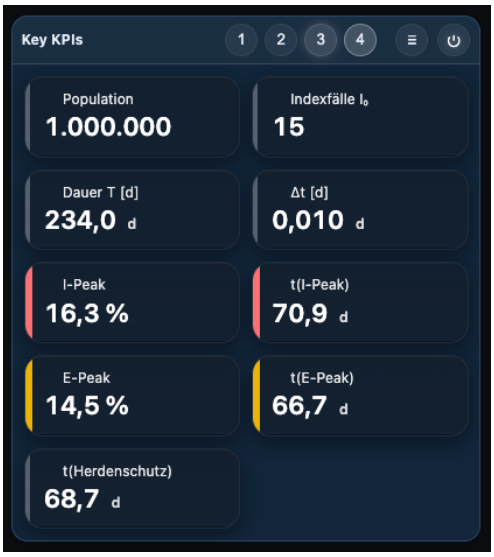
KPI-Tool
Technische Referenz v3.1.0



Zielgruppe: Entwickler:innen, Reviewer, QA

Stand: 03.10.2025
Scope: Bridge · Live · Key · Common (Compute/Registry/Style/Wiring)

---

1) Architektur im Überblick

Das KPI-Tool liefert eine konsistente Kennzahlen-Darstellung als eigenständige Präsentationsschicht. Es ist modular (ESM), ereignis-getrieben (Event-Bus) und über eine Bridge als Public API eingebunden. Zwei Ansichten (Live-Bereich und Key-Bereich) teilen sich denselben Rechenkern (STATIC/LIVE), dieselbe Registry (Labels/Ordnungen), dieselbe Stil-Logik (Controller/Classifier) und ein gemeinsames Wiring (Topics/Alias-Brücke).

Prinzipien: One-Header-Policy, Host-scoped Styles, „Deck“ als Render-Container, Sticky-Replay beim Mount, klare Trennung von Daten (model/sim) und Darstellung (cards).

---

2) Dateien & Rollen (Bridge · Live · Key · Common)

Bridge (`kpi/bridge.js`): Konsolidierter Export (Public API) für Live/Key/Compute/Registry/Tooltips/Wiring/Style. Verbraucher importieren bevorzugt aus `@uid/pres/kpi`.

Live (`kpi/live/*`): Orchestrierung des Live-Bereichs, View-Hilfen (Header/Style-Guard/Order/Card/Render) und Widget-Actions für Gruppen/Format/Komfort.

Key (`kpi/key/*`): Mount + Renderer für die kuratierte Zwei-Spalten-Sicht (statisches Deck) und Actions (Ziffern 1–4, Format, kompakt).

Common (`kpi/common/*`): Compute-Kern (STATIC/LIVE/Extras), Registry (Labels & Orders), Tooltips (DE/EN · Schul/Uni), Style (Tokens/Controller/Icons/Classifier), Wiring (Topics/Brücke/Mount-Helfer).

---

3) Datenfluss & Boot (Compute/Registry/Wiring/Style-Guard)

Beim Mount werden Header, Body und Styles initialisiert, danach werden letzte „Sticky“-Werte (Model/Sim/Pointer) via Bus abgeholt und erneut ausgesendet (Replay). STATIC-Werte stammen aus dem letzten Modellzustand, LIVE-Werte aus Simulationsreihen plus Zeiger. Die Registry definiert Labels und Reihenfolgen pro Modell (SIR/SEIR …) und Modus (Schule/Uni). Der Style-Guard erzwingt Spalten, setzt Default-Attribute (`data-format`, `data-reduced`) und klassifiziert Karten nach jedem Render.

---

4) Interaktion (Live-Bereich · Key-Bereich · Kopfzeile)

Live-Bereich (Deck): Zeigt dynamische Kennzahlen als kombinierte oder gruppierte Reihenfolge (Kompartimente, Kontext, Peaks, Outcomes). 4 Spalten, Formatumschaltung (Prozent/Absolut/Hybrid), kompakte Ansicht optional.

Key-Bereich (Deck): Kuratierte 2-Spalten-Übersicht in Gruppen: „Kuratiert“, „Modell-KPI“, „Simulations-KPI“, „Synthese“. Ziffern 1–4 schalten Gruppen.

Kopfzeile: Links Titel, rechts Actions (Ziffern, Format, „Kompakt“, Burger-Menü). Die Auswahl wird browserseitig gespeichert.

---

5) Event-Referenz (KPI/KEYKPI-Topics, Alias-Brücke)

KPI_TOPICS

- `uid:kpi:view:format` — `{ mode:'pct'|'abs'|'hybrid' }`
- `uid:kpi:view:reduced` — `{ on:true|false }`
- `uid:kpi:enabled` — `{ on:true|false }`
- `uid:kpi:group:set` — `{ id:'comp'|'context'|'peaks'|'outcomes', on:true|false }`
- `uid:kpi:deck:toggle` — kompatibler Alias (optional)

KEYKPI_TOPICS

- `uid:keykpi:view:format` — `{ mode:'pct'|'abs'|'hybrid' }`
- `uid:keykpi:view:reduced` — `{ on:true|false }`
- `uid:keykpi:enabled` — `{ on:true|false }`

Alias-Brücke: Spiegelung der View-Events zwischen `kpi` und `keykpi` (loop-safe; Payload-Guard mit interner Markierung). Optionales Replay der letzten Werte beim Start.

---

6) Accessibility (A11y) & I18N

Karten sind fokussierbar (`tabindex=0`, `role=group`) und tragen eine sprechende `aria-label` (Titel, Wert, Einheit). Die Kopfzeile ist vollständig mit der Tastatur bedienbar. Tooltips stören den Fokus nicht. Alle Labels/Tooltips liegen in DE/EN vor; die Auswahl richtet sich nach `lang` am Dokument. Schul- und Unimodus liefern unterschiedliche Texttiefe.

---

7) Styles & Klassifikation (Controller · Tokens · Icons)

Der Style-Controller injiziert ein host-lokales Stylesheet, erzwingt Spalten (`forceCols`), setzt Default-Variablen und stößt die Klassifikation an.

Klassifikation: drei Typen von Karten – *comp* (Kompartimente S/E/I/R), *assoc* (assoziierte Messwerte wie Peaks/Zeitpunkte), *metric* (neutrale Metriken). Schienenbreite, Dots und Konturen folgen dem Typ.

Tokens & Icons: Akzentfarben für S/E/I/R/D/V; Icons: Pfeil ($R_e$ff), Zielscheibe (tHIT), Herz (Attack Rate/Angriffsrate), Uhr (t-Werte).

---

8) Performance & Robustheit

Einmalige Style-Injection pro Host, leichte DOM-Strukturen (Karten als Kacheln), keine teuren Layout-Thrashes. Replay vermeidet „leere“ Decks nach dem Mount. MutationObserver triggert nur Klassifikation. Defensive Guards bei optionalen Abhängigkeiten (Bus/Tooltip-Systeme).

---

9) Verträge & Modelle (Karten-Modelle STATIC/LIVE/KEY · Deck-API)

STATIC (modellbasiert): `R0, beta, gamma, D, sigma, L, m, N, I0, T, dt, Reff0, HIT, betaEff, T2`.

LIVE (simulationsbasiert): `t, S_t, E_t, I_t, R_t, Ipeak, Epeak, tpeak, tEpeak, Reff_t, Attack, tHIT`.

KEY-Gruppen: `goal, model, sim, synth` (jeweils eigene Reihenfolge).

Deck-API: Live: kombinierte Reihenfolge aus Teilgruppen oder Einzel-Reihenfolgen; Key: statisches 2-Spalten-Deck.

---

10) Off-Policy & Enable-Flags

Widgets können deaktiviert werden, ohne sie zu entfernen. `enabled=false` wirkt nur auf die UI (sichtbar/bedienbar), das Wiring bleibt erhalten. Off/On respektiert lokale Persistenz und setzt `data-widget-enabled` entsprechend.

---

11) Rehydrate & Seeds (Replay-Flow)

Nach dem Mount werden die letzten Werte (`model:update`, `sim:data`, optional `sim:pointer`) abgefragt und erneut ausgesendet. Wenn keine Simulationsdaten vorliegen, kann ein sanfter Nudge („Seed“) gesendet werden, um Consumer zu initialisieren.

---

12) Bridge & Import-Map (Public API)

Import-Map (Beispiel):


```
{
"imports": {
"@uid/pres/kpi": "/12-3_presentation/kpi/bridge.js",
"@uid/pres/kpi/live": "/12-3_presentation/kpi/live/index.js",
"@uid/pres/kpi/key": "/12-3_presentation/kpi/key/index.js",
"@uid/pres/kpi/common/": "/12-3_presentation/kpi/common/"
}
}
```


Public API (aus `@uid/pres/kpi`): `mountLiveKPI`, `mountKeyKPI`, `attachKPIStyle`, `makeContext`, `LIVE`, `STATIC`, `getRegistry`, `kpiLabel`, `kpiTooltip`, `KPI_TOPICS`, `KEYKPI_TOPICS`, `wireKPIAliases`, `mountKPIWiring`.

---

13) QA-/Abnahme-Checkliste

1. Header vorhanden (Titel + Actions-Slot), Body als Grid.
2. `data-format` und `data-reduced` gesetzt (Defaults je Modus).
3. Style-Scope pro Host eindeutig (`data-kpi-style`).
4. Klassifikation sichtbar (Rails/Dots/Outline korrekt).
5. Live-Deck: 4 Spalten; Key-Deck: 2 Spalten erzwungen.
6. Replay greift: Deck nach Mount nicht leer.
7. Events senden/empfangen (Format/Reduced/Groups).
8. Lokale Persistenz aktiv (Schalter behalten Zustand).
9. Tooltips sichtbar, Fokus ungestört.
10. Farb-Tokens greifen, Kontrast ausreichend.
11. Dispose/Unmount hinterlässt keine Listener-Leichen.

12. Fallbacks ohne Crash (fehlender Bus/Tooltip erlaubt).

---

14) Typische Fehlerbilder & Gegenmittel

- **Leeres Deck nach Mount:** Replay prüfen; Bus verfügbar? Seeds ggf. aktivieren.
- **Falsche Reihenfolge:** Registry-Order für Modell/Modus korrekt?
- **Format wirkt nicht:** `data-format` nicht gesetzt oder Event nicht gespiegelt.
- **Kompakt ohne Effekt:** `data-reduced` am Host prüfen; Stylescope stimmt?
- **Doppelte Tooltips:** Fremde Tooltip-Binder entfernen; nur einen Initializer verwenden.
- **Ziffern schalten nicht:** Persistenz-Key/Kartengruppen-Mapping überprüfen.

---

15) Mini-Rezepte (Copy/Paste)

- **Live-Deck montieren:** `mountLiveKPI('kpi-live', { deckId:'live' })`
- **Key-Deck montieren:** `mountKeyKPI(document.getElementById('kpi-key'))`
- **Format global setzen:** Event `uid:kpi:view:format` mit `{mode:'hybrid'}` senden.
- **Kompakt umschalten:** Event `uid:kpi:view:reduced` mit `{on:true}` senden.
- **Gruppe toggeln (Live):** `uid:kpi:group:set {id:'peaks', on:false}`.
- **Style-Tokens live setzen:** Controller-API `setTokens({ 'kpi-gap':'12px' })`.

---

16) Erweiterungsvorschläge

Legend-Sync zwischen Chart und Karten, konfigurierbare Gruppenreihenfolgen, Preset-Sets pro Kurs, JSDoc an Public API, Snapshot-Tests für Deck-Render, Telemetrie-Opt-in für UI-Wege (nur aggregiert).

---

17) Changelog (KPI-Tool)

v3.1.0: Live/Key entkoppelt, View-Layer ausgelagert, Registry/Tooltips vereinheitlicht, Wiring mit Alias-Brücke, Style-Controller verfeinert, kompakte Ansicht konsolidiert, Public-API via Bridge.

Anhang A — Kartenliste & IDs

STATIC: `R0, beta, gamma, D, sigma, L, m, N, I0, T, dt, Reff0, HIT, betaEff, T2`

LIVE: `t, S_t, E_t, I_t, R_t, Ipeak, Epeak, tpeak, tEpeak, Reff_t, Attack (Angriffsrate), tHIT`
KEY-Gruppen: `goal, model, sim, synth`
Formate: `pct, abs, hybrid` · **Ansichtsflag:** `reduced:true|false`