DIVERA24/7-Integration

Die Integration mit DIVERA24/7 ist das Herzstück jedes Mandanten. Dieses Kapitel beschreibt detailliert, wie Sie die bidirektionale Schnittstelle einrichten und betreiben.

Was ist DIVERA24/7?

DIVERA24/7 ist eine cloudbasierte Einsatzleit- und Alarmierungssoftware der divera GmbH. Leitstellen nutzen DIVERA, um Einsatzaufträge zu erstellen und Einsatzkräfte zu alarmieren. Die Software ist in Deutschland bei vielen Feuerwehren, Rettungsdiensten und anderen BOS-Organisationen im Einsatz.

Wie arbeiten DIVERA und der Broker zusammen?

Die Kopplung funktioniert in **beide Richtungen**.

Inbound (DIVERA → Broker)

DIVERA sendet bei jedem neuen Einsatz einen **Webhook** (HTTP POST mit JSON-Payload) an eine URL, die Sie in DIVERA konfigurieren. Der Broker empfängt diesen Webhook, speichert ihn in der Datenbank (`alarm_divera_inbound`) und leitet den Alarm an die passenden Pager weiter – basierend auf den `divera_rics` -Einstellungen Ihrer Pager.

Outbound (Broker → DIVERA)

Wenn ein Pager eine Status-Rückmeldung sendet (z.B. „Ich komme"), übersetzt der Broker diese in einen HTTPS-Aufruf an die DIVERA-API. DIVERA kennt dann den Status der alarmierten Kraft und zeigt ihn in der Leitstellen-Übersicht an.

DIVERA-Auth-Key hinterlegen

Bevor Outbound funktionieren kann, müssen Sie Ihren DIVERA-Auth-Key im Broker eintragen.

Auth-Key in DIVERA besorgen

Loggen Sie sich in DIVERA ein. Gehen Sie zu **Verwaltung ? Schnittstellen ? API-Schlüssel** (oder ähnlich, je nach DIVERA-Version). Erstellen Sie einen neuen API-Schlüssel. Der Schlüssel ist eine lange Zeichenkette, etwa:

```
if185u4vdGQrmY3GCNF1rj7Nb4ned8c8xiMPIMNi8Pt1trKXEtxw8K9QZ98LIjUB
```

Kopieren Sie ihn in die Zwischenablage.

Auth-Key im Broker eintragen

In der Broker-GUI: **Einstellungen** ? Button **Alarmdienste verwalten**. Im geöffneten Modal sehen Sie ein Feld **DIVERA24/7 Auth-Key**. Fügen Sie den Schlüssel ein. Nach Klick auf **Speichern** wird der Schlüssel verschlüsselt in der Datenbank abgelegt.

Der Key wird **zweifach** gespeichert – in `tenants.divera_authkey` und in `system_config` unter dem Schlüssel `divera_auth_key` mit Ihrer `tenant_id` . Der Broker-Code bevorzugt beim Zugriff zuerst `tenants.divera_authkey` . Diese Redundanz ist historisch gewachsen und wird in einer zukünftigen Version konsolidiert.

Weitere Alarmdienste-Einstellungen

Im gleichen Modal stehen Toggles für verschiedene Funktionen:

DIVERA24/7 Inbound
Aktiviert die Verarbeitung eingehender Webhooks. Wenn deaktiviert, werden die Webhooks zwar in `alarm_divera_inbound` gespeichert, aber keine Pager werden alarmiert. Nützlich zum Testen/Pausieren.

DIVERA24/7 Outbound
Aktiviert das Senden von Status-Rückmeldungen (KOMME etc.) an DIVERA.

Mandown an DIVERA24/7 senden
Wenn ein Pager seine Mandown-Taste drückt (cmd `0A`), wird automatisch ein Alarm in DIVERA erzeugt. Achtung: Das erstellt einen echten Einsatz in DIVERA!

Notrufe an DIVERA24/7 senden
Analog: Bei Notruf-Taste (cmd `08`) wird ein Einsatz erzeugt.

Tear-Off Alarme weiterleiten
Bei Abreißen des Pagers von der Halterung (cmd `0B`) wird ein Einsatz erzeugt.

Die drei letzten Funktionen sind standardmäßig **deaktiviert**. Aktivieren Sie sie nur, wenn Ihre DIVERA-Konfiguration und Ihre Organisation damit umgehen kann.

Webhook-URL in DIVERA eintragen

Jetzt muss DIVERA wissen, wohin es seine Alarme senden soll.

Die URL konstruieren

Der Broker akzeptiert Webhooks auf zwei äquivalenten URLs:

Variante A (Subdomain-basiert):

```
https://10001.pager.dexa.gmbh/webhook/divera/
```

Der Mandant wird aus der Subdomain `10001` abgeleitet. Der Pfad endet mit `/webhook/divera/` (mit oder ohne trailing slash).

Variante B (Pfad-basiert):

```
https://pager.dexa.gmbh/webhook/divera/10001
```

Der Mandant wird aus dem Pfad-Segment `10001` abgeleitet. Einfacher einzutragen, weil keine Subdomain involviert ist.

Beide Varianten funktionieren identisch. Auch `broker.dexa.gmbh` statt `pager.dexa.gmbh` ist erlaubt.

Den Webhook in DIVERA konfigurieren

In DIVERA: **Verwaltung ? Schnittstellen ? Alarmierung ? Webhook (ausgehend)** oder ähnlich. Legen Sie eine neue Schnittstelle an:

- **Name:** z.B. „BOS Data Broker"
- **URL:** siehe oben
- **HTTP-Methode:** POST
- **Content-Type:** application/json
- **Trigger:** Neuer Einsatz (und ggf. Updates)

Aktivieren Sie die Schnittstelle. DIVERA sollte bei der Aktivierung eine Test-Anfrage senden – Sie sehen diese dann in Ihren Log-Tabs (System oder DIVERA).

Webhook-Payload von DIVERA

DIVERA sendet pro Einsatz ein JSON-Objekt mit allen relevanten Daten. Die wichtigsten Felder:

```
{
 "id": 32375472,
 "foreign_id": "SFH-20260422-153000-001",
 "title": "Rauchmelder ALARM",
 "text": "Rauchsensor hat Rauch erkannt: HLF20-1 RM1",
 "address": "Haferlandweg 18, 48157 Münster",
 "lat": 51.991,
 "lng": 7.651,
 "pager": ["1366481A", "1369114A"],
 "cluster_id": 12345,
 "priority": true,
 "created_at": "2026-04-22T10:11:01+02:00"
}
```

Das Feld **`pager`** ist für die Broker-Logik am wichtigsten – es enthält die RICs, die DIVERA für diesen Einsatz an Pager adressieren möchte. Der Broker gleicht diese RICs mit den **`divera_rics`** seiner Pager ab und findet so die Zieldevices.

Der Inbound-Fluss im Detail

Wenn ein DIVERA-Webhook eingeht, läuft intern folgende Kette ab:

1. **Nginx** nimmt die HTTPS-Anfrage auf Port 443 entgegen und leitet sie an **`iotwebui:8082`** weiter.
2. **iotwebui** ruft den Handler **`handleDiveraWebhook`** auf. Dieser ermittelt den Mandanten (aus Subdomain oder URL-Pfad) und fügt einen Eintrag in die Tabelle **`alarm_divera_inbound`** ein.
3. Ein PostgreSQL-**Trigger** löst ein **`NOTIFY alarm_inbound_new`** aus.
4. Der Dienst **alarm-processor** hat auf diesen Kanal gelauscht. Er liest den Eintrag, parst **`divera_raw`** und ermittelt für jede RIC im **`pager`**-Array die passenden Pager des

Mandanten.

5. Für jeden Treffer-Pager erzeugt er einen Eintrag in **tcp_outbound** mit Status **queued** .
6. Der PostgreSQL-Trigger **notify_alarm_queued** löst aus.
7. Der Dienst **iotserver** hat auf **alarm_queued** gelauscht. Er greift den Eintrag auf und dispatcht ihn – entweder per TCP-Frame oder MQTT-Publish.
8. Nach erfolgreichem Versand (TCP-Ack oder MQTT-Broker-Ack) wird **tcp_outbound.status** auf **sent** gesetzt.
9. Gelingt der Versand nicht: Status wird **retry_pending** , nächster Versuch in 60 Sekunden. Nach 60 erfolglosen Versuchen endgültig **failed** .

Im Mandanten-GUI können Sie diesen Fluss live verfolgen – unter **Alarme** für die eingegangenen DIVERA-Nachrichten, unter **Gesendete Alarme** für die ausgelieferten Pager-Nachrichten.

Der Outbound-Fluss im Detail

Wenn ein Pager eine Status-Rückmeldung sendet:

1. Pager sendet TCP-Frame **<STX>24752000751011#80###<ETX>** (für „KOMME") oder MQTT-Publish auf **p/u/24752000751011/ka** mit **{"type":"80"}** .
2. **iotserver** empfängt die Nachricht, persistiert sie in **tcp_inbound** .
3. Der PostgreSQL-Trigger löst **NOTIFY tcp_inbound_new** aus.
4. Der Dienst **divera-outbound-processor** hat auf diesen Kanal gelauscht.
5. Er prüft: ist das eine Status-Rückmeldung (cmd **02** , **80** , **82** , **84**)? Und ist DIVERA-Outbound für den Mandanten aktiv? Und gibt es einen Auth-Key?
6. Wenn ja: Er sucht in **tcp_outbound** nach dem letzten DIVERA-Alarm (**alarm_id IS NOT NULL**) für diesen Pager.
7. Er sendet einen HTTPS-POST an **https://app.divera247.com/api/setstatus?accesskey=<KEY>** mit Payload:

   ```
   {"person":"24752000751011","Status":{"key":2,"alarm":true,"alarm_id":32375472}}
   ```

8. **key** ist dabei der Status-Code: 1=EMPFANGEN, 2=KOMME, 3=KOMME SPÄTER, 4=KOMME NICHT.
9. Die Antwort von DIVERA wird in **system_logs** mit Category **DIVERA24/7 outbound** protokolliert.

Test der Integration

Nach vollständiger Konfiguration empfehlen wir folgenden Test:

1. Stellen Sie sicher, dass ein Pager online ist (Status grün).

2. Erstellen Sie in DIVERA einen Test-Einsatz mit RIC, der auch in den **divera_rics** des Test-Pagers eingetragen ist.
3. Der Pager sollte innerhalb von 2–5 Sekunden vibrieren.
4. Drücken Sie auf dem Pager „KOMME".
5. In DIVERA sollte der Status „Ich komme" für den Pager-Nutzer erscheinen.
6. Im Broker-GUI unter **Logs** finden Sie alle dazugehörigen Einträge.

Funktioniert ein Schritt nicht, nutzen Sie die **Logs**-Seite zur Diagnose. Filtern Sie nach der **alarm_id** des Test-Einsatzes, um nur die relevanten Einträge zu sehen.

DIVERA-Instance-Mapping

Falls Ihr Mandant mehrere DIVERA-Instanzen nutzt (z.B. eine Haupt-DIVERA und eine Test-Instanz), können Sie unterschiedliche **instance**-Namen konfigurieren. Die URL wäre dann:

```
https://pager.dexa.gmbh/webhook/divera/<instance-name>
```

Das **instance**-Feld wird in **alarm_divera_inbound.instance** gespeichert und hilft beim Unterscheiden der Quellen.

Fehlerbilder

HTTP 403 „Nicht autorisiert" im DIVERA-Outbound-Log
Der Auth-Key ist falsch, abgelaufen oder hat nicht die erforderlichen API-Rechte in DIVERA. Prüfen Sie: Stimmt der Key in den Mandant-Einstellungen mit dem in DIVERA überein?

Webhook geht rein, aber Pager bekommt nichts

- Ist der Pager online (Dashboard prüfen)?
- Hat der Pager die richtige RIC in **divera_rics**?
- Sendet DIVERA die richtige RIC im **pager**-Array des Webhooks?

Alarm ist in alarm_divera_inbound, aber kein Eintrag in tcp_outbound

- Ist der Dienst **alarm-processor** gestartet? **systemctl status alarm-processor** (nur Super-Admin)
- Ist DIVERA-Inbound deaktiviert (Toggle in Einstellungen)?

Diese und weitere Probleme werden im Kapitel **Betrieb & Troubleshooting** vertieft.
