```
module: 28
title: Subscriptions / Billing
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-1, LP-2, LP-7, LP-8, LP-9, LP-10, LP-13]
spec_style: requirements-pur
```

Modul 28 — Subscriptions / Billing

> ***Detail-Spec gemäß LP-9.*** *Subscription-Engine und Billing-Anbindung an Stripe. Pricing-Modell selbst ist* ***geparkt*** *— die Engine wird so flexibel gebaut, dass konkrete Plans später konfigurativ definiert werden können (LP-7 + Brainstorming „Geparkt für später").* ***Apple-Compliance (LP-8):*** *keinerlei Pricing-/Upgrade-UI in der App selbst — Self-Service-Signup ausschließlich über Marketing-Site* `hivegrid.de`*.*

Themenblöcke

```
A. Datenmodell + Pricing-Konfiguration
B. Subscription-Lifecycle + Status-Übergänge
C. Self-Service-Signup-Flow + Stripe-Checkout
D. Seat-Modell (Büro vs. Monteur)
E. Trial-Phase
F. Dunning + Past-Due
G. Downgrade-Read-Only-Middleware
H. Frozen-Modus (User-Pause)
I. Service-Metering (Trigger-Volumen, AI-Token)
J. AI-Credits-System
K. Setup-Fee-Handling
L. Demo-Buchung (extern)
M. Routes + In-App-Sicht
N. Permissions-Anwendung
O. Klartext-Hinweise/-Fehler
P. Phase 2 / Backlog
```

Block A — Datenmodell + Pricing-Konfiguration

Grundprinzip

Engine ist **konfigurations-getrieben**. Plattform-Admin definiert Produkte, Preise, Features, Usage-Tiers über Datenstrukturen — Code kennt die konkreten Plans nicht. Damit kann das Pricing-Modell jederzeit angepasst werden, ohne Code-Releases.

Entitäten

`SubscriptionProduct`

Ein Plan-Produkt (z.B. `Cloud`, `Cloud+`, `Enterprise` — Namen werden später festgelegt).

- `id` (UUID), `created_at`, `updated_at`
- `code` (technischer Schlüssel, unique, snake_case — z.B. `cloud_basic`)

- `display_name` (User-sichtbar, locale-abhängig — DE Pflicht, weitere via Modul 30)
- `description` (Markdown)
- `tier_order` (Integer — für Vergleich „höher/niedriger" bei Up-/Downgrade)
- `is_active` (Boolean — neue Tenants können nur aktive Produkte wählen)
- `is_self_service_eligible` (Boolean — `Enterprise` z.B. nur per Plattform-Admin zuweisbar)
- `setup_fee_required` (Boolean — siehe Block K)
- `trial_default_days` (Integer, 5-7 — siehe Block E; null = kein Trial)
- `requires_manual_activation` (Boolean — Vertrieb gibt Trial frei, siehe Block E)

`SubscriptionPrice`

Konkreter Preispunkt (Stripe-Price-Mapping).

- `id`, `subscription_product_id` (FK)
- `stripe_price_id` (string, eindeutig)
- `currency` (ISO 4217 — `EUR` Default)
- `interval` (Enum: `month | year | one_time` — `one_time` für Setup-Fee)
- `interval_count` (Integer, default 1)
- `unit_amount_cents` (Integer)
- `tax_behavior` (Enum: `inclusive | exclusive` — Stripe-konform)
- `is_active`
- `valid_from`, `valid_until` (für preisliche Übergangsphasen)

`SubscriptionFeature`

Ein technisches Feature-Flag (siehe M27 Block F.1 für Loader-Convention).

- `id`, `code` (unique, snake_case — z.B. `kwp_integration`, `automation_advanced_branches`, `email_marketing`, `ai_assistant`)
- `display_name`, `description`
- `category` (Enum: `module | sub_feature | integration | ai`)

`SubscriptionProductFeature` **(M:N)**

Welche Features welcher Plan enthält.

- `subscription_product_id`, `subscription_feature_id`
- `quota_value` (JSON, optional — z.B. `{"max_users": 10}` oder `null` für unbegrenzt)

`SubscriptionUsageBasedPrice`

Definition eines Verbrauchs-basierten Preisteils (Trigger-Volumen, AI-Token).

- `id`, `subscription_product_id`
- `metric_code` (Enum: `automation_trigger | ai_token | sms_outbound | other`)
- `display_name`
- `included_amount` (Integer — im Plan inklusive)
- `overage_pricing_strategy` (Enum: `tier | per_unit | block`)
- `unit_label` (z.B. `1 Trigger`, `1.000 Token`)

`SubscriptionUsageBasedTier`

Stufen-Tarife für Überschreitungen.

- `id`, `subscription_usage_based_price_id`
- `from_amount`, `to_amount` (Integer — `to_amount` null = bis unendlich)
- `unit_amount_cents`
- `flat_amount_cents` (Pauschale für Block-Strategie)

`TenantSubscription`

Aktive Subscription pro Tenant.

- `id`, `tenant_id` (unique, M:1)
- `status` (Enum: `trial | active | past_due | cancelled | frozen` — siehe Block B)
- `subscription_product_id` (FK)
- `stripe_customer_id`, `stripe_subscription_id`
- `started_at`, `current_period_start`, `current_period_end`
- `trial_start`, `trial_end`
- `cancel_at_period_end` (Boolean)
- `cancelled_at`, `cancellation_reason` (optional)
- `frozen_since` (Timestamp — siehe Block H)
- `setup_fee_paid_at` (Timestamp, nullable)
- `last_dunning_attempt_at`, `dunning_attempt_count`
- `created_by_user_id`, `created_via` (Enum: `self_service | platform_admin | api`)

`TenantSubscriptionProduct`

Snapshot der gewählten Produkt-Konfiguration zum Zeitpunkt der Buchung (für Audit + Preisstabilität).

- `id`, `tenant_subscription_id`
- `subscription_product_id` (FK auf das Produkt-Original)
- `snapshot_data` (JSONB — kompletter Stand bei Buchung)
- `valid_from`, `valid_until`

`TenantSubscriptionProductPrice`

Welche `SubscriptionPrice`-Einträge dem Tenant zugeordnet sind (Plan-Preis + Setup-Fee + ggf. Add-Ons).

- `id`, `tenant_subscription_id`, `subscription_price_id`
- `quantity` (Integer — z.B. Seat-Anzahl)
- `is_metered` (Boolean — siehe Usage-Records)
- `valid_from`, `valid_until`

`TenantSubscriptionUsageRecord`

Verbrauchs-Datensatz für metered Pricing.

- `id`, `tenant_subscription_id`, `metric_code`, `period_start`, `period_end`
- `usage_amount` (Integer)
- `reported_to_stripe_at` (Timestamp, nullable)
- `created_at`
- Indexe auf `(tenant_subscription_id, metric_code, period_start)`

`CheckoutSessionStatus`

Tracking des Stripe-Checkout-Flows pro Tenant-Anlage.

- `id`, `tenant_id` (FK), `stripe_checkout_session_id`
- `status` (Enum: `pending | completed | expired | failed`)
- `intent` (Enum: `initial_signup | plan_change | reactivate_from_frozen | reactivate_from_past_due`)
- `created_at`, `completed_at`, `expires_at`
- `error_message` (string)
- `redirect_url_success`, `redirect_url_cancel`

`Credit`

AI-Credit-System (siehe Block J). Token-Quota mit Carry-Over-Regeln.

- `id`, `tenant_id`
- `credit_type` (Enum: `ai_token | automation_trigger_bonus | other`)
- `amount` (Integer — positive = Guthaben, negative = Verbrauch)
- `source` (Enum: `plan_inclusion | manual_grant | promo_code | overage_purchase | refund`)
- `valid_from`, `valid_until` (optional — Plan-Inclusion verfällt am Period-Ende)
- `balance_after` (Integer — gerechnete Saldo nach diesem Eintrag)
- `created_at`, `created_by_user_id` (NULL für System)
- `notes` (Klartext für Audit)

Indizes (Performance)

- `TenantSubscription`: unique auf `tenant_id`, Index auf `status`, `current_period_end`, `dunning_attempt_count`
- `TenantSubscriptionUsageRecord`: Composite-Index auf `(tenant_subscription_id, metric_code, period_start)`
- `Credit`: Index auf `(tenant_id, credit_type, created_at DESC)` für Saldo-Berechnung

Sonderfälle / Constraints

- `TenantSubscription` ist **nicht** RLS-gegated (App-Code permission-gegated, da Plattform-Admin alle Tenants sehen muss — siehe M27 A.4)
- `SubscriptionProduct/Price/Feature` sind **global** (kein `tenant_id`) — sie sind Plattform-Konfiguration
- `Credit` ist tenant-scoped mit RLS
- Soft-Delete für `SubscriptionProduct` über `is_active=false` (alte Tenants behalten Snapshot via `TenantSubscriptionProduct`)

Block B — Subscription-Lifecycle + Status-Übergänge

Status-Werte (entsprechen M27 A.5 Tenant-Status-Werten — `TenantSubscription.status` und `Tenant.status` sind synchronisiert):

```
trial      → Self-Service-Signup-Phase, Vertrieb-freigegeben für 5-7 Tage
active     → Bezahlend, voller Funktionsumfang
past_due   → Stripe-Zahlung gescheitert, Read-Only (siehe Block G)
cancelled  → User-Kündigung wirkt zum Period-Ende oder sofort
frozen     → User pausiert Workspace, reduzierte Gebühr, Read-Only (siehe Block H)
```

Übergangs-Tabelle

Von	Nach	Trigger
`trial`	`active`	Trial-Konversion: Stripe-Checkout-Setup erfolgreich abgeschlossen
`trial`	`cancelled`	Trial läuft ab ohne Konversion → Tenant wird zu `deletion_pending` (M27)
`active`	`past_due`	Stripe-Webhook `invoice.payment_failed`
`past_due`	`active`	Nachzahlung erfolgreich (Stripe-Webhook `invoice.payment_succeeded`)
`past_due`	`cancelled`	Nach 14 Tagen + 3 Dunning-Versuchen erfolglos → Stripe gibt `subscription.deleted`
`active`	`frozen`	User klickt „Workspace pausieren" + bestätigt reduzierten Plan
`frozen`	`active`	User klickt „Reaktivieren" + Standard-Plan-Checkout
`frozen`	`past_due`	Reduzierte Gebühr scheitert
`*`	`cancelled`	User klickt „Tenant kündigen" — wirkt zum Period-Ende, oder sofort bei Trial
`cancelled`	`active`	User zieht Kündigung innerhalb 30 Tagen zurück (Tenant noch nicht hard-deleted)

Übergangs-Hooks (Side-Effects)

Bei jedem Übergang wird ausgelöst:
- Audit-Log-Eintrag mit alt/neu Status, Trigger-Quelle, Stripe-Event-ID
- Activity-Stream-Eintrag (`subscription_status_changed`)
- Tenant-Status-Synchronisation (M27 A.5)
- Notification an Tenant-Admin (Email + In-App via M22)
- Bei Übergang in `past_due` / `cancelled` / `frozen`: Read-Only-Middleware-Cache invalidieren

---

Block C — Self-Service-Signup-Flow + Stripe-Checkout

Signup-Pfad (extern → intern)

Externer Pfad — siehe M27 A.1:
1. Marketing-Site `hivegrid.de/signup` (Next.js, extern, **nicht** Teil dieser Codebase)
2. Leitet zu `app.hivegrid.de/signup` für Account-Erstellung
3. Nach Account-Anlage: automatischer Tenant-Anlage-Schritt + Stripe-Checkout

In-App-Signup-Schritte (`/signup`)

1. Auth-Account-Anlage (siehe M27 B.1) — Email, Pwd, AGB, DSGVO, AVV
2. Tenant-Daten — Firmenname, Slug, Branche, Größe (siehe M27 A.1)
3. Plan-Auswahl mit `is_self_service_eligible=true`-Filter
4. Stripe-Checkout-Session-Erstellung mit `intent=initial_signup`
5. Redirect auf Stripe-Checkout (extern bei Stripe gehostet)
6. Nach Erfolg → `/{tenant}/onboarding/setup/` (siehe M29)

Checkout-Konfiguration

- **Stripe-Customer** mit Tenant-`id` als `metadata.tenant_id`
- **Subscription-Setup** mit Trial (siehe Block E)
- **Steuer-Behandlung:** Stripe Tax aktiv, EU-Reverse-Charge-Logik
- **Zahlungsmethoden:** SEPA-Lastschrift (Default DE/EU), Kreditkarte

- **DPA (Auftragsverarbeitungsvertrag):** wird elektronisch im Signup unterzeichnet (M27 A.1 — AVV-Akzeptanz-Checkbox)
- **EU-Entity-Präferenz:** Stripe Ireland Ltd. (Stripe Payments Europe DAC — EU-Vertragspartner)

Stripe-Webhooks (Pflicht-Handling)

Webhook-Event	Wirkung in HiveGrid
`checkout.session.completed`	`CheckoutSessionStatus.status = completed`, Trial-Aktivierung oder direkter Active-Wechsel
`invoice.created`	Rechnung lokal als Referenz speichern (Customer-Portal lädt Original aus Stripe)
`invoice.payment_succeeded`	Falls `past_due` → `active`-Übergang triggern
`invoice.payment_failed`	`past_due`-Übergang triggern, Dunning-Counter erhöhen
`customer.subscription.updated`	Plan-Wechsel propagieren (Snapshot aktualisieren)
`customer.subscription.deleted`	`cancelled`-Übergang triggern
`customer.subscription.trial_will_end`	3 Tage vor Trial-Ende: Notification + In-App-Banner

Idempotenz

Webhook-Handler **Pflicht-idempotent** über `stripe_event_id` als Unique-Constraint in einer `StripeWebhookLog`-Tabelle (Schreib-Idempotenz, AV #8).

Block D — Seat-Modell

Spiegelt LP-2 (Büro vs. Baustelle als kontextuelle Modi).

Seat-Typen

Typ	Inhalt
Büro-Seat	Vollzugang zur Plattform, Default-UI-Modus „Büro", alle Module zugreifbar gemäß Permissions
Monteur-Seat	Eingeschränkter Zugang, Default-UI-Modus „Baustelle", reduzierte Modul-Sicht (Projekt-Detail, Tasks, Acceptance, Foto-Forms, Chat in eigenen Channels, HR-Hub eigene Doks) — siehe M27 C.2 Monteur-Rolle

Pflicht-Verknüpfung mit Rolle

- Jeder Tenant-User hat **genau einen Seat-Typ** zur aktiven Zeit
- Bei User-Einladung (M27 D.1) wählt Tenant-Admin Seat-Typ als Pflichtfeld
- **Seat-Typ ist Lizenz-Konzept**, Rolle ist Permission-Konzept — beide separat
- Ein Monteur-Seat kann **nur** Rollen mit Default-UI-Modus `Baustelle` zugewiesen bekommen (Validierungs-Regel)
- Ein Büro-Seat kann jede Rolle bekommen

Seat-Counting + Stripe-Sync

- Bei Einladung/Aktivierung eines neuen Users: Seat-Counter hochzählen, Stripe-`Subscription.items[].quantity` updaten
- Bei Deaktivierung/Entfernung: Seat-Counter runter, Stripe-Update zum Period-Ende (proration nach Tenant-Admin-Setting)

- **Übersicht in** `/{tenant}/admin/usage/` **:** aktuelle Seats pro Typ + verfügbare Seats laut Plan + Trend-Chart

Plan-Quotas

`SubscriptionProductFeature.quota_value` enthält (Beispiel):

```
{
  "max_buero_seats": 10,
  "max_monteur_seats": 50,
  "max_buero_seats_overage_blocked": false
}
```

Bei Erreichung Quota:
- **Hard-Limit** (Default): Einladung blockiert, Klartext: *„Plan-Limit von 10 Büro-Seats erreicht. Upgrade nötig — siehe Marketing-Site."*
- **Soft-Limit** (Konfiguration): zusätzliche Seats werden automatisch über Usage-Based-Price abgerechnet

---

Block E — Trial-Phase

Eigenschaften

- **Dauer:** 5-7 Tage (per `SubscriptionProduct.trial_default_days`)
- **Vertriebs-freigegeben:** Vertrieb (Plattform-Admin) muss Trial **manuell aktivieren** (`requires_manual_activation=true` für Phase-1-Trials), um Spam zu verhindern
- **Voller Funktionsumfang während Trial** (kein Read-Only)
- **Stripe:** Subscription mit `trial_end` gesetzt, keine Zahlungs-Auslösung vor Trial-Ende
- **Zahlungsmethode: nicht** zwingend zu Trial-Beginn (Vertriebs-Modus). Wird in Phase 2 für Self-Service-Trials erforderlich (User-Diskussion auslagerbar)

Trial-Aktivierungs-Flow

1. User schließt Signup ab → `TenantSubscription.status = trial`, aber `trial_start = null`
2. Tenant erscheint in Plattform-Admin-Liste „Trial-Anfragen" (M37)
3. Vertrieb prüft + aktiviert → setzt `trial_start = now()`, `trial_end = now() + trial_default_days`
4. Tenant erhält Klartext-Email: *„Dein HiveGrid-Trial ist freigeschaltet. Bis [Datum] kannst du alles testen."*

Trial-Ende-Flow

- 3 Tage vor Trial-Ende: In-App-Banner + Email-Notification mit Klartext und Hinweis auf Marketing-Site (kein In-App-Upgrade — LP-8)
- 1 Tag vor Trial-Ende: erneute Notification
- Trial-Ende ohne Konversion: `TenantSubscription.status = cancelled`, Tenant geht in `deletion_pending` (30 Tage Grace, M27 A.5)

Trial-Konversion

Tenant-Admin geht auf Marketing-Site (`hivegrid.de/upgrade`), wählt Plan, gelangt zurück nach `app.hivegrid.de/{tenant}/admin/subscription/checkout-return` mit Stripe-Session-Token → CheckoutSessionStatus + Webhook-Verarbeitung → `active`-Übergang.

Block F — Dunning + Past-Due

Stripe-Standard-Dunning

- **3 Wiederholungs-Versuche** über **14 Tage** (Stripe Smart Retries oder eigene Schedule)
- Versuche typischerweise: Tag 0 (initial), Tag 3, Tag 7, Tag 14
- HiveGrid empfängt `invoice.payment_failed` und `invoice.payment_succeeded`-Events

Past-Due-Verhalten

- Workspace **read-only** (siehe Block G)
- In-App-Banner: *„Zahlung fehlgeschlagen, Konto in 14 Tagen gesperrt — [Zahlungsmethode aktualisieren]"* mit Direkt-Link zum Stripe-Customer-Portal
- Email-Notification an Tenant-Admins mit Klartext + Direkt-Link
- Nach 3 Fehlversuchen + 14 Tagen ohne Erfolg: `cancelled`

Notification-Strategie

Event	Empfänger	Kanal
Erste Zahlungs-Fehler	Tenant-Admins	Email + In-App
Versuch 2/3 fehlgeschlagen	Tenant-Admins	Email + In-App
1 Tag vor finalem Cancel	Tenant-Admins	Email + In-App + Push (M22)
Final-Cancel	Tenant-Admins + Plattform-Admin	Email + Plattform-Admin-Alert

---

Block G — Downgrade-Read-Only-Middleware

Wirkung

Bei Status `past_due` ODER `frozen` ODER `cancelled` (mit `cancel_at_period_end=true` und Period bereits abgelaufen):

- App **rendert** noch alle Daten lesend
- **Schreibende Aktionen blockiert** auf Middleware-Ebene
- Schreibende API-Endpoints liefern 402 Payment Required mit Klartext-Body
- Schreibende UI-Elemente (Buttons, Forms) entweder ausgeblendet oder ausgegraut mit Tooltip

Genaue Read/Write-Grenzen (analog M27 A.5 Frozen-Modus)

Aktion	`past_due` / `frozen` / `cancelled-period-over`
Login (alle Rollen)	✓
Daten lesen	✓
Stammdaten-Edit, neue Datensätze	✗
Email/WhatsApp/Push senden	✗
Automationen ausführen	✗ (eingehende Trigger werden verworfen, nicht gequeued)
Eingehende Webhooks	✗ (mit 402-Response, nicht gequeued)
Customer-Portal-Zugang (Endkunden)	✓ (read-only)
API (Read)	✓
API (Write)	✗ (402)
Stripe-Customer-Portal-Zugang	✓ (Pflicht zur Reaktivierung)
Plan-Wechsel / Reaktivierung	✓ (einziger erlaubter Schreib-Pfad)

Implementierungs-Vorgabe

- Middleware **vor** Route-Loader: prüft `TenantSubscription.status` + `is_writable_status()`-Helper
- Middleware **vor** Worker-Job-Ausführung: blockiert Automation-Worker, Notification-Worker, Webhook-Outbound-Worker bei nicht-schreibendem Status
- Cache der Status-Auflösung: 30 Sekunden (TTL), bei Stripe-Webhook explizit invalidieren

Klartext-Hinweise

- In-App-Banner pro Status (siehe M27 A.5)
- Bei Klick auf blockierten Schreib-Button: Toast *„Workspace ist read-only. Status: [Klartext]. [Reaktivieren]*"

Block H — Frozen-Modus (User-Pause)

Use-Case

Tenant kann Workspace pausieren statt zu kündigen (z.B. Saison-Pause bei kleinen Betrieben). Reduzierte Gebühr, Workspace read-only, jederzeit reaktivierbar — Daten bleiben.

Pause-Flow

1. Tenant-Admin in `/{tenant}/admin/subscription/` → „Workspace pausieren"
2. Klartext-Erklärung: was bleibt, was nicht geht, welche Gebühr
3. Bestätigung mit Tipp-Eingabe „PAUSIEREN" (Multi-Step-Schutz)
4. Stripe: Subscription-Wechsel auf `frozen_plan_price` (eigener `SubscriptionPrice` mit reduzierter Gebühr)
5. `TenantSubscription.status = frozen`, `frozen_since = now()`
6. M27 Tenant-Status synchronisiert auf `frozen`

Reduzierte Gebühr

- Konfigurations-Setting (`SubscriptionProduct` hat optionalen `frozen_mode_price_id`)
- Phase 1: Default ca. **1/10** des Standard-Plan-Preises (konkrete Preise im Pricing-Setup, **geparkt**)

Reaktivierungs-Flow

1. User klickt In-App-Banner „Reaktivieren"
2. Stripe-Checkout-Session mit `intent=reactivate_from_frozen` und gewähltem Standard-Plan
3. Nach Erfolg: `frozen → active`-Übergang

Sonderfälle

- Frozen-Tenant kann sich **nicht** in den anderen End-Status (`cancelled`) wechseln, ohne erst zu reaktivieren — *außer* expliziter Kündigung mit Klartext-Hinweis
- Bei Zahlungsausfall der reduzierten Gebühr: `frozen → past_due`

Block I — Service-Metering

Metriken (Phase 1)

Engine baut bereits Tracking auf, **konkrete Pricing-Schwellen geparkt**:

Metrik-Code	Was wird gezählt
`automation_trigger`	Anzahl Trigger-Auslösungen in der Automation-Engine pro Tenant pro Period (siehe Modul 17 EC-1 Hard-Limit)
`ai_token`	LLM-Token-Verbrauch (siehe Block J) — Prompt-Token + Completion-Token pro Provider, normalisiert
`sms_outbound`	SMS-Versand falls Modul 22 aktiv (Phase 2)
`whatsapp_session`	WhatsApp-Session-Kosten (Phase 2)

Tracking-Pflichten

- Jede Automation-Trigger-Ausführung: `TenantSubscriptionUsageRecord` inkrementieren (idempotent über Trigger-Run-ID)
- Jeder LLM-Call: Token-Verbrauch nach Response loggen, in `Credit`-Verbrauch verrechnen
- Aggregation pro **Period** (entspricht Stripe-Subscription-Period — typisch monatlich)

Reporting an Stripe

- Bei `metric_code` mit `is_metered=true`-Subscription-Item: Usage-Reporting an Stripe via `usage_records.create` am Period-Ende oder real-time (Strategie pro Metrik konfigurierbar)
- `reported_to_stripe_at`-Feld für Tracking

In-App-Sicht (`/{tenant}/admin/usage/`)

Zu zeigende Daten:
- Aktuelle Period mit Verbrauch pro Metrik
- Plan-Inklusiv-Menge pro Metrik
- Verbrauch in % (Progress-Indikator)
- Trend gegenüber Vorperiode
- Historische Perioden (letzte 12)

Benötigte UI-Elemente:
- Listen-/Tabellen-Anzeige der Metriken
- Detail-Drill-Down pro Metrik mit Zeitachsen-Visualisierung
- Filter nach Period
- Export der Verbrauchs-Daten als CSV

Operationen:
- Reine Lese-Sicht — keine Verbrauchs-Änderungen
- Add-On-Käufe (Trigger-Volumen-Pakete) → **Phase 2**

Permissions: `billing.read` oder `tenant.admin`.

Block J — AI-Credits-System

Konzept

Pro Plan inkludierte AI-Token-Quota (z.B. Cloud: 1 Mio Token/Monat — konkrete Mengen geparkt).

Saldo-Berechnung

- Saldo aus `Credit`-Tabelle (Sum aller Einträge eines `credit_type`)
- Bei Plan-Period-Start: Plan-Inclusion als positiver `Credit` mit `valid_until = period_end`
- Bei jedem LLM-Call: negativer `Credit`-Eintrag mit Token-Anzahl

Verbrauchs-Verrechnung

1. LLM-Call ausgelöst via Modul 18
2. Pre-Check: Saldo > erwarteter Maximal-Verbrauch?
   - Ja → Call ausführen
   - Nein → Klartext-Fehler: *„AI-Credits aufgebraucht. Plan-Wechsel oder Add-On nötig — siehe Marketing-Site."*
   (Apple-Compliance-konform — kein In-App-Upgrade)
3. Nach Call: Token aus Response-Metadata, negativer Credit-Eintrag

Carry-Over-Regeln (Phase 1)

- Plan-Inclusion **verfällt** am Period-Ende (`valid_until` enforced)
- Manual-Grants (Vertrieb gibt Bonus) ohne `valid_until` bleiben
- Promo-Codes mit fixem `valid_until`

In-App-Sicht (`/{tenant}/admin/usage/ai-credits/`)

Zu zeigende Daten:
- Aktueller Saldo pro `credit_type`
- Verbrauchs-Historie mit Quelle (welche User, welcher Modul-Aufruf)
- Hochrechnung: bei aktuellem Verbrauchs-Trend wann wäre Saldo aufgebraucht

Benötigte UI-Elemente:
- Saldo-Anzeige
- Verbrauchs-Liste mit Filter
- Trend-Chart

Permissions: `billing.read`.

Block K — Setup-Fee-Handling

Anwendung

- Cloud-Plan: Setup-Fee einmalig
- Cloud+-Plan: Setup-Fee einmalig, höher (Pricing geparkt)

- Enterprise: individuell

Konfiguration

- `SubscriptionProduct.setup_fee_required = true` aktiviert Pflicht-Setup-Fee
- Separater `SubscriptionPrice` mit `interval=one_time`
- Im Stripe-Checkout: zwei Items (recurring + one_time)

Tracking

- `TenantSubscription.setup_fee_paid_at` wird gesetzt bei `invoice.payment_succeeded` der ersten Rechnung
- Vor Zahlung: Workspace ist im Trial- oder Pre-Active-Modus (vollfunktional, weil Trial)
- Falls Setup-Fee nicht gezahlt aber Subscription aktiv (Edge-Case): wie `past_due` behandeln

Plan-Wechsel von Cloud → Cloud+

- Falls Setup-Fee schon für Cloud bezahlt: nur Differenz berechnen
- Logik: `effective_setup_fee = neuer_plan.setup_fee - bisherige_setup_fee_summe` (clamped >= 0)

Block L — Demo-Buchung (extern)

Konzept

Enterprise-Anfragen → Marketing-Site bietet Demo-Buchung über externes Tool (Calendly o.ä.). **Nicht** in HiveGrid integriert.

Pflicht-Properties

- Marketing-Site `hivegrid.de/demo` mit Calendly-Embed
- Vertrieb erhält Termin, führt Demo, legt **manuell** Tenant über `admin.hivegrid.de` an (M37)
- Tenant erhält Initial-Login-Email mit Magic-Link (M27 B.2)
- `TenantSubscription.created_via = platform_admin`

Was nicht in dieses Modul gehört

- Demo-Booking-Engine (das ist M6 Booking — wird aber **nicht** für Demo-Termine genutzt; M6 ist tenant-internal für Endkunden-Termine)

Block M — Routes + In-App-Sicht

Sidebar-Eintrag

Sidebar-Hauptpunkt **Admin** (Workspace-Admin, siehe M27 A.3) → Sub-Eintrag „Subscription".

Routes

```
/{tenant}/admin/subscription/                  → Subscription-Übersicht
/{tenant}/admin/subscription/plan-info/        → Plan-Details (Read-Only)
/{tenant}/admin/subscription/checkout-return/  → Stripe-Checkout-Return-Handler
/{tenant}/admin/usage/                         → Verbrauchs-Übersicht (Trigger, AI-Token, Seats)
/{tenant}/admin/usage/ai-credits/              → AI-Credits-Detail
/{tenant}/admin/billing/                       → Stripe-Customer-Portal-Embed (Rechnungen +
Zahlungsmethode)
```

`/admin/subscription/` — Übersicht

Zu zeigende Daten:
- Aktueller Plan-Name + Status (Klartext)
- Period-Start + Period-Ende
- Trial-Status falls aktiv
- Seat-Übersicht pro Typ
- Inklusiv-Quotas pro Metrik mit aktuellem Verbrauch

Benötigte UI-Elemente:
- Read-Only-Anzeige aller Plan-Eigenschaften (KEIN „Upgrade"-Button — LP-8)
- Hinweis-Box: *„Plan-Wechsel über deinen Account-Manager oder hivegrid.de"* mit Link nach extern (öffnet in neuem Tab)
- Pause-Button (Frozen-Modus, siehe Block H)
- Kündigen-Button

Operationen:
- Workspace pausieren (Block H)
- Tenant kündigen (M27 A.5)
- Status anzeigen

`/admin/usage/`

Siehe Block I.

`/admin/billing/`

Stripe-Customer-Portal-Embed (Stripe-gehostet, IFrame oder Redirect):
- Rechnungs-Historie (PDF-Download)
- Zahlungsmethode aktualisieren
- Steuer-Daten / USt-Id editieren

Permission: `billing.read` für Anzeige, `tenant.admin` für Zahlungsmethoden-Änderung.

Was NICHT in der App ist (LP-8)

- Pricing-Tabelle / Plan-Vergleich
- Upgrade-Buttons
- Add-On-Käufe (Phase 2 evtl. via Marketing-Site)
- Promotional-Banner für höhere Plans
- Plan-Wechsel-UI

Block N — Permissions-Anwendung

Permission-IDs

Permission	Erlaubt
`billing.read`	Subscription-Übersicht, Usage, Rechnungen einsehen
`tenant.admin`	Plan-Wechsel, Pause/Reaktivierung, Kündigung, Zahlungsmethoden, Seat-Management

Bundle-Mapping (Variante 2)

Bundle	Atomare Permissions
`billing.none`	(keine)
`billing.read`	`billing.read`
`billing.admin`	`billing.read` + `tenant.admin`-Sub-Permissions für Subscription

Anwendung in Routes

- `/{tenant}/admin/subscription/` → `billing.read` oder `tenant.admin`
- `/{tenant}/admin/usage/` → `billing.read` oder `tenant.admin`
- `/{tenant}/admin/billing/` → `billing.read` oder `tenant.admin`
- Pause-/Kündigen-/Reaktivierung-Aktion → `tenant.admin`

Anwendung in Action-Buttons

- Workspace pausieren: `tenant.admin`
- Workspace reaktivieren: `tenant.admin`
- Tenant kündigen: `tenant.admin` (zusätzlich Last-Tenant-Admin-Schutz, M27 D.4)
- Verbrauchs-Daten exportieren: `billing.read`

Plattform-Admin-Sicht (M37)

- Sicht auf alle `TenantSubscription`-Datensätze
- Manuelle Trial-Aktivierung (`requires_manual_activation` Flow)
- Manuelle Subscription-Anlage für Enterprise
- Manuelle Credit-Grants (Bonus / Promo-Codes)
- Permission: `platform.billing.admin`

Block O — Klartext-Hinweise/-Fehler

Pflicht-Texte (LP-10 + AV #1):

- **Trial-Aktivierungs-Email:** *„Dein HiveGrid-Trial ist freigeschaltet. Bis [Datum] kannst du alles testen. Bei Fragen: [Support-Link]"*
- **Trial-Ende-3-Tage-Warning:** *„Dein Trial endet in 3 Tagen am [Datum]. Plan wählen unter hivegrid.de"*
- **Trial-abgelaufen:** *„Dein Trial ist abgelaufen. Workspace ist read-only. Plan wählen unter hivegrid.de"*
- **Past-Due-Initial:** *„Zahlung gescheitert für Rechnung vom [Datum]. Wir versuchen es in [N] Tagen erneut. [Zahlungsmethode aktualisieren]"*

- **Past-Due-Final:** *„Letzter Zahlungsversuch fehlgeschlagen. Konto wird in 24h gesperrt. [Zahlungsmethode aktualisieren]"*
- **Frozen-Banner:** *„Workspace pausiert seit [Datum]. Reaktivieren um wieder zu schreiben."*
- **Read-Only-Klick auf Schreib-Button:** *„Diese Aktion ist nicht möglich, weil dein Workspace pausiert ist. [Reaktivieren]"*
- **Seat-Quota-erreicht:** *„Plan-Limit von [N] [Typ]-Seats erreicht. Plan-Wechsel über hivegrid.de"*
- **AI-Credits-aufgebraucht:** *„AI-Credits für diese Period verbraucht. Verfügbar wieder am [Period-Start]. Plan-Wechsel über hivegrid.de"*
- **Trigger-Volumen-Limit:** *„Automation-Trigger-Limit erreicht für diese Period. Eingehende Trigger werden verworfen bis [Period-Start]. Plan-Wechsel über hivegrid.de"*
- **Stripe-Webhook-Verarbeitungsfehler:** Plattform-Admin-Alert mit Klartext-Stack-Trace
- **Plan-Wechsel-Confirmation-Email:** *„Plan-Wechsel auf [Plan-Name] wirksam ab [Datum]. Differenz: [Betrag] EUR."*
- **Setup-Fee-Hinweis im Checkout:** *„Bei Buchung wird zusätzlich eine einmalige Setup-Gebühr von [Betrag] berechnet."*

Fehler-Cases

Fehler	Klartext + Lösungs-Hinweis
Stripe-Checkout-Session abgelaufen	*„Die Bezahl-Session ist abgelaufen. Bitte Plan-Wahl wiederholen."* + Direkt-Link
Stripe-Customer-Portal nicht erreichbar	*„Die Rechnungs-Verwaltung ist gerade nicht erreichbar. Bitte später erneut versuchen oder [Support-Link]"*
Plan-Wechsel auf nicht existierenden Plan	*„Der gewählte Plan ist nicht mehr verfügbar. Bitte aktuelle Plan-Liste auf hivegrid.de wählen."*
Setup-Fee-Zahlung fehlgeschlagen, Subscription aktiv	Plattform-Admin-Alert + Tenant-Admin-Klartext: *„Setup-Gebühr-Zahlung gescheitert. Bitte Zahlungsmethode prüfen — sonst wird Konto in 14 Tagen gesperrt."*
Webhook-Signature ungültig	401-Response, Plattform-Admin-Alert (möglicher Angriff)
Tenant kündigt während Trial	Klartext: *„Dein Trial wurde beendet. Workspace wird in 30 Tagen gelöscht — [Zurückziehen]"*

Block P — Phase 2 / Backlog

Bewusst NICHT in Phase 1 (LP-8 + Apple-Compliance):

- **In-App-Subscription-Self-Service** (Plan-Wechsel direkt in der App ohne Marketing-Site-Umweg) — wartet auf App-Store-Compliance-Klärung
- **Add-On-Käufe** (zusätzliche Trigger-Volumen-Pakete, AI-Credits-Top-Ups) — Marketing-Site oder Phase-2-In-App
- **Multi-Tenant-Billing** für Konzerne (mehrere Tenants in einer Stripe-Subscription gebündelt)
- **Custom-Pricing-Tier** per Plattform-Admin (individuelle Preisverhandlung Enterprise)
- **Promo-Code-System** (Discount-Codes im Checkout)
- **Annual-Billing-Toggle** mit konfigurierbarem Jahres-Rabatt
- **Self-Service-Trial ohne Vertriebs-Freigabe** (`requires_manual_activation = false`-Modus, mit Zahlungsmethoden-Pflicht)
- **Reseller-Modell** (Agentur kauft Subscription für Endkunde-Tenant, eigene Margin)
- **Usage-Forecasting** (KI-basiert: „bei aktuellem Verbrauch wirst du den Plan in 7 Tagen sprengen")
- **Self-Service-Cancellation-Survey** mit Auto-Discount-Angebot

---

Architektur-Vorschriften (Querschnitt)

- **AV #6 — Audit-Trail-Default:** Jede Subscription-Status-Änderung, jeder Plan-Wechsel, jede Credit-Veränderung geloggt mit Diff
- **AV #8 — Idempotenz pro Webhook-Verarbeitung:** Stripe-Events nicht doppelt verarbeiten (über `stripe_event_id`)
- **AV #5 — Permissions-aware-Default:** Listen / Verbrauchs-Sicht filtert nach `billing.read` / `tenant.admin`
- **LP-8 — Apple-Compliance:** keinerlei Pricing-/Upgrade-UI in App; alle externen Links öffnen in neuem Browser-Fenster mit `noopener noreferrer`
- **LP-10 + AV #1 — Klartext-Diagnostik:** alle Subscription-/Billing-Fehler menschenlesbar mit Lösungs-Hinweis
- **LP-13 — EU-Stack:** Stripe-Vertragspartner ist Stripe Payments Europe DAC (Ireland), DPA elektronisch unterzeichnet
- **AV #11 — Read-Only-Middleware** strukturell für alle Schreib-Pfade (Routes, API, Worker)

Detail-Spec-Status: DRAFT

Erste vollständige Detail-Spec für Modul 28. **Vor Bau:** Verifizierung mit Noah + Pricing-Modell-Konkretisierung (separate Runde). Konkrete Stripe-Price-IDs, Webhook-Endpoint-URLs, EU-Entity-Vertragsdetails kommen mit Implementierung.

Offene Detail-Punkte (nicht-blockierend):
- Konkretes Pricing-Modell (Cloud/Cloud+/Enterprise-Preise, Trigger-Volumen-Schwellen, AI-Token-Mengen, Setup-Fee-Beträge, Frozen-Mode-Preis, Annual-Discount-%) — geparkt für eigene Runde
- Konkretes Branchen-Plan-Mapping (alle Branchen können alle Pläne, oder Branchen-spezifische Trial-Tage?)
- Genaue Stripe-Smart-Retries-Konfiguration vs. eigene Schedule
- Edge-Case bei Plan-Downgrade mid-period (Pro-rata vs. Period-End)
- Migration für Bestandskunden (Geerkens beim Go-Live) — Pricing-Modell-Festlegung erforderlich
- Konkrete Permission-IDs zusammen mit M27-Permission-Konsolidierung
- Subframe-UI-Designs (separat)