```
module: 19
title: KWP-Integration
phase: 1
status: detail-spec-draft
last_updated: 2026-05-08
session_source: brainstorming-session-2026-05-07-2206.md
related_lps: [LP-4, LP-7, LP-9, LP-10, LP-11]
spec_style: requirements-pur
```

Modul 19 — KWP-Integration

> ***Detail-Spec gemäß LP-9.*** *Dritte USP-Säule (Legacy-Anbindung). Geerkens-Pflicht. Über M33a Adapter-Pattern (LP-11). Phase 1: Makeshift-Variante mit kwpsync.exe; nach KWP-Geschäftsführer-Termin ggf. Native API.*

Themenblöcke

```
A. Architektur-Übersicht (Cloud-Queue + Windows-Client + MSSQL)
B. Datenmodell
C. Setup-Wizard
D. Sync-Mechanik (Push/Pull/Document)
E. Schema-Drift-Sentinel (PM-4)
F. kwpsync.exe Windows-Client
G. Persistente Queue + Rate-Limited-Replay (EC-8)
H. Concurrent-Edit-Handling (EC-9)
I. Adapter-Interface (für M33a)
J. Permissions
K. Phase 2: Taifun + andere ERPs
```

Block A — Architektur-Übersicht

Drei-Komponenten-Architektur

```
HiveGrid Cloud (Server-side)
  ↕ HTTPS + Token
kwpsync.exe (Windows-Client beim Tenant)
  ↕ MSSQL Direct Connection
KWP MSSQL-Datenbank (Tenant-Server)
```

Komponente 1: HiveGrid Cloud-API — Push/Pull-API, Queue + Reservierungen, Schema-Drift-Sentinel, Audit-Trail. Endpoints: `/api/v1/integrations/kwp/queue/{tenant_token}/...`

Komponente 2: kwpsync.exe (Windows-Client) — Windows-Service auf Geerkens' Server. Polling-Loop (30 Sek). Lokale MSSQL-Verbindung. Heartbeat (5 Min). Persistente lokale Queue (SQLite). Auto-Update.

Komponente 3: KWP MSSQL-Datenbank — Schreib-/Lese-Zugriff via SQL. Reverse-Engineered (kein offizielles Schema-Doc).

Bidirektionaler Datenfluss

HiveGrid → KWP (Push): Neuer Auftrag aus Won-Handoff, Status-Updates, Document-Uploads. Trigger via M17-Action `kwp.push_auftrag`.

KWP → HiveGrid (Pull): Stammdaten-Updates, Belege/Rechnungen, Status-Änderungen. Trigger `kwp.auftrag_pulled` in M17.

Adapter-Pattern (LP-11, M33a)

KWP ist eine **App** in M33a. Connection-Type `windows_client` (KWP-Spezial-Variante: lokaler Client mit Token).

Block B — Datenmodell

```
KwpSettings              – Tenant-spezifische KWP-Einstellungen
KwpRegistry              – Mapping HiveGrid-Entität ↔ KWP-Entity
KwpProjectQueue          – Queue für Push-Operationen (Cloud-Side)
KwpProjectNumberSequence – Pro Tenant fortlaufende Projektnummern
KwpProjectNumberReservation – Reservierte Nummern für Drafts
KwpApiKey                – Token für kwpsync.exe-Auth
KwpDocument              – Synchronisierte Dokumente
KwpErrorReport           – Fehler-Log für Tenant-Admin
KwpSchemaSnapshot        – Letztes bekanntes KWP-Schema
```

KwpSettings

- `id`, `tenant_id`, `data_connection_id` (FK auf M33a `DataConnection`)
- `kwp_db_host`, `kwp_db_port`, `kwp_db_name` (lokal verwaltet, **nicht** in Cloud gespeichert für DSGVO)
- `kwpsync_token` (verschlüsselt — Cloud-API-Token)
- `kwpsync_version`
- `kwpsync_last_heartbeat_at`
- `polling_interval_seconds` (Default 30)
- `enabled_features[]` (`auftrag_push`, `belege_pull`, `stammdaten_pull`, `document_sync`)

KwpRegistry

- `id`, `tenant_id`
- `hivegrid_entity_type`, `hivegrid_entity_id`
- `kwp_entity_type`, `kwp_entity_id`
- `last_synced_at`, `sync_direction` (`hivegrid_to_kwp | kwp_to_hivegrid | bidirectional`)
- Idempotenz und Mapping

KwpProjectQueue

- `id`, `tenant_id`, `created_at`
- `operation` (`create_auftrag | update_auftrag | upload_document | ...`)
- `payload` (JSON), `idempotency_key`
- `status` (`pending | in_progress | success | failed | retrying`)
- `attempts`, `last_error_text`, `processed_at`

KwpProjectNumberSequence

- `tenant_id`, `current_year`, `current_number`
- Pro Jahr fortlaufend (KWP-Konvention: „2026-0042")

KwpProjectNumberReservation

- `tenant_id`, `reserved_number`, `reserved_at`, `reserved_by_user_id`
- `expires_at` (Default 24h)
- `consumed_at`

KwpApiKey

- `id`, `tenant_id`, `token_hash`
- `created_at`, `last_used_at`, `expires_at`
- Rotation alle 365 Tage (M33a-Default)

KwpDocument

- `id`, `tenant_id`, `kwp_document_id`, `hivegrid_file_id`
- `direction` (`uploaded_to_kwp | downloaded_from_kwp`)
- `synced_at`, `kwp_path`

KwpErrorReport

- `id`, `tenant_id`, `created_at`
- `error_kind` (`schema_drift | connection_lost | sync_conflict | data_validation | ...`)
- `summary_text` (Klartext)
- `tech_details` (JSON für Plattform-Admin)
- `severity` (`info | warn | error | critical`)
- `acknowledged_by_user_id`, `acknowledged_at`

KwpSchemaSnapshot

- `id`, `tenant_id`, `captured_at`
- `schema_hash`
- `tables_metadata` (JSON für Diff-Anzeige)

Persistente Queue: zwei Layer

- **Cloud-Queue:** `KwpProjectQueue` (HiveGrid → kwpsync ausstehend)
- **Lokale Queue (kwpsync.exe):** SQLite-DB, Operationen die bei lokalem MSSQL-Down zwischengepuffert werden, plus Pull-Daten die bei Cloud-API-Down zwischengepuffert werden
- Zwei Resilienz-Layer

Block C — Setup-Wizard

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

1. **Voraussetzungen-Check (Klartext):** *„Du brauchst Admin-Zugang zur KWP-DB. kwpsync.exe muss auf dem Server laufen, auf dem KWP installiert ist (oder einem Server mit Netzwerk-Zugriff). Bestätigen."* + Doku-Link
2. **Token-Generierung:** einmalig sichtbar, Klartext-Warnung
3. **kwpsync.exe-Download:** signiertes Installer-Paket
4. **Konfiguration:** lokal in kwpsync.exe-Config:
   - HiveGrid-API-URL
   - Tenant-Token
   - MSSQL-Connection-String (**lokal**, nicht an Cloud)
   - Als Windows-Service registrieren

5. **Initial-Health-Check:** HiveGrid wartet auf ersten Heartbeat, Klartext-Bestätigung oder Timeout-Hinweis
6. **Initial-Sync (optional):** Klartext-Vorschau *„1.247 Kontakte und 384 Aufträge aus KWP übernehmen?*", Async-Job mit Progress
7. **Feature-Aktivierung:** Tenant-Admin wählt Sync-Features (alle Default aktiv)

Block D — Sync-Mechanik

Push (HiveGrid → KWP)

1. HiveGrid-Event triggert Push (z.B. Won-Handoff → `kwp.push_auftrag`-Action)
2. Cloud-Queue-Eintrag (`KwpProjectQueue`)
3. kwpsync.exe pollt Cloud-Queue alle 30 Sek
4. Pull der Queue-Operationen
5. kwpsync.exe schreibt in lokale MSSQL-DB
6. Erfolg: Status `success`, `KwpRegistry`-Eintrag erstellt
7. Fehler: lokale Queue cached, Retry mit Exponential Backoff

Pull (KWP → HiveGrid)

1. kwpsync.exe pollt MSSQL alle 30 Sek (Trigger-basiert Phase 2)
2. Cursor-basierter Pull (Delta seit letztem Sync)
3. Push zu HiveGrid Cloud-API
4. Verarbeitung: Stammdaten-Updates, Belege, Status-Änderungen
5. Trigger-Event `kwp.auftrag_pulled` für Automation-Engine
6. `KwpRegistry`-Eintrag aktualisiert

Document-Sync

Push: Acceptance-PDF wird beim `acceptance_completed` zu KWP-Auftrags-Akte hochgeladen. Andere Dokumente per Action `kwp.upload_document`.

Pull: Neue Dokumente in KWP werden bei nächstem Sync zu HiveGrid gespiegelt. Sichtbar im Project-Files-Tab unter „Aus KWP".

Idempotenz

- Pro Operation `idempotency_key`
- KwpRegistry für Mapping HiveGrid ↔ KWP
- Doppelte Pushes erkannt + übersprungen

Block E — Schema-Drift-Sentinel (PM-4)

Daily-Check

- Daily-Cron triggert kwpsync.exe einen Schema-Snapshot
- Übertragung zu HiveGrid
- Vergleich mit `KwpSchemaSnapshot.schema_hash`

Bei Abweichung

- `KwpErrorReport` mit `error_kind = schema_drift` und `severity = critical`

- Sync **automatisch pausiert** (Sicherheit > Verfügbarkeit)
- Push-Notification + Email an **Plattform-Admin**, nicht Tenant
- Klartext im Tenant-UI: *„KWP-Update erkannt — manche Felder wurden geändert. Sync pausiert für Sicherheit. Wir sind informiert und beheben das. [Plattform-Admin kontaktieren]"*
- Plattform-Admin gibt Sync nach Analyse + Adapter-Update wieder frei

Tenant-Sicht

- Status-Indicator wird `degraded` (M33a)
- Klartext mit Lösungs-Hinweis (LP-10)

Recovery

- Plattform-Admin (M37) hat Schema-Diff-Tool
- Manuelle Anpassung der Adapter-Logik
- Re-Aktivierung nach Test
- Audit-Trail-Eintrag

Block F — kwpsync.exe Windows-Client

Anforderungen

- **Plattform:** Windows 10+ (auch Server 2016+)
- **Sprache:** .NET 8 oder Go (Boilerplate-Entscheidung)
- **Autostart:** Windows-Service (NSSM oder eingebauter Wrapper)
- **Auto-Recovery:** Restart bei Crash via Windows-Service-Manager
- **Auto-Update:** Daily-Check, signiertes Update-Paket, Restart
- **Logs:** lokal in `C:\ProgramData\HiveGrid\kwpsync\logs\` (rotating)

Heartbeat (EC-7)

- Alle 5 Min HTTPS-Ping
- Cloud protokolliert in `KwpSettings.kwpsync_last_heartbeat_at`
- 15 Min ohne Heartbeat: Status `broken`, Push-Alert an Tenant-Admin + Plattform-Admin

Auto-Recovery (EC-7)

- Windows-Service-Wrapper sorgt für Restart
- Auto-Start beim Boot
- Klartext bei `broken`-Status: *„kwpsync.exe ist nicht erreichbar. Mögliche Ursachen: Windows-Update läuft, Service wurde gestoppt, Server-Reboot. Anleitung: [Schritt für Schritt]"*

Konfigurations-Datei

- `kwpsync.config.json` lokal:
- HiveGrid-API-URL
- Tenant-Token
- MSSQL-Connection-String
- Polling-Interval, Log-Level
- Verschlüsselung: Connection-String + Token via Windows-DPAPI

Update-Mechanismus

- Signiertes Installer-Paket (Code-Signing-Zertifikat)
- Daily-Check auf neue Version
- Auto-Update bei Tenant-Admin-Opt-In
- Manuell: Update-Funktion in HiveGrid-UI mit Download-Link

Block G — Persistente Queue + Rate-Limited-Replay (EC-8)

Lokale Queue (kwpsync.exe)

- SQLite-DB: `C:\ProgramData\HiveGrid\kwpsync\queue.db`
- Tabelle für ausstehende Push-Operationen
- Tabelle für ausstehende Pull-Operationen (Cloud-Down)
- Unbegrenzte Größe (Disk-Limit)

Cloud-Queue (HiveGrid)

- `KwpProjectQueue` mit Timestamps und Idempotency-Keys

Rate-Limited-Replay nach Outage

Bei 6h HiveGrid-Down, 1.230 Operationen in kwpsync-Queue:
- Bei Wiederverbindung: chronologisches Replay
- **Rate-Limit:** max. 100 Ops/Min
- Sichtbarer Progress: *„Re-Sync läuft, 847/1.230 abgearbeitet (4 Min verbleibend)"*

Konflikt-Handling

- Konflikt zwischen Queue und HiveGrid-Stand: Pause + Konflikt-Resolution-View
- Klartext: *„Auftrag #4729 wurde in beiden Systemen editiert. Welcher Stand soll gelten? [HiveGrid] [KWP-Queue] [Manuell mergen]"*

Block H — Concurrent-Edit-Handling (EC-9)

Last-Write-Wins mit Audit-Sichtbarkeit (Phase-1-Default)

- KWP-Sync überschreibt HiveGrid-Stand → vorheriger HiveGrid-Stand wird im Audit gespeichert
- Klartext-Notification an User: *„Dein Edit von 14:19 wurde durch eine KWP-Änderung überschrieben. Dein vorheriger Stand: [Restore-Funktion]"*

Optimistic Locking auf HiveGrid-Seite

- Auftrag mit `version_hash` geladen
- Beim Speichern: Version-Mismatch → Block + Klartext: *„Während du editiert hast, kam eine Änderung aus KWP rein. Hier die Differenz [Diff]. Welcher Stand soll gelten?"*

Visueller Indikator „gerade in KWP bearbeitet"

- kwpsync.exe erkennt Stand-Mismatch (anhand `last_modified`-Feld in MSSQL)
- HiveGrid zeigt Hinweis: *„⚠️ KWP-Änderung erkannt seit deinem letzten Aufruf — bitte neu laden um aktuellen Stand zu sehen."*

Block I — Adapter-Interface (LP-11, M33a)

Adapter-Implementation

KWP-Adapter implementiert das M33a Adapter-Interface mit Connection-Type `windows_client`:

- `connect(tenantId, credentials)` — Token generieren, für kwpsync.exe-Konfiguration zurückgeben
- `disconnect(tenantId, connectionId)` — Token revoken
- `validateConnection(connection)` — `kwpsync_last_heartbeat_at` checken
- `syncPull(connection, since)` — Pull-Request an kwpsync.exe
- `syncPush(connection, items)` — `KwpProjectQueue` schreiben, kwpsync.exe holt ab
- `handleWebhook(connection, payload)` — kwpsync.exe pusht Delta-Updates und Heartbeats

Capabilities

- `pull_kontakte`, `pull_auftraege`, `pull_belege`
- `push_auftrag`, `push_status_update`, `push_document`
- `webhook_inbound`
- `schema_drift_detection`

i18n-Klartexte

`app/integrations/kwp/i18n/de.json` enthält alle Klartexte für Status und Fehler.

Block J — Permissions

Permission	Erlaubt
`kwp.read`	Connection-Status sehen, Sync-Logs einsehen
`kwp.connect`	Setup-Wizard durchführen, Token generieren
`kwp.manage`	Connection editieren, Sync-Features aktivieren/deaktivieren
`kwp.manual_sync`	Manuellen Re-Sync triggern
`kwp.disconnect`	Connection trennen
`kwp.admin`	Schema-Drift-Bypass, Force-Re-Connect — **Plattform-Admin-Sache**

Default-Rollen-Zuweisung (SHK-Template)

Rolle	Permission
GF	`kwp.read` + `kwp.connect` + `kwp.manage` + `kwp.disconnect`
Büro / Backoffice	`kwp.read` + `kwp.manual_sync`
Andere	`kwp.read`

Plattform-Admin (M37)

- `kwp.admin` exklusiv für uns
- Schema-Drift-Bypass nur durch uns (Sicherheit)

Block K — Phase 2: Taifun + andere ERPs

Über das gleiche Adapter-Pattern (LP-11, M33a):

- **Taifun** — analog kwpsync.exe-Pattern (Windows-Client gegen lokale DB)
- **TopKontor** — vermutlich auch lokale-DB-Access
- **Streit V.1** — Cloud-API verfügbar (vermutlich einfacher)
- **Native KWP-API** falls KWP-Geschäftsführer-Termin erfolgreich → ablöst Makeshift-Variante

Implementierung als jeweils eigene App in `app/integrations/{erp-key}/`.

Architektur-Vorschriften (Querschnitt)

- **LP-7:** KWP bleibt Buchhaltungs-Motor, HiveGrid ist Cockpit
- **LP-11, AV #2:** Adapter-Pattern (Block I)
- **LP-10, AV #1:** Klartext-Diagnostik in allen Status-Hinweisen
- **AV #6:** Audit-Trail für alle Sync-Operationen
- **AV #11, PM-4:** Schema-Drift-Sentinel (Block E)
- **AV #15, EC-9:** Optimistic Locking + Last-Write-Wins (Block H)
- **AV #18, EC-7:** Heartbeat + Auto-Recovery (Block F)
- **AV #19, EC-8:** Persistente Queue + Rate-Limited-Replay (Block G)

Detail-Spec-Status: DRAFT

Modul 19 komplett spec'd. **Nach KWP-Geschäftsführer-Termin neu evaluieren** — Native API könnte Modul stark vereinfachen.

Offene Detail-Punkte:
- kwpsync.exe-Sprache (.NET vs. Go) — Boilerplate-Entscheidung
- Konkrete KWP-Tabellen-Mappings (Reverse-Engineering — wird mit Geerkens-Server konkretisiert)
- Schema-Snapshot-Algorithmus-Details
- Konflikt-Resolution-UX-Details bei sehr häufigen Concurrent-Edits
- KWP-spezifische Auftrag-Lifecycle-Mappings (welche KWP-Status entsprechen welchen HiveGrid-Stages?)