TranceLate – Protokoll Benutzer- & Sprachenverwaltung (Stand 2025-08-25)

0) Ziel & Scope

Ziel dieses Protokolls ist es, die aktuell umgesetzten und geplanten Mechanismen für **Benutzer-/ Rollenverwaltung (RBAC)**, **Sprachenverwaltung** und die **Übersetzungs-Engine** (inkl. Provider-Gating) präzise zu dokumentieren. Es dient gleichzeitig als **Handover-Dokument** und als **Kurz-Runbook** für Betrieb & Weiterentwicklung.

---

1) Benutzer- & Rollenverwaltung (RBAC)

1.1 Datenmodell (Kern)

- `public.profiles`
  Felder (Auszug): `id (uuid, PK)`, `role (text, default 'client')`, `is_expert_admin (bool)` …
- Rollen/Permissions (projektbezogen):
- `public.roles` — System- & Custom-Rollen (`code`, `scope`, `is_system_role` …)
- `public.member_roles` — Zuordnung *User* ↔ *Projekt* ↔ *Rolle*
- `public.role_permissions` — Zuordnung *Rolle* ↔ *Permission*

Seed/Systemrollen (Auszug): `project_owner`, `admin`, `editor`, `viewer`, `translator`, `reviewer`, `guest`, `api_user`, `billing_manager` …

1.2 Server-Funktionen (RBAC-Basis)

- `public.is_user_portal_owner(user_uuid uuid)` → `boolean`
- `public.is_user_expert_admin(user_uuid uuid)` → `boolean`
- `public.get_user_admin_level(user_uuid uuid)` → `text` mit Werten `user | admin | expert_admin | portal_owner`
- **Snapshot**: `public.authz_snapshot(user_uuid uuid)` → Tabelle `(admin_level text, is_portal_owner bool, is_expert_admin bool)`
  → *zentrale, stabile Abfrage für Client-Hooks*

1.3 Client-Hooks & Realtime

- `usePortalOwner()`
  Verwendet `authz_snapshot`, liefert `isPortalOwner`, `adminLevel`, Audit-Logging-Helper u.a.
- `useExpertAdmin()`
  Ebenfalls `authz_snapshot`, berechnet `isExpertAdmin` (inkl. implizit bei `portal_owner`).
- `subscribeAuthz(userId, onChange)`
  Konsolidiert alle authz-relevanten Tabellen in **einem** Realtime-Channel (Debounce). Nutzt: `member_roles`, `role_permissions`, `profiles`.

1.4 Route-Gates & UI-Indikatoren

- `useRouteAccess()` liest **ROUTE_RULES** (Expert/Owner-Gates + Permission-Gates) und entscheidet Navigation.
- UI-Komponenten:
- `RoleIndicator` & `UserRoleDisplay`: konsistente Anzeige des Admin-Levels / Rolle, Badge/Icons.
- `AppSidebar`: Badge-Render basierend auf `adminLevel` (`portal_owner`, `expert_admin`, `admin`, `user`).

1.5 Fixes/Diagnostik

- **Fehlerbild**: zeitweise Doppelquelle/Inkonsistenz (ein Widget zeigte `user`, anderes `portal_owner`).
  Ursache: gemischte Nutzung älterer RPCs vs. `authz_snapshot` + Caching.
  Fix: überall konsistente Nutzung von `authz_snapshot`; Defensive Normalisierung `admin_level` im Client.
- **Schnell-Check** (SQL):

```
select * from public.authz_snapshot('<USER_UUID>');
```

---

2) Sprachenverwaltung (Language Management)

2.1 Datenmodell & Begriffe

- **Master**: `public.languages(code [PK], …)` — Referenzliste aller Sprachcodes (ISO, 2–5 Zeichen).
- **Projektsprachen**: `public.project_languages(project_id, language_code, is_default, path_mode, auto_publish, …)`
- **Pfad-Whitelist**: `public.project_language_paths(project_id, language_code, path_pattern, is_enabled)`
  Wird genutzt, wenn `path_mode='whitelist'`.
- **Entitlements (Abo-Fenster)**: `public.project_language_entitlements(project_id, language_code, active_from, active_to, is_active, allow_exact_after_expiry)`

Begriffsklarheit - **Default-Sprache**: *Basissprache* (Quelle der Wahrheit; wird nicht übersetzt). Pro Projekt **genau eine** (Unique Index empfohlen). - **Auto-Publish / effective_auto_publish**: Steuerung, ob neue Übersetzungen ohne Review live gehen; `effective_…` berücksichtigt globale/übergeordnete Regeln. - **Path-Mode**: `all` (alle Pfade erlaubt) oder `whitelist` (nur explizit erlaubte Pfade). - **Entitlement**: Zeitfenster (Abo) und Policy bei Ablauf (Bestand anzeigen vs. „hartes Cutoff“).

2.2 Server-Funktionen (Auszug)

- `list_project_languages_effective(p_project_id)` → `(language_code, is_default, auto_publish, effective_auto_publish)`
- `set_language_active(p_project_id, p_language_code, p_active bool, p_is_default bool)`

- `list_language_paths(p_project_id, p_language_code)` → `(path_pattern, is_enabled, path_mode)`
- `path_is_allowed(p_project_id, p_language_code, p_path)` → `boolean`
- `entitlement_is_active(p_project_id, p_language_code)` → `boolean`

Hinweis: In frühen Ständen lieferte `list_project_languages_effective` nur **registrierte** (aktive) Sprachen. Die UI wurde darauf angepasst (siehe 2.4), um Doppelquellen zu vermeiden.

2.3 Admin-UI (aktuelle Architektur)

- **Page**: `src/pages/admin/languages.tsx`
- nutzt `useLanguageManagement()` als **Single Source of Truth** (SSOT)
- gibt `langs` (registrierte Sprachen), `selectedLang`, `paths`, `pathMode`, `pendingApprovals` u.a. an Kinder weiter
- lädt **Settings** (Entitlements, Pfadmodus, effective_auto_publish) für **registrierte Sprachen** und stellt sie als `settingsByCode` bereit
- **Hook**: `src/hooks/useLanguageManagement.ts`
- State: `selectedLang`, `langs`, `paths`, `pendingApprovals`, `pathMode`, `loading`, `settingsByCode`
- Actions: `handleLanguageSelect`, `addLanguage`, `toggleActive`, `setAutoPublish`, `setPathMode`, `addPath`, `togglePath`, `approveAll`
- **Wichtig**: Nach Aktionen wird `loadLanguages()` **erneut aufgerufen**, das wiederum `loadSettingsForLangs()` ausführt → UI wird sofort konsistent
- **Komponenten**:
- `LanguageHeader` — listet **registrierte** Sprachen (optional Suchfeld), klickbare Auswahl
- `LanguageDetailView` — Detailkarte inkl. Pfad- & Auto-Publish-Steuerung, Entitlement-Infos
- `TranslationControlOverview` — **nur präsentational**, zeigt Status der **registrierten** Sprachen anhand `langs + settingsByCode` (keine eigenen RPCs)

2.4 Bedienlogik (erwartetes Verhalten)

1. **Übersicht (Overview)**: zeigt **alle registrierten** Sprachen; Status: Default/Auto/Entitlement/Fenster/Path-Mode.
2. **Details**: zeigt Konfiguration der ausgewählten Sprache.
3. *Aktivieren*: `addLanguage(code)` → registriert Sprache und lädt State neu.
4. *Deaktivieren*: `set_language_active(..., active=false)` → entfernt Registrierung und lädt State neu.
5. **Konsistenz**: Overview & Details beziehen Daten aus derselben Quelle (Hook + Settings-Map). Keine doppelten RPCs.

2.5 Operationen, die bereits durchgeführt wurden

- **Master & Projekt** mit **~47** Codes befüllt (z. B. `en, fr, es, …, zh`).
- **Entitlements** für alle registrierten Sprachen auf „immer aktiv“ hochgezogen: `active_to = 'infinity'`, `allow_exact_after_expiry = true` (Bestand bleibt sichtbar).
- **Default-Sprache** auf `de` umgestellt und empfohlenen Unique-Index angegeben: *max. 1 Default pro Projekt*.

---

3) Provider-Gating & Kostenkontrolle (Engine-Sicherheit)

3.1 Harte Bremse durch Policy-RPC (Server)

- Tabellen:
- `public.project_provider_defaults(project_id PK, provider_enabled bool default false, rpm_limit, tpm_limit, updated_at)`
- `public.project_provider_overrides(id, project_id, language_code, provider_enabled, rpm_limit, tpm_limit, updated_at, unique(project_id, language_code))`
- RPC: `public.get_provider_policy(p_project_id, p_language_code)`
  Liefert **effektiv**: `provider_enabled`, `rpm_limit`, `tpm_limit` (Override > Default > Fallback AUS).
- **Default** für das Projekt wurde auf **AUS** gesetzt (`provider_enabled=false`).

3.2 Edge-Gate (Backend-only)

- Der Übersetzungs-Endpoint darf **nur** vom Backend/Worker aufgerufen werden:
  Header-Gate: `x-trancelate-backend == BACKEND_TRANSLATE_SECRET` → sonst `403`.
- **Umweltvariable**: `FEATURE_PROVIDER=false` (bis GUI stabil) → Provider-Calls bleiben deaktiviert.

3.3 Ablauf der Engine (vereinfacht)

1. **TM-Exact**: Treffer aus `translations` (Hash/Key auf `textid` + `source_lang` + ggf. `style`).
2. **TM-Approx (Soft-Match)**: gemäß *Soft-match Spezifikation (tm-fuzzy Lookup)*.
3. **Provider-Gate**: nur wenn `FEATURE_PROVIDER=true` **und** `get_provider_policy(...).provider_enabled=true` (ggf. Rate-Limits beachten).
   Sonst: `missReason = 'model-off'` für UI-Banner/Telemetry.

3.4 TranslationStatusBanner (Client)

- Erklärt Nicht-Treffer (Miss-Reasons: `entitlement`, `path`, `budget`, `model-off`, `provider-fail`, `no-exact`, `unknown`).
- Verlinkt zur Sprachen-Administration (z. B. `/admin/languages`).

---

4) Runbook (Betrieb)

4.1 Sprachen sichtbar machen / prüfen

```
-- Alle Projektsprachen (registriert)
select language_code, is_default, path_mode, auto_publish
from public.project_languages
where project_id = '<PROJECT_ID>'
order by language_code;

-- Entitlements
select language_code, is_active, active_from, active_to,
```

```
allow_exact_after_expiry
from public.project_language_entitlements
where project_id = '<PROJECT_ID>'
order by language_code;
```

4.2 Default-Sprache setzen (Beispiel: `de`)

```
-- sicherstellen, dass de existiert
insert into public.project_languages(project_id, language_code, is_default)
select '<PROJECT_ID>', 'de', false
where not exists (
  select 1 from public.project_languages
  where project_id = '<PROJECT_ID>' and lower(language_code)='de'
);

-- alle Defaults aus
update public.project_languages set is_default=false where
project_id='<PROJECT_ID>';
-- de als Default
update public.project_languages set is_default=true
where project_id='<PROJECT_ID>' and lower(language_code)='de';

-- optional: Einzigkeit absichern
create unique index if not exists ux_project_languages_default_one
  on public.project_languages(project_id)
  where is_default is true;
```

4.3 Provider gezielt freischalten

```
-- Projektweit aktivieren (Demo-Fenster)
update public.project_provider_defaults
set provider_enabled=true, rpm_limit=60, tpm_limit=50000, updated_at=now()
where project_id='<PROJECT_ID>';

-- Nur einzelne Sprachen (z. B. en + fr)
insert into public.project_provider_overrides(project_id, language_code,
provider_enabled, rpm_limit, tpm_limit)
values
  ('<PROJECT_ID>','en',true,60,50000),
  ('<PROJECT_ID>','fr',true,60,50000)
on conflict(project_id,language_code) do update
set provider_enabled=excluded.provider_enabled,
    rpm_limit=excluded.rpm_limit,
    tpm_limit=excluded.tpm_limit,
    updated_at=now();
```

4.4 Frontend darf nie übersetzen (Check)

- Edge-Funktion: **ganz oben** prüfen:

```
const REQUIRED = process.env.BACKEND_TRANSLATE_SECRET;
const got = request.headers.get('x-trancelate-backend') ?? '';
if (!REQUIRED || got !== REQUIRED) return new Response('forbidden:
backend only', { status: 403 });
```

- `.env` / Secrets: `BACKEND_TRANSLATE_SECRET=<starkes_secret>`

---

5) Offene Punkte & To-Dos

1. **RLS/Privileges prüfen**: frühere Meldung *"forbidden (set_language_path_mode)"* → sicherstellen, dass Admin-RPCs als `SECURITY DEFINER` laufen und `search_path` auf `public` gesetzt ist.
2. **Optimistic Updates**: aktuell wird nach Aktionen re-geloadet (robust). Für bessere UX optional lokalen Reducer einführen (langs/paths/settingsByCode direkt mergen).
3. **GUI-Feinschliff**:
4. Fehlende Interaktion „Jetzt aktivieren" → bereits funktional, aber Success-Feedback/Spinner ergänzen.
5. Filter „Nur aktive" im Header optional wieder anbieten, **klar** als registriert vs. inaktiv kennzeichnen.
6. **Budgeting/Rate-Limit**: `rpm_limit`, `tpm_limit` aus der Policy vollständig im Edge anwenden (lokaler Rate-Limiter/Token-Bucket).
7. **Abo-Modelle**: Ablauf-Policy steuerbar machen (Hard-Cutoff vs. Bestand) per GUI + RPC.
8. **Smoke-Tests**:
9. Overview == Details (registrierte Sprachen),
10. Aktivieren/Deaktivieren ändert UI sofort,
11. Entitlement-Fenster wird korrekt angezeigt,
12. Provider-Gate sperrt wie erwartet.

---

6) Anhänge (Referenz)

6.1 Wichtige RPCs (Signaturen)

- `authz_snapshot(user_uuid uuid)` → `(admin_level text, is_portal_owner bool, is_expert_admin bool)`
- `list_project_languages_effective(p_project_id uuid)` → `(language_code text, is_default bool, auto_publish bool, effective_auto_publish bool)`
- `set_language_active(p_project_id uuid, p_language_code text, p_active bool, p_is_default bool default false)`
- `list_language_paths(p_project_id uuid, p_language_code text)` → `(path_pattern text, is_enabled bool, path_mode text)`
- `path_is_allowed(p_project_id uuid, p_language_code text, p_path text)` → `bool`
- `entitlement_is_active(p_project_id uuid, p_language_code text)` → `bool`

- `get_provider_policy(p_project_id uuid, p_language_code text)` → `(provider_enabled bool, rpm_limit int, tpm_limit int)`

6.2 Relevante Tabellen (Kurzschema)

- `languages(code PK, …)`
- `project_languages(project_id, language_code, is_default, path_mode, auto_publish, …)`
- `project_language_paths(project_id, language_code, path_pattern, is_enabled)`
- `project_language_entitlements(project_id, language_code, active_from, active_to, is_active, allow_exact_after_expiry)`
- `profiles(id PK, role, is_expert_admin, …)`
- `roles(id, code, label, scope, is_system_role)`
- `member_roles(id, user_id, project_id, role_id, assigned_at)`
- `role_permissions(id, role_id, permission_id, conditions, created_at)`

6.3 Environment Flags (Betrieb)

- `FEATURE_PROVIDER` — `false` = Provider immer aus; `true` = per Policy steuerbar.
- `BACKEND_TRANSLATE_SECRET` — Secret für Backend-only Gate (Edge).

---

7) Kurzfazit

- **RBAC**: auf `authz_snapshot` konsolidiert; UI zeigt konsistent `portal_owner` / `expert_admin`.
- **Sprachenverwaltung**: eine Quelle im Hook, registrierte Sprachen + Settings; Overview/Details sind synchron.
- **Kosten/Control**: Provider standardmäßig blockiert (Policy + Edge-Gate); Aktivierung pro Projekt/Sprache möglich.
- **Betrieb**: SQL-Snippets & Checks vorhanden; weitere GUI-Finetunes und Tests sind planbar ohne Risiko für Kostenlawinen.