatting behaviour
- Runtime loading and merging of shared and app-specific catalogues

shared-i18n does not:

- Store runtime user preferences

- Resolve locale selection
- Perform formatting
- Store server-side translations

It is a static catalogue plus generated type contract.

Error Code Alignment

Backend services return stable error.code values.

Clients must map error.code to translation keys in shared-i18n.

Example:

Backend:

```
error.code = "not_found"
```

Client mapping:

```
errors.generic.not_found
```

Rules:

- Backend must not return localised human strings.
- Clients must not rely on the message field for localisation.
- error.code must be stable and treated as part of the API contract.

polite-intervention performs error mapping only and does not localise.

CI Requirements

CI for shared-i18n must:

- Validate identical key sets across all locales.
- Validate JSON syntax.
- Validate placeholder consistency.

- Prevent duplicate keys.
- Prevent unused placeholders.

CI must fail if:

- A key exists in canonical but is missing in another locale.
- A key exists in another locale but not in canonical.
- Placeholder parameters differ across locales.

Each client must:

- Snapshot test at least one screen in each supported locale.
- Verify generic error states render localised messages.

Promotion Rule

If a translation key is used by more than one client:

- It must be promoted to shared-i18n.
- It must be removed from app-specific translation files.
- A new version must be released.

Ownership

- Platform team owns shared-i18n.
- Application teams may propose additions via pull request.
- Removing or renaming keys requires architectural review.

Responsibility Summary

Layer	Responsibility
shared-i18n	Shared cross-application translations, canonical key set, locale catalogues, generated type contract, versioning rules
surface-detail	App-specific UI strings and runtime i18n logic

Layer	Responsibility
open-invitation	App-specific UI strings and runtime i18n logic
personal-interface	App-specific UI strings and runtime i18n logic
Backend services	Structured error codes only
polite-intervention	Error mapping, no localisation