```
module: 33a
title: Apps-Engine / Adapter-Layer
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-9, LP-10, LP-11]
spec_style: requirements-pur
```

Modul 33a — Apps-Engine / Adapter-Layer

> ***Detail-Spec gemäß LP-9.*** *Foundation für alle externen Integrationen (LP-11). Wenn das sauber ist, hängen sich KWP, AirCall, Gmail, Stripe, Calendar konsistent dran.*

Themenblöcke

```
A. Adapter-Pattern-Architektur
B. Datenmodell (App, DataConnection, AppSyncRun, AppSyncRunLog, AppSyncItem)
C. Adapter-Registry (Code-Konvention)
D. App-Lifecycle (Connect, Token-Refresh, Reconnect, Disconnect)
E. Sync-Mechanik (Push/Pull/Webhook/Manual)
F. Einheitliche Settings-UI
G. Klartext-Diagnostik / Connection-Health
H. Permissions
I. Phase 2 (Community-Marketplace, Templates-Marketplace)
```

Block A — Adapter-Pattern-Architektur

Kern-Idee

Jede externe Integration ist eine **App**. Apps implementieren ein gemeinsames Adapter-Interface. Ergebnis: einheitliche Settings-UI, Sync-Logs, Klartext-Diagnostik, Re-Sync-Mechanik, Audit-Trail-Integration.

Adapter-Interface (Implementierungs-Konzept)

Jede App-Implementierung liefert:

Connection-Methoden:
- `connect(tenantId, credentials)`
- `disconnect(tenantId, connectionId)`
- `refreshTokens(connection)` (für OAuth)
- `validateConnection(connection)` — Health-Check, gibt Status (`healthy | degraded | broken`) + Klartext-Hinweis zurück

Sync-Methoden:
- `syncPull(connection, since)` — Cursor-basierter Pull
- `syncPush(connection, items)` — Push mit Idempotency-Keys
- `handleWebhook(connection, payload)` — Webhook-Eingang verarbeiten

Capability-Deklaration:
- `capabilities[]` — z.B. `pull_mails` , `send_mail` , `create_invoice` , `webhook_inbound`

- `triggers[]` — Automation-Trigger-Typen (M17)
- `actions[]` — Automation-Actions (M17)

Connection-Schema:
- JSON-Schema der erforderlichen Credentials (für UI-Generierung)
- Connection-Type (`oauth2 | api_key | username_password | windows_client | custom`)

Klartext-Übersetzungen:
- Pro Fehler-Code: Klartext-Text + Lösungs-Hinweis (LP-10)

Block B — Datenmodell

App (Plattform-weit, nicht tenant-scoped)

- `id` , `key` (z.B. `kwp` , `gmail` , `aircall` , `pandadoc`)
- `name` , `description`
- `category` (`erp | communication | calendar | signature | payment | productivity | ai`)
- `connection_type` (`oauth2 | api_key | windows_client | custom`)
- `connection_schema` (JSON-Schema)
- `capabilities[]`
- `phase` (`phase_1 | phase_2 | beta | deprecated`)
- `requires_subscription_feature` (optional — z.B. KWP nur Cloud+)
- `documentation_url`

App-Tabelle ist Spiegelung der Code-Registry für UI/Connection-Verwaltung.

DataConnection

- `id` , `tenant_id` , `app_id` , `created_at` , `updated_at` , `created_by_user_id`
- `name` (User-Label, z.B. „Geerkens Hauptbüro KWP")
- `credentials` (JSON, **verschlüsselt at-rest** mit Tenant-Schlüssel)
- `connection_status` (`pending | active | degraded | broken | revoked`)
- `last_health_check_at` , `last_health_check_result`
- `last_sync_at` , `next_sync_at`
- `last_error` (Klartext bei `degraded` / `broken`)
- `metadata` (App-spezifisch: Token-Ablauf, Mailbox-ID, etc.)

Mehrere DataConnections pro App pro Tenant möglich (z.B. drei Gmail-Mailboxen, zwei KWP-Server).

AppSyncRun

- `id` , `tenant_id` , `data_connection_id`
- `kind` (`pull | push | webhook | health_check | manual_resync`)
- `started_at` , `finished_at`
- `status` (`running | success | partial_success | failed`)
- `items_total` , `items_succeeded` , `items_failed`
- `summary_text` (Klartext: *„Sync mit KWP — 14 Aufträge gepullt, 2 fehlgeschlagen wegen veraltetem Schema"*)
- `triggered_by_user_id` (nullable)

AppSyncRunLog

- `id` , `app_sync_run_id`
- `timestamp` , `severity` (`debug | info | warn | error | critical`)
- `message_text` (Klartext)

- `tech_details` (JSON — für Plattform-Admin)

AppSyncItem

- `id`, `app_sync_run_id`
- `entity_type`, `entity_id` (HiveGrid-Entität, optional)
- `external_id`
- `operation` (`create | update | delete | upsert | noop`)
- `status` (`success | skipped | failed`)
- `error_text` (Klartext bei Fehler)

Block C — Adapter-Registry (Code-Konvention)

Verzeichnis-Struktur

```
app/integrations/{app-key}/
  ├── index.ts          (Adapter-Implementierung)
  ├── manifest.ts       (App-Metadata)
  ├── i18n/
  |   └── de.json       (Klartext-Übersetzungen)
  ├── setup.tsx         (Setup-Komponente)
  └── README.md         (Implementierungs-Doku)
```

Build-Time-Registrierung

Beim App-Start scannt Server `app/integrations/`, alle Manifests werden in DB synchronisiert. Drift-Check: bei Inkonsistenz → Klartext-Warnung im Plattform-Admin.

Tenant-Sicht

User sieht nur Apps mit:
- `phase = phase_1` (alle Tenants)
- `phase = beta` UND Tenant hat Feature-Flag
- `requires_subscription_feature = null` ODER Tenant hat das Feature

Block D — App-Lifecycle

Connect-Flow (pro Connection-Type unterschiedlich, aber UI einheitlich)

- **OAuth2** (Gmail, Microsoft, Google Calendar): Redirect-Flow → Auto-Token-Speicherung → Health-Check
- **API-Key** (PandaDoc, Stripe, Mistral): User trägt Key in Form ein → Validation → Speicherung
- **Username/Password** (Legacy): User trägt Credentials → Validation
- **Windows-Client** (KWP): User lädt `kwpsync.exe` aus Setup-Wizard, konfiguriert lokal Server-DB-Connection-String, kwpsync.exe registriert sich beim Cloud-API mit Token

Setup-Wizard pro App

1. Voraussetzungen-Check (z.B. *„Du brauchst Admin-Zugang zur KWP-DB. Bestätigen."*)
2. Connection-Schema-Formular (auto-generiert aus `connection_schema`)
3. Validation-Call → Health-Check
4. Initial-Sync starten (optional)

Token-Refresh

- OAuth2-Tokens werden automatisch refreshed (Background-Worker checkt `metadata.token_expires_at`)
- Bei Refresh-Fehler: `connection_status = degraded` + Klartext-Notification an Tenant-Admin

Reconnect

- Bei `degraded` / `broken`: Re-Connect-Funktion öffnet wieder Setup-Wizard
- Re-Connect erhält dieselbe `DataConnection`-ID (kein History-Verlust)

Disconnect

- Multi-Step-Confirm
- Cleanup: Pending-Sync-Items abbrechen, Webhook-Subscriptions kündigen, OAuth-Tokens revoken
- `DataConnection` **nicht hard-gelöscht** — Status `revoked`, History bleibt für Audit
- Klartext: *„Beim Trennen werden alle Sync-Jobs gestoppt. Eingehende Webhook-Events werden nicht mehr verarbeitet. Bestätigen?"*

Block E — Sync-Mechanik

Sync-Modi

(1) Pull-Sync (Cron-basiert): Regelmäßiges Abrufen externer Daten (Gmail alle 5 Min, KWP alle 30 Sek). Cursor-basiert. Lock-Mechanismus: nur ein Pull-Sync pro Connection gleichzeitig.

(2) Push-Sync (Event-basiert): HiveGrid-Event triggert Push (z.B. „Auftrag fertig" → KWP-Push). Idempotenz pro Item (siehe M17 EC-3).

(3) Webhook-Sync (Inbound): Externe API pusht zu HiveGrid. Endpoint: `app.hivegrid.de/api/v1/webhooks/{app_key}/{tenant_id}/{connection_id}` mit Signatur-Validation. Webhook-Payload wird in `AppSyncRun` mit `kind=webhook` geloggt.

(4) Manueller Re-Sync: Tenant-Admin triggert *„Jetzt syncen"*. Force-Pull oder selektiver Subset. Klartext-Progress-Anzeige bei langen Resyncs.

Schema-Drift-Sentinel (PM-4)

- Optional `validateSchema()`-Methode pro App
- Adapter checkt periodisch externes Schema
- Bei Abweichung: `connection_status = degraded` + Klartext + Push-Alert an Plattform-Admin (M37)
- Tenant-User sieht nur Klartext-Status

Loop-Guard / Anomalie-Detection (EC-1)

- Webhook-Storm-Schutz: konfigurierbare Anzahl Webhooks pro Stunde, Rest verworfen mit Notification
- Anomalie: Sync-Volumen 10x normal → Pause + Klartext

Block F — Einheitliche Settings-UI

`/{tenant}/admin/integrations/`:

Zu zeigende Daten:
- Liste aller verfügbaren Apps (Catalog), gruppiert nach Kategorie

- Pro App: Name, Beschreibung, Connection-Status, Anzahl aktive Connections
- Filter: Kategorie, Status, Phase

Benötigte UI-Elemente:
- App-Catalog-Liste (Suche + Filter)
- Pro App: Detail-Funktion mit Connection-Verwaltung
- Pro DataConnection:
- Status-Indikator (grün `active` / gelb `degraded` / rot `broken` / grau `revoked` / `pending`)
- Klartext-Status-Hinweis
- Letzter Sync-Zeitpunkt
- Manuelle Re-Sync-Funktion
- Sync-History (Liste der `AppSyncRun`s mit Klartext)
- Reconnect-Funktion bei `degraded` / `broken`
- Disconnect-Funktion

Operationen:
- App connecten (Setup-Wizard)
- Connection editieren (Credentials updaten)
- Manueller Re-Sync
- Connection disconnecten
- Sync-History einsehen (Filter nach Datum/Status/Operation)

Permission: `tenant.admin` ODER `integrations.admin`.

Block G — Klartext-Diagnostik / Connection-Health

LP-10 zentral umgesetzt. Jede App liefert in i18n-Datei pro Fehler-Code Klartext-Text + Lösungs-Hinweis.

Health-Check-Intervalle

- Aktive Connection: alle 5 Min
- `degraded`: 1 Min (schneller Recovery)
- `broken`: 15 Min (weniger Last)

Status-Bedeutungen

Status	Bedeutung	Indikator
`pending`	Setup nicht abgeschlossen	grau
`active`	Alles okay	grün
`degraded`	Funktioniert teilweise	gelb
`broken`	Funktioniert gar nicht	rot
`revoked`	Manuell disconnectet	grau

Klartext-Beispiele

- *„Verbindung zu KWP läuft. Letzte erfolgreiche Sync vor 4 Min."* (active)
- *„KWP-Verbindung verlor Token-Aktualität. Bitte in den Settings den Verbindungs-Token erneuern. [Reconnect]"* (degraded)
- *„Verbindung zu KWP nicht erreichbar. Mögliche Ursachen: kwpsync.exe nicht gestartet, Netzwerk unterbrochen, Server-Update läuft. [Anleitung] [Reconnect]"* (broken)

- *„KWP-Update erkannt — manche Felder wurden geändert. Sync pausiert für Sicherheit. Wir sind informiert und beheben das. [Plattform-Admin kontaktieren]"* (Schema-Drift)

Notification-Strategie

- `degraded`: Email + In-App-Notification an Tenant-Admin nach 30 Min Persistenz
- `broken`: sofortige Email + Push + In-App
- Recovery zu `active`: Klartext-Bestätigung *„KWP-Verbindung ist wieder aktiv."*

Block H — Permissions

Permission	Erlaubt
`integrations.read`	Liste der Connections sehen, Status sehen
`integrations.connect`	Neue Connection anlegen
`integrations.disconnect`	Connection trennen
`integrations.manual_sync`	Manuellen Re-Sync triggern
`integrations.view_logs`	Sync-Logs einsehen
`integrations.admin`	Voll-Admin

Default-Rollen-Mapping

Rolle	Permission
GF	`integrations.admin`
Vertriebsleiter	`integrations.read`
Büro	`integrations.read` + `integrations.manual_sync`
Andere	`integrations.read`

Block I — Phase 2 (Marketplace)

M33b CommunityApp-Marketplace:
- Drittanbieter bauen Apps für HiveGrid
- Tenants installieren aus Marketplace
- Sandboxing + Review-Prozess
- `CommunityApp`, `CommunityAppInstall`

M33c Template-Marketplace:
- Tenants kaufen vorgefertigte Templates (Pipelines, Forms, Automation-Workflows, Dashboard-Layouts)
- `TemplateMarketplaceDefinition`, `TemplateMarketplacePurchase`, `TemplateMarketplaceInstall`

Beide Phase 2/3 — kein Geerkens-Bedarf.

Architektur-Vorschriften (Querschnitt)

- **LP-10, AV #1:** Klartext-Diagnostik zentral hier umgesetzt
- **LP-11, AV #2:** Adapter-Pattern als Architektur-Vorschrift für alle Integrations-Module
- **AV #6:** Audit-Trail für Connection-Änderungen
- **AV #18:** Heartbeat + Auto-Recovery (Spezialfall windows_client/KWP)
- **AV #19:** Persistente Queue mit Timestamps + Rate-Limited-Replay (KWP-Spezial)

Detail-Spec-Status: DRAFT

Modul 33a komplett spec'd. **Foundation für alle Integrations-Module** (M9, M12, M19, M25, M28).

Offene Detail-Punkte:
- Konkrete Credential-Encryption-Strategie (KMS / Tenant-Key-Derivation)
- Webhook-Signature-Validation pro App-Type
- Backoff-Strategie für Sync-Retries (Exponential / Linear)
- Multi-Connection-UX-Detail (wie zeigt UI mehrere Gmail-Mailboxen pro Tenant?)