GDT-Integration – Betrieb

Table of Contents

CLI-Referenz . . . 1
- Aufruf . . . 1
- Parameter . . . 1
- Exit-Codes (CLI-Modus) . . . 2
- Beispiele . . . 2

Verzeichnisstruktur im laufenden Betrieb . . . 3
Verarbeitungsablauf . . . 3
Gleichzeitige Verarbeitung . . . 4
Fehlerbehandlung . . . 4
HL7-Nachrichten-Queue . . . 4
Logging . . . 5
Typische Konsolenausgabe . . . 5

CLI-Referenz

Aufruf

```
gdt-hl7-bridge.exe [--daemon] [--studyuid <uid>] [--accnbr <nr>] [--patid <id>]
                   [--nachname <name>] [--vorname <name>]
                   [--geburtsdatum <TTMMJJJJ>] [--fallnummer <nr>]
                   [--help]
```

Parameter

Parameter	Modus	Beschreibung
`--daemon`	Daemon	Startet den GDT-Polling-Dienst. Läuft dauerhaft im Hintergrund.
`--studyuid <uid>`	CLI	DICOM Study UID als Suchparameter für den PDF-Abruf.
`--accnbr <nr>`	CLI	Auftragsnummer / Accession Number als Suchparameter.
`--patid <id>`	CLI	Patientennummer als Suchparameter.
`--scope <ebene>`	CLI	Freigabeebene des Zugangscodes: `patient`, `accession`, `study` oder `auto`. Ohne Angabe gilt `gdtImport.scope`, sonst die Portal-Einstellung. Siehe Freigabeebene.

Parameter	Modus	Beschreibung
`--profile <name>`	CLI	Profil, dessen Einstellungen verwendet werden. Pflicht, wenn mehrere Profile konfiguriert sind. Siehe Profile.
`--print-profiles`	Kontrolle	Gibt alle Profile so aus, wie die Bridge sie verwendet (Vorlage und Profil zusammengeführt): Eingangsverzeichnis, GDT-Felder, Suchreihenfolge, Freigabeebene und alle Ausgaben. Im Profil gesetzte Werte sind mit * markiert, das Integrations-Token wird maskiert. Mit `--profile <name>` nur ein Profil. Ruft das Portal nicht auf und legt keine Verzeichnisse an.
`--nachname <name>`	CLI	Nachname – optional, für Dateinamen-Templates und HL7-Kontext.
`--vorname <name>`	CLI	Vorname – optional, für Dateinamen-Templates und HL7-Kontext.
`--geburtsdatum <TTMMJJJJ>`	CLI	Geburtsdatum im GDT-Format – optional, wird für HL7 automatisch nach JJJJMMTT konvertiert.
`--fallnummer <nr>`	CLI	Fallnummer – optional, für Dateinamen-Templates und HL7 (PV1-Segment).
`--help` / `-h`	beide	Hilfe anzeigen und beenden. Wird auch ohne Parameter ausgegeben.

Sind mehrere Suchparameter angegeben, bestimmt `gdtImport.priority` (Standard: `studyuid → accnbr → patid`), welcher für den API-Aufruf verwendet wird.

Exit-Codes (CLI-Modus)

Code	Bedeutung
`0`	Pipeline erfolgreich abgeschlossen
`1`	Fehler (kein Suchparameter, HTTP-Fehler, MLLP-Timeout, o.ä.)

Beispiele

```
# PDF per Study UID abrufen, HL7 MDM senden
gdt-hl7-bridge.exe --studyuid 2.25.186760260011452398476108131496539831798

# PDF per Auftragsnummer, mit Patientendaten für HL7-Kontext
gdt-hl7-bridge.exe --accnbr 186B7928 --patid 3209648408 --nachname Müller --vorname
Max

# Aufruf aus einem Batch-Skript mit Fehlerprüfung
gdt-hl7-bridge.exe --studyuid %STUDYUID%
if errorlevel 1 echo FEHLER beim PDF-Abruf
```

Verzeichnisstruktur im laufenden Betrieb

`gdt-hl7-bridge` verwaltet drei Unterverzeichnisse unterhalb von `gdtImport.gdtInputDir`:

```
gdtInputDir/
 ├──── *.gdt                    ← Neue Eingangsdateien (werden von der Praxissoftware
abgelegt)
 ├──── done/
 │     └──── *.gdt               ← Erfolgreich verarbeitete Dateien
 └──── error/
     ├──── *_YYYYMMDD-HHmmss.gdt      ← Fehlgeschlagene Dateien (mit Zeitstempel im
Namen)
     └──── *_YYYYMMDD-HHmmss.gdt.err  ← Fehlermeldung als Textdatei (gleicher Name)
```

Die Unterverzeichnisse `done/` und `error/` werden beim ersten Bedarf automatisch angelegt.

Verarbeitungsablauf

Für jede gefundene `*.gdt`-Datei läuft folgender Ablauf:

1. **Stabilitätsprüfung** – Dateien, die in den letzten 500 ms geändert wurden, werden übersprungen (Schutz vor noch laufenden Schreibvorgängen der Praxissoftware).
2. **GDT-Parsing** – Die Eingangsdatei wird im Format GDT 2.1 (ISO-8859-1) gelesen. Sowohl das kompakte Format (ohne Leerzeichen) als auch das Standard-Format werden unterstützt.
3. **Feldauflösung** – Alle im `gdtImport.mapping` definierten Felder werden aus der Datei extrahiert.
4. **Pflichtfeldprüfung** – Fehlende Pflichtfelder aus `gdtImport.required` führen zu einem Fehler.
5. **Prioritätsauswahl** – Der erste Schlüssel aus `gdtImport.priority`, für den ein Wert vorliegt, bestimmt den API-Parameter.
6. **Freigabeebene** – Aus dem GDT-Feld `mapping.scope` bzw. `scope` des Profils wird die Freigabeebene ermittelt (siehe Freigabeebene).
7. **PDF-Abruf** – Die PDF wird vom Portal unter `/api/reports/pdf?session=…&<param>=<wert>&scope=<ebene>` heruntergeladen. HTTP-Weiterleitungen (301, 302, 307) werden automatisch verfolgt. *(wenn `pdfFile.enabled`)*
8. **PDF speichern** – Die PDF wird unter `pdfFile.pdfDir/<pdfFileName>` abgelegt. *(wenn `pdfFile.enabled`)*
9. **GDT-Antwort schreiben** – Die Antwortdatei wird unter `gdtWriter.gdtOutputDir/<gdtFileName>` erzeugt. *(wenn `gdtWriter.enabled`)*
10. **HL7-MDM^T02 senden/speichern** – Die HL7-Nachricht wird per MLLP/TCP gesendet *(wenn `hl7Sender.enabled`)* und/oder als Datei abgelegt *(wenn `hl7Writer.enabled`)*. Schlägt der Versand fehl und ist `hl7Sender.queue.enabled`, wird die Nachricht in die Queue geschrieben statt die Verarbeitung abzubrechen.
11. **Aufräumen** – Die Eingangsdatei wird nach `done/` verschoben.

Gleichzeitige Verarbeitung

Mehrere GDT-Dateien im Eingangsverzeichnis werden gleichzeitig (parallel) verarbeitet. Jede Datei wird intern als separater Job geführt; eine bereits in Bearbeitung befindliche Datei wird beim nächsten Poll-Zyklus übersprungen.

Fehlerbehandlung

Tritt während der Verarbeitung ein Fehler auf (z.B. HTTP-Fehler beim PDF-Abruf, fehlendes Pflichtfeld, MLLP-Timeout, ungültige GDT-Datei), wird:

1. Die Fehlermeldung in der Konsole ausgegeben.
2. Eine Textdatei `<dateiname>.gdt.err` mit der Fehlermeldung in `error/` angelegt.
3. Die Eingangsdatei nach `error/<dateiname>.gdt` verschoben.

Beispiel nach einem HTTP-400-Fehler

```
error/
 ├──── input-accnbr_20260701-141203.gdt
 └──── input-accnbr_20260701-141203.gdt.err   ← Inhalt: "HTTP 400 von
https://portal.../api/reports/pdf?session=*** [STUDY_NOT_FOUND]: Die angeforderte
Untersuchung wurde nicht gefunden."
```

Beispiel nach MLLP-Timeout

```
error/
 ├──── eingabe_20260701-141203.gdt
 └──── eingabe_20260701-141203.gdt.err   ← Inhalt: "MLLP timeout after 10000ms
connecting to ris-server:2333"
```

HL7-Nachrichten-Queue

Wenn `hl7Sender.queue.enabled: true` konfiguriert ist, gehen HL7-Nachrichten bei einem Verbindungsfehler nicht verloren.

Situation	Verhalten
Empfänger nicht erreichbar (ECONNREFUSED, Timeout)	Nachricht wird als `.hl7`-Datei in `queue.dir` gespeichert. Verarbeitung gilt als erfolgreich — GDT-Datei wandert nach `done/`.
Daemon-Modus	Alle `queue.retryIntervalMs` Millisekunden (Standard: 60 s) wird versucht, die Queue zu leeren. Beim ersten Fehlschlag wird abgebrochen und beim nächsten Intervall erneut versucht.

Situation	Verhalten
CLI-Modus	Nach der Verarbeitung wird die Queue sofort einmalig geleert. Der nächste Aufruf wiederholt dies.
Empfänger wieder erreichbar	Gespeicherte Nachrichten werden in chronologischer Reihenfolge nachgesendet und nach erfolgreichem ACK aus der Queue gelöscht.

```
queue/hl7/
 ├──── 1751111295123-a4f2b.hl7   ← ausstehende Nachricht
 └──── 1751111301456-c9d1e.hl7   ← ausstehende Nachricht
```

Logging

Die Anwendung schreibt alle Meldungen auf die Standardausgabe (`stdout`). Zusätzlich kann über das `log`-Modul in `config.json` eine rotierende Logdatei aktiviert werden — siehe Modul: log.

Die Logdatei rotiert täglich und enthält Zeitstempel:

```
2026-06-28 13:48:15 [INFO] gdt-hl7-bridge gestartet [--daemon]
2026-06-28 13:48:15 [INFO]   Input:    C:\RadioWeb\GDT\Input
2026-06-28 14:12:03 [INFO] Processing: eingabe.gdt
2026-06-28 14:12:03 [INFO]   HL7:    send failed (connect ECONNREFUSED ...) – queued
as 1751111295123-a4f2b.hl7
2026-06-28 14:13:03 [INFO] Queue: 1 message(s) pending – retrying ris-server:2333
2026-06-28 14:13:03 [INFO] Queue: sent and removed 1751111295123-a4f2b.hl7
```

TIP Das `log`-Modul übernimmt Rotation und Aufbewahrung automatisch. Eine manuelle Umleitung der Konsolenausgabe ist nicht mehr notwendig.

Typische Konsolenausgabe

Daemon-Modus – normaler Betrieb

```
gdt-hl7-bridge gestartet [--daemon]
  Input:     C:\RadioWeb\GDT\Input
  GDT Out:   C:\RadioWeb\GDT\Output
  PDFs:      C:\RadioWeb\PDFs
  HL7 TCP:   ris-server:2333
  HL7 Queue: queue/hl7 (retry 60000ms)
  HL7 Out:   C:\RadioWeb\HL7\Output
  Interval:  2000 ms
Processing: eingabe.gdt
  Lookup: patid=3209648408
  PDF:    C:\RadioWeb\PDFs\3209648408__1719475200000.pdf
  GDT:    C:\RadioWeb\GDT\Output\3209648408__1719475200000.gdt
```

```
  HL7:    sending to ris-server:2333
  HL7:    sent
  HL7:    C:\RadioWeb\HL7\Output\3209648408__1719475200000.hl7
  Done
```

Daemon-Modus – Empfänger nicht erreichbar, Queue läuft

```
Processing: eingabe.gdt
  Lookup: patid=3209648408
  HL7:    sending to ris-server:2333
  HL7:    send failed (connect ECONNREFUSED 10.0.0.1:2333) — queued as 1751111295123-
a4f2b.hl7
  Done
Queue: 1 message(s) pending — retrying ris-server:2333
Queue: sent and removed 1751111295123-a4f2b.hl7
```

CLI-Modus

```
gdt-hl7-bridge [CLI]
  Lookup: studyuid=2.25.186760260011452398476108131496539831798
  PDF:    C:\RadioWeb\PDFs\3209648408__1719475200000.pdf
  HL7:    sending to ris-server:2333
  HL7:    sent
Fertig.
```