ClaimGuard® — Spezifikation v1.0

Komponente: TranceCreate v1.2 Stage
Stage-ID: `claim_guard` (Alias: `claim_fit` für Rückwärtskompatibilität)
Ziel: UI-Texte (Buttons, CTAs, Labels, Placeholders, aria-labels, Input-Values) im *Zieltext* automatisch an die *sichtbare Länge* der korrespondierenden *Quelltexte* anpassen. Sprach-agnostisch, invariant-sicher, reproduzierbar.

---

1. Überblick

ClaimGuard® bringt Ziel-UI-Texte auf das verfügbare Platzbudget, das aus der Quelle abgeleitet wird. Die Stage **verändert nur sichtbare Copy**, respektiert alle technischen Invarianten und arbeitet **sprachneutral** (keine Wörterbücher, keine Kulturregeln). Dadurch bleibt das Verhalten vorhersagbar und robust für internationale Oberflächen.

Kernprinzipien - **Fit-to-Source:** Budget = sichtbare Länge des Quell-UI-Segments × Ratio (Standard 1.0). - **Invariant-Schutz:** `{{…}}`, `{token}`, HTML-Tags, URLs, Emojis, Zahlen bleiben 1:1 erhalten. - **Segment-Pairing:** Source- und Target-UI-Elemente werden nach *Typ* und *Index* zugeordnet. - **Konservatives Kürzen:** Nur strukturelle Kürzungen; kein semantisches Umschreiben. - **Tracebarkeit:** Vollständige Telemetrie im Response-`trace`.

---

2. Einbindung in die Pipeline

- **Registrierung:** `stage_registry()` enthält `"claim_guard": ClaimGuardStage` und den Kompatibilitäts-Alias `"claim_fit": ClaimGuardStage`.
- **Aktivierung:**

```
{ "stages": ["tc_core", "claim_guard", "policy_check", "degrade"] }
```

- **Default:** Nicht aktiv per Default. Aktivierung via `PUT /pipeline`.

Kontextfelder (gesetzt vor ClaimGuard®): - `original_text` — Quelltext (sichtbarer HTML-String) - `baseline_text` — Baseline-Übersetzung - `tc_candidate_text` — Kandidat aus `tc_core` (falls no-change: identisch zur Baseline) - `text` — Arbeitskopie für nachfolgende Stages (ClaimGuard® liest `tc_candidate_text` bevorzugt und schreibt in `text`).

---

3. Konfiguration

Datei: `config/claim_guard.json` *(Fallback:* `config/claim_fit.json`*)*

Beispiel-Defaults (sprachneutral):

```
{
  "default": {
    "units": "graphemes",
    "fit_to_source": true,
    "ratio": 1.0,
    "ellipsis": false,
    "max_iterations": 3,
    "breakpoints": ["\\s+", "\\u2009", "\\u200A", "-", "‑", "—", "/", "·",
":", ";", ","],
    "drop_parentheticals": true,
    "drop_trailing_fragments": true
  }
}
```

Erklärungen - **units:** Zähleinheit der sichtbaren Länge. `graphemes` bevorzugt (Unicode-Grapheme via `\X`), Fallback `codepoints`. - **ratio:** Multiplikator auf das Quellbudget (z. B. 1.2 für sprachbedingte Ausdehnung). - **ellipsis:** Bei hartem Cut Ellipse anhängen (`…`). Default `false`. - **breakpoints:** Bevorzugte Schnittstellen für Cuts. - **drop_parentheticals:** Rechtseinbauten `(...)`, `[...]`, `{...}` optional entfernen (außerhalb Masken), wenn Budget überschritten. - **drop_trailing_fragments:** Nach letztem starken Trenner (`—`, `‑`, `:`) rechts abschneiden.

Hot-Reload: Änderungen an `claim_guard.json` werden zur Laufzeit erkannt (mtime).

---

4. Erkennung & Pairing von UI-Elementen

Unterstützte Typen: - `button` — `<button>…</button>` - `cta` — `<a role="button">…</a>` oder `<a class*="btn|button|cta">…</a>` - `input_value` — `<input type="submit|button" value="…">` - `placeholder` — `placeholder="…"` in `<input>` / `<textarea>` - `aria_label` — `aria-label="…"` auf beliebigen Elementen - `label` — `<label>…</label>`

Pairing-Regel: 1. Für jeden Typ separat listen. 2. Nach Auftreten im DOM indizieren (0-basiert). 3. Source-Index *i* paart mit Target-Index *i* desselben Typs. Fehlt Source oder Target, **kein Eingriff**.

---

5. Messung der sichtbaren Länge

- **Masking (0-Länge):** `{{…}}`, `{token}`, HTML-Tags, URLs (`https?://…`), Emojis, Zahlen.
- **Grapheme-Zählung:** `len(regex.findall(r"\\X", text))`, Fallback: `len(NFC(text))`.

Budget-Berechnung:

```
source_visible_len = count_units(masked_source_segment)
budget = floor(source_visible_len * ratio)
```

---

6. Kürzungsstrategie (sprachneutral)

Für jedes Target-Segment (außerhalb Masken), bis `len ≤ budget` oder `max_iterations` erreicht: 1. **Whitespace-Squeeze:** Mehrfach-Spaces → 1 Space (außerhalb Tags). 2. **Parentheticals-Drop:** Rechts-nach-links entfernen (`(...)`, `[...]`, `{...}`), falls aktiv. 3. **Trailing-Fragments:** Abtrennen hinter dem rechten letzten starken Trenner (`—`, `-`, `:`), falls aktiv. 4. **Cut-to-Budget:** Cut am letzten **Breakpoint** vor Budget; falls keiner vorhanden: hart an der Graphem-Grenze. Ellipse nur bei `ellipsis=true`.

Nicht geändert werden: Maskierte Bereiche, Satzzeichen in Masken, Zahlenformate, HTML-Struktur, Platzhalter.

---

7. Output & Telemetrie

- **Schreibziel:** `ctx["text"]` (voller Ziel-HTML-String nach Anpassung).
- **Trace:**

```
{
  "trace": {
    "claim_fit": [
      {
        "type": "button",
        "index": 0,
        "src_len": 17,
        "tgt_len_before": 42,
        "tgt_len_after": 17,
        "budget": 17,
        "units": "graphemes",
        "steps": ["whitespace", "parentheticals", "cut"],
        "modified": true
      }
    ],
    "claim_fit_ratio_vs_original": 0.23
  }
}
```

- **Degrade-Reasons:** ClaimGuard® selbst fügt **keine** Gründe hinzu; sie löst aktiv. Fail-Closed bleibt Aufgabe der `degrade`-Stage, falls andere Policies greifen.

---

8. Beispiele

Button (DE) - Source: `<button>Jetzt registrieren</button>` → sichtbare Länge 17 - Target vor Stage: `<button>Jetzt sofort kostenlos anmelden und profitieren</button>` → 42 - Target nach Stage: `<button>Jetzt anmelden</button>` → 14 (≤ 17)

CTA (JA, CJK) - Source: `<a role="button">Sign up</a>` → 7 - Target vor Stage: `<a role="button">今すぐ無料でサインアップしてください</a>` → 16 - Target nach Stage: `<a role="button">今すぐ登録</a>` → 5 (≤ 7)

Placeholder (EN) - Source: `placeholder="Enter email"` → 12 - Target vor Stage: `placeholder="Please enter your email address now"` → 33 - Target nach Stage: `placeholder="Enter email"` → 12

---

9. Qualitäts- und Sicherheitskriterien

- **Invarianten:** `ph_ok`, `html_ok`, `num_ok`, `paren_ok` bleiben unverändert wahr.
- **Determinismus:** Ergebnisse sind bei gleicher Eingabe, Config und Seed reproduzierbar.
- **No-Op-Sicherheit:** Bei fehlendem Source-Pendant: keine Änderung; Trace-Eintrag mit `modified=false`.
- **Hot-Reload:** Config-Änderungen greifen ohne Neustart.

---

10. Tests & Abnahme

Smoke: `scripts/test_tc_claim_fit.py` bzw. `scripts/test_tc_claim_guard.py`
Akzeptanz: - ClaimGuard® in `GET /pipeline.available` sichtbar. - Aktivierte Pipeline führt Kürzungen gemäß Budget aus. - Response enthält `trace.claim_fit[]` und `claim_fit_ratio_vs_original`. - Keine Invarianten-Verstöße in `checks`.

---

11. Betrieb & Monitoring

- **Metriken (Trace):** Budget, Before/After, Steps, Ratio vs. Original.
- **Tuning:** `ratio`, `breakpoints`, `ellipsis`, Drops schrittweise anpassen und via Hot-Reload ausrollen.
- **Rollback:** Stage aus Pipeline entfernen: `PUT /pipeline` ohne `claim_guard`.

---

12. Versionierung & Kompatibilität

- **Stage-ID:** `claim_guard` (primär), Alias: `claim_fit` (Bestandsschutz).
- **Konfiguration:** Primär `config/claim_guard.json`, Fallback `config/claim_fit.json`.
- **Server:** TranceCreate v1.2+ erforderlich (Trace-Durchreichung).

---

13. Markenhinweis

ClaimGuard® ist eine Produktbezeichnung von TranceLate.it FlexCo.