CAR-READER®-DLE

Local API Documentation

Contents

Reading tip: Chapters 2, 3, 5 and 7 are sufficient for an initial integration. For live cameras and multiple lanes, Chapters 9 to 11 are also relevant.

1. Introduction and Integration Model

CAR-READER®-DLE Runtime is a locally operated license plate recognition system. Applications communicate with the Runtime over HTTP and receive structured JSON responses. Image processing takes place on the system running the Runtime.

1.1 Typical Integration Options

Option	Suitable for	Recommended Endpoint
Single image	Barrier control, access control, camera snapshots and manual inspection	POST /api/single
Local folder	Acceptance testing, quality checks and controlled batch processing on the Runtime system	POST /api/batch
RTSP live camera	Entrance, exit, parking area, factory gate and lane monitoring	POST /api/rtsp/connect and /api/rtsp/analyze
Multiple cameras	Multiple separate cameras or lanes using a shared Runtime	camera_id/lane_id and /api/rtsp/cameras
OEM/Backend	Embedding in proprietary software, services and control systems	REST API with a stable request_id

1.2 Local and Remote Use

- Default local use: The Runtime is available at http://127.0.0.1:5099 by default.
- Remote use: Must be explicitly enabled in the customer configuration and protected by network rules.
- API token: An API token is recommended for network access. The Runtime accepts either a product-specific header or Bearer authentication.
- Encryption: For access across system boundaries, use a protected environment or a customer-managed TLS reverse proxy.

Important
A local path such as C:\... or /opt/... in an API request always refers to a path on the system running the Runtime, not on the calling client.

2. Quick Start

2.1 Check Availability

```
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 Recognize the First Image

```
curl -X POST "http://127.0.0.1:5099/api/single" \
  -F "image=@license-plate.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"
}
```

Meaning of ok

ok=true confirms that the request was processed successfully at the technical level. An empty text field means that no usable license plate was detected in the image.

3. Transport, Authentication and Security

3.1 Protocol and Formats

Property	Value
Transport	HTTP/1.1
Character encoding	UTF-8
JSON responses	application/json; charset=utf-8
Image upload	multipart/form-data
Default base URL	http://127.0.0.1:5099
API version	1

3.2 API-Token

If an API token is configured, it can be sent in either of two equivalent ways:

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

Authorization: Bearer <API-TOKEN>
```

Example:

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

Unauthenticated Basic Endpoints

The health check, API overview and OpenAPI description may be made available without a token for monitoring and interface discovery. They do not contain credentials.

3.3 Security Rules for Integrators

- Do not store API tokens in source-code repositories, log files or URLs.
- Provide tokens through secure application configuration or protected environment variables.
- For network access, restrict firewall rules to the required source systems.
- Do not display RTSP usernames or passwords in plain text in user interfaces or logs.
- Log the request_id; store image and license plate data only when required and permitted for the use case.

4. Recognition Modes and Profiles

4.1 Profiles

Profile	Description	Recommendation
auto	Automatic selection between standard, Dutch and two-line formats.	Mixed international image sets or unknown layouts. Usually requires more processing time.
international_latin	General Latin-character license plates with flexible grouping.	Default profile for European and international installations.
de	German-specific formatting.	Known German lane or controlled German image set.
at	Austrian-specific formatting.	Known Austrian lane.
ch	Swiss-specific formatting.	Known Swiss lane.
nl	Dutch multi-separator formats.	Known Dutch lane.
two_line	Two-line or stacked license plates.	Motorcycle, commercial-vehicle or special plates with a recognizable line layout.

Performance Recommendation

When the country or layout is known, use a specific profile. auto provides flexibility but may be slower.

4.2 Single-Plate and Multi-Plate Recognition

Mode	Behavior	Typical Use
Single plate	Returns the best detected license plate in the image.	Single lane, barrier or snapshot.
Multiple plates	Returns a plates list containing multiple visible license plates.	Parking overview, access area or multiple vehicles in one image.

Multi-plate recognition is enabled per request with multi_plate=true or through the customer configuration.

4.3 Lane and Camera ID

lane_id and camera_id are logical identifiers. For single-image requests, both names are equivalent. In live operation, each physical camera or lane should have a permanently stable ID, such as lane-01 or gate-west.

Protection Against Stale Frames

If recognition is already running for the same lane_id, the Runtime may discard a new frame with HTTP 429. This prevents stale video frames from being processed from a long queue.

5. Common Headers and Response Fields

5.1 Request Headers

Header	Required	Description
Content-Type	Depends on endpoint	application/json or multipart/form-data. For multipart requests, let the HTTP library generate the boundary.
X-CAR-READER-DLE-Token	When token authentication is enabled	Product-specific authentication header.
Authorization	Alternative	Bearer <Token>.
X-Request-ID	Optional	Correlation ID assigned by the integrator. Returned in the response.
Accept	Optional	application/json.

5.2 Response Headers

Header	Description
Content-Type	application/json; charset=utf-8
X-CAR-READER-DLE-API-Version	Active API major version.
X-Request-ID	Correlation ID of the request.

5.3 Standard Fields in JSON Responses

Field	Type	Description
ok	boolean	Technical success of the operation.
api_version	string	API major version.
request_id	string	Correlation ID for logs and support.
timestamp	string	Time of the response on the Runtime system.
instance_name	string	Logical name of the Runtime instance.
error	string	Human-readable error message when ok=false.
error_code	string	Machine-readable error code.
details	object	Optional additional error information.

Robust JSON Processing
Integrations should tolerate unknown additional fields. Rely on documented fields and evaluate ok, the HTTP status, error_code and request_id.

6. API Overview and System Endpoints

Method	Path	Purpose
GET	/api	API overview and integration information
GET	/api/openapi.json	Machine-readable OpenAPI description
GET	/api/meta/info	Detailed API and Runtime overview
GET	/api/meta/routes	List of available public routes
GET	/health	Health check and inference status
GET	/api/license/status	Public license status
GET	/api/runtime/capabilities	Profiles, capabilities and parallel-processing status
GET	/api/runtime/config	Effective public configuration
GET	/api/security/status	Public security settings
GET	/api/diagnostics	Customer-safe diagnostic overview
GET	/api/diagnostics/package	Create support package
GET	/api/quality/selftest	Run the installed quality self-test

6.1 API Overview

GET **/api** Overview

Returns product, license, configuration and capability information, together with references to important endpoints.

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

6.2 OpenAPI

GET **/api/openapi.json** Machine-Readable Contract

Suitable for API browsers, code generators and automated integration checks. The actual Runtime version remains the authoritative source.

6.3 Health Check

GET **/health** Operational Readiness

For watchdogs and before production recognition requests. /api/health is accepted as an alias.

Field	Meaning
status	ok means that the HTTP service is responding.
license_ok	Indicates whether license validation was successful.
inference.workers	Number of configured inference workers.
inference.active	Recognition jobs currently running.
inference.queued	Recognition jobs currently waiting.
inference.rejected	Queue jobs rejected since startup.
rtsp_camera_count	Registered RTSP cameras.
rtsp_connected_count	Currently connected RTSP cameras.
uptime_seconds	Service uptime in seconds.

6.4 License Status

GET **/api/license/status** Check License

Returns a customer-oriented status. The integration should evaluate at least ok and reason.

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

6.5 Capabilities and Configuration

GET **/api/runtime/capabilities** Query Capabilities

Returns supported profiles, enabled features, the multi-camera limit and parallel-processing status.

GET **/api/runtime/config** Configuration

Returns settings such as the default profile and output fields. This is a read-only response; configuration changes are made through the customer configuration and require a Runtime restart.

6.6 Diagnostics and Support

GET **/api/diagnostics/package** Create Support Package

Creates a diagnostic package. The response contains its status and storage location. The package should be sent only to authorized support contacts.

GET **/api/quality/selftest** Installation Self-Test

Checks the operational readiness of the installed recognition system. This endpoint is intended for installation, maintenance and support, not for normal recognition requests.

7. Single-Image Recognition

POST `/api/single` Recognize an Image

This is the primary endpoint for snapshots and event-triggered camera images. The image is transferred as multipart/form-data.

7.1 Form Fields

Field	Type	Required	Description
image	File	Yes	JPEG, PNG, BMP, TIFF or WebP.
plate_profile	Text	No	auto, international_latin, de, at, ch, nl or two_line.
country_profile	Text	No	Alternative profile field; country is accepted as an alias.
lane_id	Text	No	Stable lane ID for parallel and live operation.
camera_id	Text	No	Alias for lane_id.
use_separator_detection	boolean	No	Enables formatted output. Alias: use_gap_reconstruction.
use_gap_ranker	boolean	No	Enables advanced separator analysis.
multi_plate	boolean	No	Detects multiple visible license plates.
max_plates	integer	No	Maximum number of results in multi-plate mode.
include_annotated_image	boolean	No	Returns an annotated image as Base64, if permitted.
include_plate_details	boolean	No	Controls layout, profile and separator fields.
allow_fast_path	boolean	No	Uses the configured quality-preserving fast path.
config	JSON text	No	Request-specific overrides for known public configuration sections.

Recommendation

For normal integrations, image, plate_profile and optionally lane_id are sufficient. Use advanced parameters only when specifically required.

7.2 Basic 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 Typical Response

```
{
  "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 Result Fields

Field	Type	Meaning
text	string	Normalized license plate text without formatting separators.
formatted_text	string	Formatted representation with detected separators.
confidence	number	Recognition confidence between 0 and 1.
processing_time_seconds	number	Recognition processing time within the Runtime.
plate_layout	string	single_line or two_line.
line_count	integer	Detected number of lines.
line_texts	array	Line texts for two-line license plates.
separator_positions	array	Positions of formatting separators in the normalized text.
country_profile	string	Detected or applied country profile.
profile	string	Profile actually used.
annotated_image_b64	string	Optional Base64-encoded JPEG annotation.

7.5 No License Plate Detected

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

Not an HTTP Error

A technically valid image processed without detecting a license plate normally returns HTTP 200 with an empty text field.

7.6 Multiple License Plates

```
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"
    }
  ]
}
```

In multi-plate mode, the main fields text, formatted_text and confidence refer to the result with the highest confidence. The plates array is authoritative for the complete set of results.

7.7 Request-Specific Configuration

Individual settings can be overridden for one request by supplying JSON text in the config field. The global customer configuration remains unchanged.

```
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 Processing

POST **/api/batch** Process a Local Image Folder

The batch endpoint processes supported image files in a local folder on the Runtime system. The request is sent as JSON.

Scope Limitation
/api/batch does not upload files from the calling client. For remote files or parallel camera events, use repeated /api/single requests.

8.1 Request Fields

Field	Type	Required	Description
folder_path	string	Yes	Local folder on the Runtime system.
plate_profile	string	No	Recognition profile.
use_gap_reconstruction	boolean	No	Detect formatting separators.
use_gap_ranker	boolean	No	Advanced separator analysis.
save_non_matches	boolean	No	Copy non-matching files to a subfolder.
export_char_boxes	boolean	No	Create additional annotated review exports.
export_only_non_matches	boolean	No	Limit review exports to mismatches.
allow_fast_path	boolean	No	Use the quality-preserving fast path.
config	object	No	Request-specific configuration overrides.

8.2 Request Example

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

8.3 Response


```
{
  "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
  }
}
```


File Names as Reference Values

The batch quality metrics compare the result with the file name. For meaningful exact_accuracy and soft_accuracy values, the file name must contain the expected license plate, for example M-AB1234.jpg. When arbitrary file names are used, these QA fields must not be interpreted as recognition-quality metrics.

8.4 Supported File Formats

JPEG, PNG, BMP, TIFF and WebP. Only files in the specified folder are processed; subfolders are not searched recursively.

9. RTSP and Multi-Camera API

Each camera receives a unique camera_id. The Runtime maintains separate stream and trigger states for each camera, while recognition jobs use a shared bounded worker pool.

9.1 Connect a Camera

POST	/api/rtsp/connect Register and Connect an RTSP Stream		
Field	Type	Required	Description
camera_id	string	Recommended	Unique camera or lane identifier. Alias: lane_id.
profile_name	string	No	Display name for user interfaces.
rtsp_url	string	Yes	RTSP address of the camera.
username	string	No	RTSP username, if required.
password	string	No	RTSP password, if required.
wait_seconds	number	No	Maximum time to wait for the first frame.
lane_roi	string	No	Normalized polygon in the format x,y;x,y;x,y.
export_dir	string	No	Optional local export folder, if enabled for the project.

```
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": "Entrance 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 Coordinates

ROI points are specified as normalized coordinates: 0,0 is the top-left corner and 1,1 is the bottom-right corner. Use at least three and no more than eight points.

9.2 Camera Status

GET /api/rtsp/status?camera_id=lane-01 Camera Status

Returns connection state, frame age, resolution, actual capture FPS, trigger state, ROI and the latest result.

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

9.3 All Cameras

GET /api/rtsp/cameras Camera Registry

```
{
  "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 Preview Frame

GET /api/rtsp/frame?camera_id=lane-01&max_width=960 Retrieve Latest Frame

Returns frame_image_b64 as a Base64-encoded JPEG together with status and timing fields. This endpoint is intended for preview and setup, not for high-frequency video streaming.

9.5 Set or Clear the ROI

POST `/api/rtsp/lane_roi` Update Lane ROI

```
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"
  }'
```

To clear the ROI, send lane_roi as an empty string.

9.6 Analyze the Current Frame

POST `/api/rtsp/analyze` Manual or Event-Triggered Analysis

Field	Type	Description
camera_id	string	Camera to analyze.
plate_profile	string	Recognition profile.
use_gap_reconstruction	boolean	Enable formatting separators.
use_gap_ranker	boolean	Advanced separator analysis.
auto_trigger	boolean	false: analyze the latest frame immediately; true: poll the trigger state.
config	object	Request-specific public 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
  }'
```

With auto_trigger=true, the response may contain triggered=false. This is a normal state while no suitable event has occurred.

9.7 Disconnect

POST `/api/rtsp/disconnect` Disconnect One or All Cameras

```
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. Error Handling and Retry Strategy

10.1 Error Response

```
{
  "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 Important Error Codes

HTTP	error_code	Meaning	Recommended Action
400	MISSING_IMAGE_FIELD	image is missing for /api/single.	Correct the request; do not retry automatically.
400	REQUEST_PROCESSING_FAILED or endpoint-specific error	Invalid data or operation cannot be performed.	Log error and request_id.
401	unauthorized	Token is missing or invalid.	Check authentication.
404	not_found	Path does not exist.	Check the API path and version.
409	RTSP_CAMERA_LIMIT_REACHED	Configured camera limit reached.	Disconnect an unused camera or adjust max_cameras.
413	payload_too_large	Request exceeds the size limit.	Reduce the image size or check the configured customer limit.
429	LANE_BUSY_FRAME_DROPPED	A job is already running for the same lane.	Usually ignore this for live video; send a new current frame later.
500	REQUEST_PROCESSING_FAILED	Unexpected processing error.	Record the request_id, check health status and create a support package.
503	INFERENCE_QUEUE_FULL	Workers and queue are at capacity.	Retry after retry_after_seconds with jitter.

10.3 Retry Rules

Situation	Retry
HTTP 429 for a live frame	Do not resend the same old frame. Wait for the next current frame.
HTTP 503	Observe retry_after_seconds and add random jitter of approximately 50 to 250 ms.
HTTP 401/404/413	Do not retry automatically. Correct the configuration or request.
HTTP 500	Retry once with a limit; if the error occurs again, run diagnostics.
Network timeout	Retry only if the operation is idempotent or duplicate handling is implemented.

```
# Example for HTTP 503
wait_time = response.retry_after_seconds + random_value(0.05, 0.25)
wait(wait_time)
send_current_frame()
```

11. Parallel Processing and Performance

The Runtime processes multiple recognition requests through a bounded worker pool. Additional jobs wait in a bounded queue. These limits protect latency and memory during load peaks.

11.1 Performance Settings

Field	Meaning
inference_workers	Maximum number of recognition jobs running simultaneously.
cpu_threads	General CPU threads per worker.
plate_detector_cpu_threads	License plate detector CPU threads per worker.
inference_queue_size	Maximum number of additional waiting jobs.
drop_frames_when_lane_busy	Discards stale subsequent frames from the same lane.
onnx_allow_spinning	May reduce latency with a single worker; usually disable with multiple workers.

11.2 Recommended Presets

Use Case	Worker	Threads/Worker	Detector Threads	Spinning	Notes
Single camera / small PC	1	2	2	true	Low complexity and low memory requirements.
2-4 cameras / office PC	2	2	2	false	Good starting point for typical installations.
4-8 cameras / modern PC	4	1-2	1-2	false	Compare 4x1 and 4x2 on the target hardware.
High Throughput	4+	hardware-dependent	hardware-dependent	false	Only after load testing and a memory check.

No General Camera-Count Guarantee

Achievable throughput depends on the CPU, image resolution, profile, traffic density and concurrency. The number of cameras and OCR jobs per second are not the same metric.

11.3 Integration Rules for Cameras

- Run full OCR in response to events, not for every video frame.
- Assign a stable lane_id or camera_id to every camera.
- For HTTP 429, do not resend the discarded old frame.
- Monitor queue metrics through /health.
- Run a realistic 4-camera or 8-camera burst test on the target system.

11.4 Timeouts

Application	Recommended Client Timeout
Single image, normal load	10 to 30 seconds
Load peak with multiple cameras	30 to 60 seconds
Batch	Depends on the number of images; often several minutes
RTSP connect	wait_seconds plus an appropriate network margin

12. Integration Examples

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"
```

12.2 Python Without Additional Packages

```
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"])
```

12.3 JavaScript / Browser or 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 can access the API through Indy, for example. Content-Type and the multipart boundary should be generated by TIdMultipartFormDataStream.

```
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);
    // Parse ResponseText using a JSON library.
  finally
    Form.Free;
    Http.Free;
  end;
end;
```

Delphi Note
Depending on the Indy version used, the exact AddFormField overload may differ slightly. The HTTP structure remains the same.

13. Versioning and Compatibility Rules

- API major version: The api_version field and the response header indicate the active major version.
- Backward-compatible extensions: New optional fields may be added. JSON parsers must tolerate unknown fields.
- Required fields: Documented required fields and error_code values should be handled explicitly.
- OpenAPI: The running Runtime provides its machine-readable description at /api/openapi.json.
- Request ID: Supplying your own X-Request-ID values simplifies support and end-to-end tracing.

Integration Contract
Undocumented fields, local files and internal processes are not stable parts of the customer interface.

14. Acceptance Checklist

Test	Expected Result
/health	HTTP 200, status=ok, license_ok=true.
Authentication	Requests without a token or with an invalid token are handled as expected.
/api/single	The test license plate is recognized with a plausible confidence value.
No license plate	HTTP 200 with an empty text field and no technical error.
Profile	A suitable profile has been selected for the project image set.
HTTP 429 behavior	The live integration does not resend a stale frame.
HTTP 503 behavior	Retry using retry_after_seconds and jitter is implemented.
Multiple cameras	Each camera has a unique camera_id.
Timeouts	Client timeouts match the expected load and target hardware.
Data protection	Storage of images and license plate data is governed by project-specific rules.
Support	The request_id and diagnostic package can be provided when troubleshooting.

Appendix A: Field Reference

A.1 Single-Image Response

Field	Present	Description
ok	Always	Technical processing status.
text	In recognition responses	Normalized text.
formatted_text	In recognition responses	Formatted text.
confidence	In recognition responses	Confidence from 0 to 1.
processing_time_seconds	In recognition responses	Recognition time within the Runtime.
plate_layout	When details are enabled	single_line or two_line.
line_count	When details are enabled	Detected number of lines.
line_texts	When details are enabled	Texts of the individual lines.
separator_positions	When details are enabled	Separator positions.
separator_count	When details are enabled	Number of separators.
country_profile	When details are enabled	Country format.
profile	When details are enabled	Profile used.
plates	Multi-plate mode	List of detected license plates.
plate_count	Multi-plate mode	Number of license plates.
annotated_image_b64	When enabled	Base64 JPEG.
lane_id	When provided	Logical lane ID.
api_version	Always	API major version.
request_id	Always	Correlation ID.
timestamp	Always	Response time.
instance_name	Always	Runtime instance.

A.2 Multi-Plate Result Item

Field	Description
index	Sequential number in the sorted result list.
text / formatted_text	Normalized and formatted license plate text.
confidence	Overall OCR confidence.
xyxy	Bounding box [x1, y1, x2, y2] in input-image pixel coordinates.
separator_positions	Formatting separators.
plate_layout / line_count / line_texts	Layout information.
country_profile	Applied or detected format profile.

Appendix B: HTTP Status Codes

Status	Meaning
200 OK	Request processed. Evaluate ok and the result fields.
400 Bad Request	A required field is missing, the image is invalid or the operation cannot be performed.
401 Unauthorized	Authentication failed.
404 Not Found	Unknown API path.
409 Conflict	RTSP camera limit reached.
413 Payload Too Large	The upload or JSON exceeds the configured limit.
429 Too Many Requests	A subsequent frame from the same lane was deliberately discarded.
500 Internal Server Error	Unexpected error; use request_id and diagnostics.
503 Service Unavailable	Inference queue full; retry later in a controlled manner.

Appendix C: Glossary

Term	Meaning
ALPR / ANPR	Automatic license plate recognition / Automatic number plate recognition
Confidence	Numerical recognition confidence between 0 and 1.
Lane	Logical lane or camera source.
ROI	Region of Interest; relevant area of the image.
RTSP	Network protocol for camera streams.
Worker	Recognition unit executed in parallel by the Runtime.
Queue	Bounded queue for recognition jobs.
Multipart	HTTP format for file and text fields in one request.
Base64	Text encoding for binary data, such as annotated JPEG images.
OpenAPI	Machine-readable description of a REST interface.

Support Case

For technical problems, please provide the request_id, timestamp, endpoint used, HTTP status and, if required, a support package generated by the Runtime.