CAR-READER®-DLE

Lokale API-Dokumentation

Inhaltsübersicht

Lesetipp: Für eine erste Integration genügen die Kapitel 2, 3, 5 und 7. Für Livekamerasund mehrere Fahrspurensind zusätzlich die Kapitel 9 bis 11 relevant.

1. Einführung und Integrationsmodell

CAR-READER®-DLE Runtime ist ein lokal betriebenes System zur Kennzeichenerkennung. Anwendungen kommunizieren über HTTP mit der Runtime und erhalten strukturierte JSON-Antworten. Die Bildverarbeitung findet auf dem System statt, auf dem die Runtime ausgeführt wird.

1.1 Typische Integrationsvarianten

Variante	Geeignet für	Empfohlener Endpunkt
Einzelbild	Schrankensteuerung, Zutrittskontrolle, Kamera-Snapshots, manuelle Prüfung	POST /api/single
Lokaler Ordner	Abnahme, Qualitätsprüfung und kontrollierte Stapelverarbeitung auf dem Runtime-Rechner	POST /api/batch
RTSP-Livekamera	Einfahrt, Ausfahrt, Parkplatz, Werkstor und Fahrspurüberwachung	POST /api/rtsp/connect und /api/rtsp/analyze
Mehrkamera	Mehrere getrennte Kameras oder Fahrspuren mit gemeinsamer Runtime	camera_id/lane_id und /api/rtsp/cameras
OEM/Backend	Einbettung in eigene Software, Dienste und Steuerungssysteme	REST-API mit stabiler request_id

1.2 Lokale und entfernte Nutzung

- **Lokale Standardnutzung:** Die Runtime ist standardmäßig unter http://127.0.0.1:5099 erreichbar.
- **Entfernte Nutzung:** Muss ausdrücklich in der Kundenkonfiguration freigegeben und durch Netzwerkregeln geschützt werden.
- **API-Token:** Für Netzwerkzugriffe wird ein API-Token empfohlen. Die Runtime akzeptiert einen produktspezifischen Header oder Bearer-Authentifizierung.
- **Verschlüsselung:** Für Zugriffe über Rechnergrenzen sollte die Verbindung in einer geschützten Umgebung oder über einen kundenseitig verwalteten TLS-Reverse-Proxy erfolgen.

Wichtig
Der lokale Pfad C:\... oder /opt/... in einer API-Anfrage bezeichnet immer einen Pfad auf dem Rechner, auf dem die Runtime läuft - nicht auf dem aufrufenden Client.

2. Schnellstart

2.1 Verfügbarkeit prüfen

```
curl http://127.0.0.1:5099/health


{
  "status": "ok",
  "product": "CAR-READER-DLE Runtime",
  "product_version": "1.0.0",
  "license_ok": true,
  "inference": {
    "workers": 2,
    "active": 0,
    "queued": 0
  },
  "api_version": "1",
  "request_id": "...",
  "timestamp": "2026-07-19 12:00:00"
}
```

2.2 Erstes Bild erkennen

```
curl -X POST "http://127.0.0.1:5099/api/single" \
  -F "image=@kennzeichen.jpg" \
  -F "plate_profile=international_latin"

{
  "ok": true,
  "text": "MAB1234",
  "formatted_text": "M-AB1234",
  "confidence": 0.9945,
  "processing_time_seconds": 0.31,
  "plate_layout": "single_line",
  "line_count": 1,
  "separator_positions": [1],
  "profile": "international_latin",
  "api_version": "1",
  "request_id": "...",
  "timestamp": "2026-07-19 12:00:01"
}
```

Bedeutung von ok

ok=true bestätigt, dass die Anfrage technisch verarbeitet wurde. Ein leeres text-Feld bedeutet, dass in diesem Bild kein verwertbares Kennzeichen erkannt wurde.

3. Transport, Authentifizierung und Sicherheit

3.1 Protokoll und Formate

Eigenschaft	Wert
Transport	HTTP/1.1
Zeichencodierung	UTF-8
JSON-Antworten	application/json; charset=utf-8
Bildupload	multipart/form-data
Standard-Basis-URL	http://127.0.0.1:5099
API-Version	1

3.2 API-Token

Ist ein API-Token konfiguriert, kann es auf zwei gleichwertige Arten gesendet werden:

```
X-CAR-READER-DLE-Token: <API-TOKEN>

Authorization: Bearer <API-TOKEN>
```

Beispiel:

```
curl -H "X-CAR-READER-DLE-Token: <API-TOKEN>" \
  http://127.0.0.1:5099/api/runtime/capabilities
```

Unauthentifizierte Basisendpunkte

Der Health Check, die API-Übersicht und die OpenAPI-Beschreibung können für Überwachung und Schnittstellenerkennung ohne Token bereitgestellt werden. Sie enthalten keine Zugangsdaten.

3.3 Sicherheitsregeln für Integratoren

- API-Token nicht in Quellcode-Repositories, Logdateien oder URLs speichern.
- Token über sichere Anwendungskonfiguration oder geschützte Umgebungsvariablen bereitstellen.
- Bei Netzwerkzugriffen Firewall-Regeln auf die erforderlichen Quellsysteme begrenzen.
- RTSP-Benutzernamen und Kennwörter nicht in Benutzeroberflächen oder Logs im Klartext anzeigen.
- request_id protokollieren; Bild- und Kennzeichendaten nur speichern, wenn dies für den Anwendungsfall erforderlich und zulässig ist.

4. Erkennungsmodi und Profile

4.1 Profile

Profil	Beschreibung	Empfehlung
auto	Automatische Auswahl zwischen normalen, niederländischen und zweizeiligen Formaten.	Gemischte internationale Bildbestände oder unbekannte Layouts. Benötigt meist mehr Rechenzeit.
international_latin	Allgemeine lateinische Kennzeichen mit flexibler Gruppierung.	Standardprofil für europäische und internationale Installationen.
de	Deutschlandbezogene Formatierung.	Bekannte deutsche Fahrspur oder kontrollierter deutscher Bestand.
at	Österreichbezogene Formatierung.	Bekannte österreichische Fahrspur.
ch	Schweizbezogene Formatierung.	Bekannte schweizer Fahrspur.
nl	Niederländische Mehrseparator-Formate.	Bekannte niederländische Fahrspur.
two_line	Zweizeilige beziehungsweise gestapelte Kennzeichen.	Motorrad-, Nutzfahrzeug- oder Sonderkennzeichen mit erkennbarem Zeilenlayout.

Performanceempfehlung

Ist das Land oder Layout bekannt, sollte ein konkretes Profil verwendet werden. auto bietet Flexibilität, kann aber langsamer sein.

4.2 Einzelkennzeichen und Mehrfachkennzeichen

Modus	Verhalten	Typischer Einsatz
Einzelkennzeichen	Liefert das beste erkannte Kennzeichen im Bild.	Einzelne Fahrspur, Schranke, Snapshot.
Mehrfachkennzeichen	Liefert eine plates-Liste mit mehreren sichtbaren Kennzeichen.	Parkplatzübersicht, Zufahrtsbereich, mehrere Fahrzeuge im Bild.

Mehrfachkennzeichen wird pro Anfrage mit multi_plate=true oder über die Kundenkonfiguration aktiviert.

4.3 Fahrspur- und Kamera-ID

lane_id und camera_id sind logische Kennungen. Bei Einzelbildanfragen sind beide Bezeichnungen gleichwertig. Im Livebetrieb sollte jede reale Kamera oder Fahrspur eine dauerhaft stabile ID erhalten, beispielsweise lane-01 oder gate-west.

Schutz vor veralteten Frames

Wenn für dieselbe lane_id bereits eine Erkennung läuft, kann die Runtime einen neuen Frame mit HTTP 429 verwerfen. Dadurch werden keine veralteten Videoframes in einer langen Warteschlange verarbeitet.

5. Gemeinsame Header und Antwortfelder

5.1 Request-Header

Header	Pflicht	Beschreibung
Content-Type	Je nach Endpunkt	application/json oder multipart/form-data. Bei multipart die Boundary durch die HTTP-Bibliothek erzeugen lassen.
X-CAR-READER-DLE-Token	Bei aktivem Token	Produktspezifischer Authentifizierungsheader.
Authorization	Alternative	Bearer <Token>.
X-Request-ID	Optional	Vom Integrator vergebene Korrelations-ID. Wird in der Antwort zurückgegeben.
Accept	Optional	application/json.

5.2 Response-Header

Header	Beschreibung
Content-Type	application/json; charset=utf-8
X-CAR-READER-DLE-API-Version	Aktive API-Hauptversion.
X-Request-ID	Korrelations-ID der Anfrage.

5.3 Standardfelder in JSON-Antworten

Feld	Typ	Beschreibung
ok	boolean	Technischer Erfolg der Operation.
api_version	string	API-Hauptversion.
request_id	string	Korrelations-ID für Logs und Support.
timestamp	string	Zeitpunkt der Antwort auf dem Runtime-System.
instance_name	string	Logischer Name der Runtime-Instanz.
error	string	Lesbare Fehlermeldung bei ok=false.
error_code	string	Maschinenlesbarer Fehlercode.
details	object	Optionale Zusatzinformationen zum Fehler.

Robuste JSON-Verarbeitung
Integrationen sollten unbekannte Zusatzfelder tolerieren. Verlassen Sie sich auf dokumentierte Felder und prüfen Sie ok, HTTP-Status, error_code und request_id.

6. API-Übersicht und Systemendpunkte

Methode	Pfad	Zweck
GET	/api	API-Übersicht und Integrationsinformationen
GET	/api/openapi.json	Maschinenlesbare OpenAPI-Beschreibung
GET	/health	Health Check und Inferenzstatus
GET	/api/license/status	Öffentlicher Lizenzstatus
GET	/api/runtime/capabilities	Profile, Funktionen und Parallelitätsstatus
GET	/api/runtime/config	Effektive öffentliche Konfiguration
GET	/api/diagnostics/package	Support-Paket erzeugen
GET	/api/quality/selftest	Installierten Qualitätsselbsttest ausführen

6.1 API-Übersicht

GET **/api** Überblick

Liefert Produkt-, Lizenz-, Konfigurations- und Funktionsinformationen sowie Verweise auf wichtige Endpunkte.

```
curl http://127.0.0.1:5099/api
```

6.2 OpenAPI

GET **/api/openapi.json** Maschinenlesbarer Vertrag

Geeignet für API-Browser, Codegeneratoren und automatisierte Integrationsprüfungen. Die tatsächliche Runtime-Version bleibt die maßgebliche Quelle.

6.3 Health Check

GET **/health** Betriebsbereitschaft

Für Watchdogs und vor produktiven Erkennungsanfragen. /api/health wird als Alias akzeptiert.

Feld	Bedeutung
status	ok bedeutet, dass der HTTP-Dienst antwortet.
license_ok	Zeigt, ob die Lizenzprüfung erfolgreich ist.
rtsp_camera_count	Registrierte RTSP-Kameras.
rtsp_connected_count	Aktuell verbundene RTSP-Kameras.
uptime_seconds	Laufzeit des Dienstes in Sekunden.

6.4 Lizenzstatus

GET **/api/license/status** Lizenz prüfen

Liefert einen kundenorientierten Status. Die Integration sollte mindestens ok und reason auswerten.

```
{
  "ok": true,
  "reason": "ok",
  "required": true,
  "license_file_configured": true,
  "public_key_file_configured": true,
  "api_version": "1",
  "request_id": "..."
}
```

6.5 Funktionen und Konfiguration

GET **/api/runtime/capabilities** Funktionen abfragen

Liefert unterstützte Profile, freigeschaltete Funktionen, Mehrkamera-Unterstützung und das Kameralimit.

GET **/api/runtime/config** Konfiguration

Liefert die Einstellungen, beispielsweise Standardprofil und Ausgabefelder. Die Antwort ist lesend; Konfigurationsänderungen erfolgen über die Kundenkonfiguration und einen Neustart der Runtime.

6.6 Diagnose und Support

GET **/api/diagnostics/package** Support-Paket erzeugen

Erzeugt ein Diagnosepaket. Die Antwort enthält Status und Speicherort. Das Paket sollte nur an autorisierte Supportkontakte übermittelt werden.

GET **/api/quality/selftest** Installations-Selbsttest

Prüft die Betriebsfähigkeit der installierten Erkennung. Der Endpunkt ist für Installation, Wartung und Support vorgesehen, nicht für normale Erkennungsanfragen.

7. Einzelbild-Erkennung

POST `/api/single` Ein Bild erkennen

Dies ist der zentrale Endpunkt für Snapshots und ereignisgesteuerte Kameraaufnahmen. Das Bild wird als multipart/form-data übertragen.

7.1 Formularfelder

Feld	Typ	Pflicht	Beschreibung
image	Datei	Ja	JPEG, PNG, BMP, TIFF oder WebP.
plate_profile	Text	Nein	auto, international_latin, de, at, ch, nl oder two_line.
country_profile	Text	Nein	Alternative Profilangabe; country wird als Alias akzeptiert.
lane_id	Text	Nein	Stabile Fahrspur-ID für Parallel- und Livebetrieb.
camera_id	Text	Nein	Alias zu lane_id.
use_separator_detection	boolean	Nein	Aktiviert formatierte Ausgabe. Alias: use_gap_reconstruction.
use_gap_ranker	boolean	Nein	Aktiviert die erweiterte Separatorauswertung.
multi_plate	boolean	Nein	Erkennt mehrere sichtbare Kennzeichen.
max_plates	integer	Nein	Maximale Zahl der Ergebnisse im Mehrfachmodus.
include_annotated_image	boolean	Nein	Liefert eine annotierte Darstellung als Base64, falls erlaubt.
include_plate_details	boolean	Nein	Steuert Layout-, Profil- und Separatorfelder.
allow_fast_path	boolean	Nein	Verwendet die konfigurierte qualitätsschonende Schnellverarbeitung.
config	JSON-Text	Nein	Request-spezifische Overrides für bekannte öffentliche Konfigurationsbereiche.

Empfehlung

Für normale Integrationen genügen image, plate_profile und optional lane_id. Erweiterte Parameter sollten nur gezielt verwendet werden.

7.2 Einfacher Request

```
curl -X POST "http://127.0.0.1:5099/api/single" \
  -H "X-Request-ID: gate-west-20260719-0001" \
  -F "image=@C:/Images/capture.jpg" \
  -F "plate_profile=international_latin" \
  -F "lane_id=gate-west"
```

7.3 Typische Antwort

```
{
  "ok": true,
  "text": "MAB1234",
  "formatted_text": "M-AB1234",
  "confidence": 0.9945,
  "processing_time_seconds": 0.31,
  "plate_layout": "single_line",
  "line_count": 1,
  "line_texts": [],
  "separator_positions": [1],
  "separator_count": 1,
  "country_profile": "INTERNATIONAL_LATIN",
  "profile": "international_latin",
  "lane_id": "gate-west",
  "api_version": "1",
  "request_id": "gate-west-20260719-0001",
  "timestamp": "2026-07-19 12:00:01",
  "instance_name": "car-reader-dle-runtime"
}
```

7.4 Ergebnisfelder

Feld	Typ	Bedeutung
text	string	Normalisierter Kennzeichentext ohne Formatseparatoren.
formatted_text	string	Formatierte Darstellung mit ermittelten Separatoren.
confidence	number	Erkennungskonfidenz zwischen 0 und 1.
processing_time_seconds	number	Runtime-interne Verarbeitungszeit der Erkennung.
plate_layout	string	single_line oder two_line.
line_count	integer	Ermittelte Zeilenzahl.
line_texts	array	Zeilentexte bei zweizeiligen Kennzeichen.
separator_positions	array	Positionen der Formatseparatoren im normalisierten Text.
country_profile	string	Ermitteltes oder angewendetes Länderprofil.
profile	string	Tatsächlich verwendetes Profil.
annotated_image_b64	string	Optionale Base64-kodierte JPEG-Annotation.

7.5 Kein Kennzeichen erkannt

```
{
  "ok": true,
  "text": "",
  "formatted_text": "",
  "confidence": 0.0,
  "processing_time_seconds": 0.18,
  "plate_layout": "single_line",
  "api_version": "1",
  "request_id": "..."
}
```

Kein HTTP-Fehler

Ein technisch korrekt verarbeitetes Bild ohne Kennzeichen ist normalerweise eine erfolgreiche HTTP-200-Antwort mit leerem text-Feld.

7.6 Mehrfachkennzeichen

```
curl -X POST "http://127.0.0.1:5099/api/single" \
  -F "image=@parking.jpg" \
  -F "plate_profile=international_latin" \
  -F "multi_plate=true" \
  -F "max_plates=8"

{
  "ok": true,
  "text": "MAB1234",
  "formatted_text": "M-AB1234",
  "confidence": 0.9945,
  "multi_plate_enabled": true,
  "plate_count": 2,
  "plates": [
    {
      "index": 1,
      "text": "MAB1234",
      "formatted_text": "M-AB1234",
      "confidence": 0.9945,
      "xyxy": [120, 210, 410, 305],
      "plate_layout": "single_line"
    },
    {
      "index": 2,
      "text": "BAB987",
      "formatted_text": "B-AB987",
      "confidence": 0.9871,
      "xyxy": [620, 230, 890, 318],
      "plate_layout": "single_line"
    }
  ]
}
```

Die Hauptfelder text, formatted_text und confidence beziehen sich im Mehrfachmodus auf das Ergebnis mit der höchsten Konfidenz. Für alle Ergebnisse ist ausschließlich plates maßgeblich.

7.7 Request-spezifische Konfiguration

Einzelne Einstellungen können für eine Anfrage als JSON-Text im Feld config überschrieben werden. Die globale Kundenkonfiguration bleibt unverändert.

```
curl -X POST "http://127.0.0.1:5099/api/single" \
  -F "image=@capture.jpg" \
  -F 'config={"recognition":{"plate_profile":"auto"},"output":{"include_plate_details":true}}'
```

8. Batch-Verarbeitung

POST **/api/batch** Lokalen Bildordner verarbeiten

Der Batch-Endpunkt verarbeitet unterstützte Bilddateien in einem lokalen Ordner auf dem Runtime-Rechner. Der Request wird als JSON gesendet.

Einsatzgrenze
/api/batch lädt keine Dateien vom aufrufenden Client hoch. Für entfernte Dateien oder parallele Kameraereignisse verwenden Sie wiederholte /api/single-Aufrufe.

8.1 Request-Felder

Feld	Typ	Pflicht	Beschreibung
folder_path	string	Ja	Lokaler Ordner auf dem Runtime-System.
plate_profile	string	Nein	Erkennungsprofil.
use_gap_reconstruction	boolean	Nein	Formatseparatoren ermitteln.
use_gap_ranker	boolean	Nein	Erweiterte Separatorauswertung.
save_non_matches	boolean	Nein	Nicht passende Dateien in einen Unterordner kopieren.
export_char_boxes	boolean	Nein	Zusätzliche annotierte Prüfexporte erzeugen.
export_only_non_matches	boolean	Nein	Prüfexporte auf Abweichungen beschränken.
allow_fast_path	boolean	Nein	Qualitätsschonende Schnellverarbeitung verwenden.
config	object	Nein	Request-spezifische Konfigurationsüberschreibungen.

8.2 Request-Beispiel

```
curl -X POST "http://127.0.0.1:5099/api/batch" \
  -H "Content-Type: application/json" \
  -d '{
    "folder_path": "C:/CAR-READER-DLE/Testbilder",
    "plate_profile": "international_latin",
    "use_gap_reconstruction": true,
    "use_gap_ranker": true,
    "save_non_matches": false,
    "export_char_boxes": false
  }'
```

8.3 Antwort


```
{
  "ok": true,
  "records": [
    {
      "filename": "M-AB1234.jpg",
      "text": "MAB1234",
      "formatted_text": "M-AB1234",
      "confidence": 0.9945,
      "processing_time_seconds": 0.29,
      "ok": true
    }
  ],
  "stats": {
    "total_files": 100,
    "loaded": 100,
    "exact_accuracy": 98.0,
    "soft_accuracy": 99.0,
    "average_confidence": 0.989,
    "duration_seconds": 31.5,
    "average_time_per_image": 0.315
  }
}
```


Dateinamen als Referenz

Die Batch-Qualitätswerte vergleichen das Ergebnis mit dem Dateinamen. Für aussagekräftige exact_accuracy- und soft_accuracy-Werte muss der Dateiname das erwartete Kennzeichen enthalten, zum Beispiel M-AB1234.jpg. Bei beliebigen Dateinamen sollten diese QA-Felder nicht als Erkennungsqualität interpretiert werden.

8.4 Unterstützte Dateiformate

JPEG, PNG, BMP, TIFF und WebP. Es werden nur Dateien im angegebenen Ordner verarbeitet; Unterordner werden nicht automatisch rekursiv durchsucht.

9. RTSP- und Mehrkamera-API

Jede Kamera erhält eine eindeutige camera_id. Die Runtime hält Stream- und Triggerzustände pro Kamera getrennt, während Erkennungsaufträge einen gemeinsamen begrenzten Worker-Pool verwenden.

9.1 Kamera verbinden

POST	/api/rtsp/connect RTSP-Stream registrieren und verbinden		
Feld	Typ	Pflicht	Beschreibung
camera_id	string	Empfohlen	Eindeutige Kamera- oder Fahrspurkennung. Alias: lane_id.
profile_name	string	Nein	Anzeigename für Bedienoberflächen.
rtsp_url	string	Ja	RTSP-Adresse der Kamera.
username	string	Nein	RTSP-Benutzername, falls erforderlich.
password	string	Nein	RTSP-Kennwort, falls erforderlich.
wait_seconds	number	Nein	Maximale Wartezeit auf den ersten Frame.
lane_roi	string	Nein	Normiertes Polygon im Format x,y;x,y;x,y.
export_dir	string	Nein	Optionaler lokaler Exportordner, sofern projektseitig vorgesehen.

```
cur
l -X POST "http://127.0.0.1:5099/api/rtsp/connect" \
  -H "Content-Type: application/json" \
  -d '{
    "camera_id": "lane-01",
    "profile_name": "Einfahrt 1",
    "rtsp_url": "rtsp://192.168.1.50/stream",
    "username": "camera-user",
    "password": "<PASSWORD>",
    "wait_seconds": 10,
    "lane_roi": "0.10,0.30;0.90,0.30;0.95,0.95;0.05,0.95"
  }'
```

ROI-Koordinaten
ROI-Punkte werden normiert angegeben: 0,0 ist links oben und 1,1 rechts unten. Mindestens drei und höchstens acht Punkte werden verwendet.

9.2 Status einer Kamera

GET /api/rtsp/status?camera_id=lane-01 Kamerastatus

Liefert Verbindung, Framealter, Auflösung, tatsächliche Capture-FPS, Triggerzustand, ROI und das letzte Ergebnis.

```
curl "http://127.0.0.1:5099/api/rtsp/status?camera_id=lane-01"
```

9.3 Alle Kameras

GET /api/rtsp/cameras Kameraregistry

```
{
  "ok": true,
  "camera_count": 4,
  "connected_count": 4,
  "max_cameras": 8,
  "connected": true,
  "cameras": [
    {"camera_id": "lane-01", "connected": true},
    {"camera_id": "lane-02", "connected": true}
  ]
}
```

9.4 Vorschauframe

GET /api/rtsp/frame?camera_id=lane-01&max_width=960 Letzten Frame abrufen

Liefert frame_image_b64 als Base64-kodiertes JPEG sowie Status- und Zeitfelder. Dieser Endpunkt ist für Vorschau und Einrichtung gedacht, nicht für hochfrequentes Videostreaming.

9.5 ROI setzen oder löschen

POST `/api/rtsp/lane_roi` Fahrspurbereich aktualisieren

```
curl -X POST "http://127.0.0.1:5099/api/rtsp/lane_roi" \
  -H "Content-Type: application/json" \
  -d '{
    "camera_id": "lane-01",
    "lane_roi": "0.10,0.30;0.90,0.30;0.95,0.95;0.05,0.95"
  }'
```

Zum Löschen der ROI wird lane_roi als leerer String gesendet.

9.6 Aktuellen Frame analysieren

POST `/api/rtsp/analyze` Manuelle oder ereignisgesteuerte Analyse

Feld	Typ	Beschreibung
camera_id	string	Zu analysierende Kamera.
plate_profile	string	Erkennungsprofil.
use_gap_reconstruction	boolean	Formatseparatoren aktivieren.
use_gap_ranker	boolean	Erweiterte Separatorauswertung.
auto_trigger	boolean	false: neuesten Frame sofort analysieren; true: Triggerstatus pollen.
config	object	Request-spezifische öffentliche Overrides.

```
curl -X POST "http://127.0.0.1:5099/api/rtsp/analyze" \
  -H "Content-Type: application/json" \
  -d '{
    "camera_id": "lane-01",
    "plate_profile": "international_latin",
    "auto_trigger": false
  }'
```

Bei auto_trigger=true kann die Antwort triggered=false enthalten. Dies ist ein regulärer Zustand, solange noch kein geeignetes Ereignis vorliegt.

9.7 Verbindung trennen

POST `/api/rtsp/disconnect` Eine oder alle Kameras trennen

```
curl -X POST "http://127.0.0.1:5099/api/rtsp/disconnect" \
  -H "Content-Type: application/json" \
  -d '{"camera_id":"lane-01"}'
```

```
curl -X POST "http://127.0.0.1:5099/api/rtsp/disconnect" \
  -H "Content-Type: application/json" \
  -d '{"all_cameras":true}'
```

10. Fehlerbehandlung und Wiederholungsstrategie

10.1 Fehlerantwort

```
{
  "ok": false,
  "error": "The runtime is currently processing the configured maximum number of recognition jobs.",
  "error_code": "INFERENCE_QUEUE_FULL",
  "retry_after_seconds": 0.5,
  "api_version": "1",
  "request_id": "...",
  "timestamp": "..."
}
```

10.2 Wichtige Fehlercodes

HTTP	error_code	Bedeutung	Empfohlene Reaktion
400	MISSING_IMAGE_FIELD	image fehlt bei /api/single.	Request korrigieren; nicht automatisch wiederholen.
400	REQUEST_PROCESSING_FAILED oder Endpunktfehler	Ungültige Daten oder Operation nicht ausführbar.	error und request_id protokollieren.
401	unauthorized	Token fehlt oder ist ungültig.	Authentifizierung prüfen.
404	not_found	Pfad existiert nicht.	API-Pfad und Version prüfen.
409	RTSP_CAMERA_LIMIT_REACHED	Konfiguriertes Kameralimit erreicht.	Nicht benötigte Kamera trennen oder max_cameras anpassen.
413	payload_too_large	Request überschreitet das Größenlimit.	Bild verkleinern oder Kundenlimit prüfen.
429	LANE_BUSY_FRAME_DROPPED	Für dieselbe Lane läuft bereits ein Auftrag.	Bei Livevideo meist ignorieren; neuen aktuellen Frame später senden.
500	REQUEST_PROCESSING_FAILED	Unerwarteter Verarbeitungsfehler.	request_id sichern, Health prüfen, Support-Paket erzeugen.
503	INFERENCE_QUEUE_FULL	Verarbeitungskapazität vorübergehend ausgelastet.	Nach retry_after_seconds mit Jitter wiederholen.

10.3 Retry-Regeln

Situation	Retry
HTTP 429 bei Liveframe	Nicht denselben alten Frame erneut senden. Auf den nächsten aktuellen Frame warten.
HTTP 503	retry_after_seconds beachten; zusätzlich zufälligen Jitter von etwa 50 bis 250 ms verwenden.
HTTP 401/404/413	Nicht automatisch wiederholen. Konfiguration oder Request korrigieren.
HTTP 500	Einmal begrenzt wiederholen; bei erneutem Fehler Diagnose durchführen.
Netzwerk-Timeout	Nur wiederholen, wenn die Operation idempotent oder eine Duplikatbehandlung vorhanden ist.

```
# Beispiel für 503
wartezeit = response.retry_after_seconds + zufallswert(0.05, 0.25)
warte(wartezeit)
aktuellen_frame_senden()
```

11. Parallelverarbeitung und Performance

Die Runtime begrenzt die gleichzeitige Verarbeitung automatisch, um Latenz und Speicherbedarf bei Lastspitzen stabil zu halten.

11.1 Betriebliche Leistungsgrundsätze

Feld	Bedeutung
Livevideo	Vollständige Erkennung ereignisgesteuert statt für jeden Frame ausführen.
ROI	Relevanten Fahrbahnbereich vollständig, aber nicht unnötig groß abdecken.
Profile	Bei bekanntem Kennzeichenbestand ein passendes Länder- oder Layoutprofil verwenden.
Bildausgabe	Annotationen, JPEG und Base64 nur bei tatsächlichem Bedarf anfordern.
Lastspitzen	HTTP 429 und 503 gemäß Retry-Regeln behandeln und Zielhardware realistisch testen.

11.2 Planung nach Einsatzfall

Einzelkamera / gelegentliche Bilder	Niedrige Komplexität und geringer Speicherbedarf.
Mehrere Kameras	Eindeutige camera_id-Werte, ereignisgesteuerte Analyse und realistischer Burst-Test.
Hohe Ereignisdichte	Leistungsfähigere Zielhardware oder mehrere Runtime Instanzen einplanen.
Edge-System	Bildgröße und ROI passend wählen; Lasttest auf dem konkreten Gerät durchführen.

11.3 Integrationsregeln für Kameras

- Vollständige OCR ereignisgesteuert ausführen, nicht für jeden Videoframe.
- Jeder Kamera eine stabile lane_id oder camera_id zuweisen.
- Bei 429 den verworfenen alten Frame nicht erneut senden.
- Queue-Metriken über /health überwachen.
- Auf dem Rechner einen realistischen 4- oder 8-Kamera-Burst testen.
- Für auto ein höheres Zeitbudget einplanen als für ein bekanntes Länderprofil.

11.4 Timeouts

Anwendung	Empfohlener Client-Timeout
Einzelbild, normale Last	10 bis 30 Sekunden
Lastspitze mit mehreren Kameras	30 bis 60 Sekunden
Batch	Abhängig von Bildzahl; häufig mehrere Minuten
RTSP connect	wait_seconds plus angemessener Netzwerkpuffer

12. Integrationsbeispiele

12.2 Python ohne Zusatzpakete

```
import json
import mimetypes
import uuid
from pathlib import Path
from urllib.request import Request, urlopen

url = "http://127.0.0.1:5099/api/single"
image_path = Path("capture.jpg")
boundary = "----CRDLE" + uuid.uuid4().hex
image = image_path.read_bytes()

parts = []
def field(name, value):
    parts.append(
        f"--{boundary}\r\n"
        f'Content-Disposition: form-data; name="{name}"\r\n\r\n'
        f"{value}\r\n".encode("utf-8")
    )

field("plate_profile", "international_latin")
field("lane_id", "lane-01")
parts.append(
    f"--{boundary}\r\n"
    f'Content-Disposition: form-data; name="image"; filename="{image_path.name}"\r\n'
    f"Content-Type: {mimetypes.guess_type(image_path.name)[0] or 'application/octet-stream'}\r\n\r\n".encode()
    + image + b"\r\n"
)
parts.append(f"--{boundary}--\r\n".encode())
body = b"".join(parts)

request = Request(url, data=body, method="POST", headers={
    "Content-Type": f"multipart/form-data; boundary={boundary}",
    "X-Request-ID": uuid.uuid4().hex,
})
with urlopen(request, timeout=30) as response:
    result = json.loads(response.read().decode("utf-8"))
print(result["formatted_text"], result["confidence"])
```

12.1 PowerShell

```
$Uri = "http://127.0.0.1:5099/api/single"
$Image = "C:\Images\capture.jpg"

curl.exe -sS -X POST $Uri `
  -F "image=@$Image" `
  -F "plate_profile=international_latin" `
  -F "lane_id=lane-01"
```

12.3 JavaScript / Browser oder Node.js

```
const form = new FormData();
form.append("image", file);
form.append("plate_profile", "international_latin");
form.append("lane_id", "lane-01");

const response = await fetch("http://127.0.0.1:5099/api/single", {
  method: "POST",
  body: form,
  headers: { "X-Request-ID": crypto.randomUUID() }
});

const result = await response.json();
if (!response.ok || !result.ok) {
  throw new Error(`${result.error_code}: ${result.error}`);
}
console.log(result.formatted_text, result.confidence);
```

12.4 C#

```
using var client = new HttpClient { Timeout = TimeSpan.FromSeconds(30) };
using var form = new MultipartFormDataContent();
using var image = File.OpenRead(@"C:\Images\capture.jpg");

form.Add(new StreamContent(image), "image", "capture.jpg");
form.Add(new StringContent("international_latin"), "plate_profile");
form.Add(new StringContent("lane-01"), "lane_id");
client.DefaultRequestHeaders.Add("X-Request-ID", Guid.NewGuid().ToString("N"));

using var response = await client.PostAsync(
    "http://127.0.0.1:5099/api/single", form);
string json = await response.Content.ReadAsStringAsync();
response.EnsureSuccessStatusCode();
```

12.5 Delphi

Delphi kann die API beispielsweise über Indy ansprechen. Content-Type und Multipart-Boundary sollten von TIdMultipartFormDataStream erzeugt werden.

```
uses IdHTTP, IdMultipartFormData, System.SysUtils, System.Classes;

procedure RecognizePlate(const ImageFile: string);
var
  Http: TIdHTTP;
  Form: TIdMultipartFormDataStream;
  ResponseText: string;
begin
  Http := TIdHTTP.Create(nil);
  Form := TIdMultipartFormDataStream.Create;
  try
    Http.ReadTimeout := 30000;
    Http.ConnectTimeout := 5000;
    Http.Request.CustomHeaders.Values['X-Request-ID'] :=
      StringReplace(TGUID.NewGuid.ToString, '-', '', [rfReplaceAll]);

    Form.AddFile('image', ImageFile, 'image/jpeg');
    Form.AddFormField('plate_profile', 'international_latin', 'utf-8');
    Form.AddFormField('lane_id', 'lane-01', 'utf-8');

    ResponseText := Http.Post(
      'http://127.0.0.1:5099/api/single', Form);
    // ResponseText mit einer JSON-Bibliothek auswerten.
  finally
    Form.Free;
    Http.Free;
  end;
end;
```

Delphi-Hinweis
Je nach eingesetzter Indy-Version kann die konkrete Überladung von AddFormField geringfügig abweichen. Die HTTP-Struktur bleibt gleich.

13. Versions- und Kompatibilitätsregeln

- **API-Hauptversion:** Das Feld api_version und der Response-Header geben die aktive Hauptversion an.
- **Abwärtskompatible Erweiterungen:** Neue optionale Felder können ergänzt werden. JSON-Parser müssen unbekannte Felder tolerieren.
- **Pflichtfelder:** Dokumentierte Pflichtfelder und error_code-Werte sollten explizit behandelt werden.
- **OpenAPI:** Die laufende Runtime stellt ihre maschinenlesbare Beschreibung über /api/openapi.json bereit.
- **Request-ID:** Eigene X-Request-ID-Werte vereinfachen Support und End-to-End-Nachverfolgung.

Integrationsvertrag
Nicht dokumentierte Felder, lokale Dateien oder interne Prozesse sind kein stabiler Bestandteil der Kundenschnittstelle.

14. Abnahmecheckliste

Prüfung	Erwartetes Ergebnis
/health	HTTP 200, status=ok, license_ok=true.
Authentifizierung	Anfragen ohne oder mit falschem Token werden erwartungsgemäß behandelt.
/api/single	Testkennzeichen wird mit plausibler confidence erkannt.
Kein Kennzeichen	HTTP 200 mit leerem text, kein technischer Fehler.
Profile	Für den Projektbestand geeignetes Profil festgelegt.
429-Verhalten	Liveintegration sendet keinen veralteten Frame erneut.
503-Verhalten	Retry mit retry_after_seconds und Jitter implementiert.
Mehrkamera	Jede Kamera besitzt eine eindeutige camera_id.
Timeouts	Client-Timeouts passen zu Last und Zielhardware.
Datenschutz	Speicherung von Bildern und Kennzeichen ist projektspezifisch geregelt.
Support	request_id und Diagnosepaket können im Störungsfall bereitgestellt werden.

Anhang A: Feldreferenz

A.1 Einzelbildantwort

Feld	Vorhanden	Beschreibung
ok	Immer	Technischer Verarbeitungsstatus.
text	Bei Erkennungsantwort	Normalisierter Text.
formatted_text	Bei Erkennungsantwort	Formatierter Text.
confidence	Bei Erkennungsantwort	Konfidenz 0 bis 1.
processing_time_seconds	Bei Erkennungsantwort	Erkennungszeit innerhalb der Runtime.
plate_layout	Wenn Details aktiv	single_line oder two_line.
line_count	Wenn Details aktiv	Ermittelte Zeilenzahl.
line_texts	Wenn Details aktiv	Texte einzelner Zeilen.
separator_positions	Wenn Details aktiv	Separatorpositionen.
separator_count	Wenn Details aktiv	Zahl der Separatoren.
country_profile	Wenn Details aktiv	Länderformat.
profile	Wenn Details aktiv	Verwendetes Profil.
plates	Mehrfachmodus	Liste erkannter Kennzeichen.
plate_count	Mehrfachmodus	Anzahl der Kennzeichen.
annotated_image_b64	Wenn aktiviert	Base64-JPEG.
lane_id	Wenn angegeben	Logische Fahrspur-ID.
api_version	Immer	API-Hauptversion.
request_id	Immer	Korrelations-ID.
timestamp	Immer	Antwortzeit.
instance_name	Immer	Runtime-Instanz.

A.2 Mehrfachkennzeichen-Element

Feld	Beschreibung
index	Laufende Nummer in der sortierten Ergebnisliste.
text / formatted_text	Normalisierter und formatierter Kennzeichentext.
confidence	OCR-Gesamtkonfidenz.
xyxy	Bounding Box [x1, y1, x2, y2] in Pixelkoordinaten des Eingabebildes.
separator_positions	Formatseparatoren.
plate_layout / line_count / line_texts	Layoutinformationen.
country_profile	Angewendetes oder ermitteltes Formatprofil.

Anhang B: HTTP-Statuscodes

Status	Bedeutung
200 OK	Anfrage verarbeitet. Inhalt von ok und Ergebnisfeldern auswerten.
400 Bad Request	Pflichtfeld fehlt, Bild ungültig oder Operation fachlich nicht ausführbar.
401 Unauthorized	Authentifizierung fehlgeschlagen.
404 Not Found	API-Pfad unbekannt.
409 Conflict	RTSP-Kameralimit erreicht.
413 Payload Too Large	Upload oder JSON überschreitet das konfigurierte Limit.
429 Too Many Requests	Folgeframe derselben Lane wurde bewusst verworfen.
500 Internal Server Error	Unerwarteter Fehler; request_id und Diagnose verwenden.
503 Service Unavailable	Inferenz-Queue voll; kontrolliert später wiederholen.

Anhang C: Glossar

Begriff	Bedeutung
ALPR / ANPR	Automatische Kennzeichenerkennung.
Confidence	Numerische Sicherheit der Erkennung zwischen 0 und 1.
Lane	Logische Fahrspur oder Kameraquelle.
ROI	Region of Interest; relevanter Bildbereich.
RTSP	Netzwerkprotokoll für Kamera-Streams.
Worker	Parallel ausführende Erkennungseinheit der Runtime.
Queue	Begrenzte Warteschlange für Erkennungsaufträge.
Multipart	HTTP-Format für Datei- und Textfelder in einer Anfrage.
Base64	Textkodierung für Binärdaten, beispielsweise annotierte JPEG-Bilder.
OpenAPI	Maschinenlesbare Beschreibung einer REST-Schnittstelle.

Supportfall

Bitte übermitteln Sie bei technischen Problemen die request_id, den Zeitpunkt, den verwendeten Endpunkt, den HTTP-Status und sofern erforderlich, ein über die Runtime erzeugtes Support-Paket.