



# MAS-004\_RPI-Databridge Interface Manual

**Dokumentversion**

3.0

**Softwarestand**

MAS-004\_RPI-Databridge `0.3.0`

**Autor**

Erwin Egli

**Datum**

2026-02-19

# MAS-004\_RPI-Databridge Interface Manual

**Dokumentversion:** 3.0

**Softwarestand:** MAS-004\_RPI-Databridge `0.3.0`

**Autor:** Erwin Egli

**Datum:** 2026-02-19

**System:** Raspberry PLC (Industrial Shields) als Datenbrücke zwischen Microtom und simulierten/realen Subsystemen

## 1. Zweck und Geltungsbereich

Dieses Dokument beschreibt den aktuellen Stand der Applikation `mas004_rpi_databridge` vollständig:

1. Architektur und Nachrichtenfluss.
2. Bedienung der Web-Oberflächen.
3. Vollständige API-Beschreibung aller Endpunkte.
4. Security/TLS, Shared-Secret, Token.
5. Troubleshooting und Betriebschecks.

Die Doku ist für Betrieb, Inbetriebnahme, Test und Erweiterung auf reale Subsysteme (ESP32-PLC, Videojet 3350, Videojet 6530) ausgelegt.

## 2. Aktuelles Zielsystem (Stand 2026-02-19)

### 2.1 Netzwerkadressen

1. Raspi Databridge: <https://192.168.210.20:8080>
2. Microtom Peer (Simulator/Server): <https://192.168.210.10:9090>

### 2.2 Wichtige UI-URLs

1. Home: <https://192.168.210.20:8080/>
2. API Docs: <https://192.168.210.20:8080/docs>
3. Parameter UI: <https://192.168.210.20:8080/ui/params>
4. Settings UI: <https://192.168.210.20:8080/ui/settings>
5. Test UI: <https://192.168.210.20:8080/ui/test>
6. Health: <https://192.168.210.20:8080/health>

## 3. Web-UI Übersicht (mit Screenshots)

### 3.1 Home

The screenshot shows the Videojet web interface. At the top, there is a navigation bar with tabs: Home (selected), API Docs, Parameter, Test UI, and Settings. The main content area has a title "MAS-004\_RPI-Databridge". Below the title, there are several status indicators:

- eth0:** 192.168.210.20
- Outbox:** 0
- Peer:** <https://192.168.210.10:9090>
- eth1:** 192.168.2.100
- Inbox pending:** 0
- Watchdog host:** 192.168.210.10

Below this section, there is a heading "Logs (Read-only)". Under "Logs (Read-only)", there are four log entries:

- All Channels:** (keine Eintraege)
- Raspi:** (keine Eintraege)
- ESP32-PLC:** (keine Eintraege)
- VJ6530 (TTO):** (keine Eintraege)
- VJ3350 (Laser):** (keine Eintraege)

## 3.2 API Docs

The screenshot shows the API documentation for the **MAS-004\_RPI-Databridge** API, version **0.3.0**. The interface includes a navigation bar with Home, API Docs (selected), Parameter, Test UI, and Settings. The main content area displays the API documentation with various endpoints listed under the **default** category. Each endpoint is shown with its method (e.g., GET, POST), path, and description. Some endpoints are highlighted in green, indicating they are part of the current view or selection.

Method	Path	Description
GET	/ Home	
GET	/ui Ui	
GET	/health Health	
GET	/api/ui/status/public	Ui Status Public
GET	/api/ui/status	Ui Status
GET	/api/config	Get Config
POST	/api/config	Update Config
GET	/api/system/network	Get Network
POST	/api/system/network	Set Network
POST	/api/outbox/enqueue	Api Outbox Enqueue
POST	/api/test/send	Api Test Send
POST	/api/inbox	Api Inbox
GET	/api/inbox/next	Api inbox Next

### 3.3 Parameter UI

The screenshot shows the 'Parameter UI' section of the VIDEOJET application. At the top, there is a navigation bar with links: Home, API Docs, Parameter (which is highlighted in blue), Test UI, and Settings. Below the navigation bar, the title 'Parameter UI' is displayed in bold. There are three search/filter input fields: 'Suche' (Search) containing 'pkey / name / message', 'ParamType' (Parameter Type) containing 'z.B. TTP', and 'Excel Import (.xlsx)' with a 'Choose File' button and a message 'No file chosen'. To the right of these are buttons for 'Reload', 'Export XLSX', and 'Import XLSX' (which is currently loading). Below these controls is a table header titled 'Liste' with columns: pkey, min, max, default, rw, current, effective, name, message, and edit. The main area below the header is currently empty, indicating no data is present.

## 3.4 Settings UI



VIDEOJET®

Home API Docs Parameter Test UI **Settings**

### System Settings

Token wird im Browser gespeichert (localStorage). Änderungen an Network können dich aussperren - daher "Apply now" bewusst setzen.  
Hinweis: Subnet-Maske (z.B. 255.255.255.0) und Prefix (z.B. /24) sind identisch - nur andere Schreibweise.

UI Token: MAS004...

#### Raspi Network (eth0/eth1)

eth0 IP	Subnet	Prefix	GW	DNS (eth0)
<input type="text"/>	<input type="text" value="255.255.255.0"/>	<input type="text" value="24"/>	<input type="text"/>	<input type="text" value="Z.B. 10.28.193.4, 10.27.30.201"/>
eth1 IP	Subnet	Prefix	GW	DNS (eth1)
<input type="text"/>	<input type="text" value="255.255.255.0"/>	<input type="text" value="24"/>	<input type="text"/>	<input type="text" value="optional"/>

Hinweis: Für Produktions-/Firmennetz normalerweise nur 'eth0' mit Gateway und DNS setzen. 'eth1' Gateway leer lassen, falls nur Maschinen-LAN.

Apply now (live setzen)

#### Status

#### Databridge / Microtom

peer_base_url	peer_watchdog_host	peer_health_path
<input type="text"/>	<input type="text"/>	<input type="text"/>
http_timeout_s	tls_verify	eth0_source_ip
<input type="text"/>	<input type="text" value="true/false"/>	<input type="text"/>
shared_secret	(leer = aus)	
<input type="text"/>	(leer = aus)	
<input type="checkbox"/> shared_secret löschen (auf leer setzen)		
<input type="button" value="Save Bridge + Restart"/>		

#### Device Endpoints (ESP / VJ3350 / VJ6530)

ESP host	ESP port	ESP watchdog host	<input type="checkbox"/> Simulation
<input type="text"/>	<input type="text"/>	<input type="text" value="leer = esp_host"/>	<input type="checkbox"/> Simulation
VJ3350 host	VJ3350 port		<input type="checkbox"/> Simulation
<input type="text"/>	<input type="text"/>		<input type="checkbox"/> Simulation
<input type="button" value="Save Devices + Restart"/>			

#### Daily Log Files

Keep days (All)	Keep days (ESP32)	Keep days (TTO)	Keep days (Laser)
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Save Log Settings + Restart"/>		<input type="button" value="Reload Log File List"/>	

Datei	Typ	Datum	Groesse	Aktion

## 3.5 Test UI

The screenshot shows the VIDEOJET Test UI interface with four main sections:

- RASPI-PLC**: Manual input goes directly to Microtorm. Multi-send supported. Input field: ParamType hint (optional) e.g. TTP00002=23, TTP00003=10 or MAP0001=. Buttons: Send, Clear Output. Log buttons: Reload Log, Download Log, Clear Log (red), loading... (green).
- ESP-PLC**: Manual input goes ESP-PLC -> RASPI -> Microtorm. Multi-send supported. Input field: ParamType hint (MAS) e.g. 0026=20, 0027=11 or MAP0001=500. Buttons: Send, Clear Output. Log buttons: Reload Log, Download Log, Clear Log (red), loading... (green).
- VJ3350 (Laser)**: Manual input goes VJ3350 -> RASPI -> Microtorm. Multi-send supported. Input field: ParamType hint (LSE) e.g. 1000=1; 1001=0 or LSW1000=1. Buttons: Send, Clear Output. Log buttons: Reload Log, Download Log, Clear Log (red), loading... (green).
- VJ6530 (TTO)**: Manual input goes VJ6530 -> RASPI -> Microtorm. Multi-send supported. Input field: ParamType hint (TTE) e.g. TTP00002=23, TTP00003=10. Buttons: Send, Clear Output. Log buttons: Reload Log, Download Log, Clear Log (red), loading... (green).

## 4. Architektur und Datenfluss

### 4.1 Hauptkomponenten

1. **FastAPI Web/API Server** in `mas004_rpi_databridge/webui.py`
2. **Router-Loop** in `mas004_rpi_databridge/router.py`
3. **Sender-Loop / Outbox Worker** in `mas004_rpi_databridge/service.py`
4. **Persistenz (SQLite)** in `mas004_rpi_databridge/db.py`
5. **Parameter- und Regelwerk** in `mas004_rpi_databridge/params.py`
6. **Watchdog/Health** in `mas004_rpi_databridge/watchdog.py`

### 4.2 Persistente Queues

1. **Inbox**: eingehende Requests (dedupliziert über Idempotency-Key)
2. **Outbox**: ausgehende Requests mit Retry/Backoff bis erfolgreichem 2xx

### 4.3 Business-Nachrichtenformat

Allgemein:

```
<PTYPE><PID>=<WERT>
```

Read:

```
<PTYPE><PID>=?
```

Beispiele: 1. `TTP00002=?` 2. `TTP00002=50` 3. `MAS0026=20` 4. `LSE1000=1`

## 4.4 Routing-Logik

Prefix-basiert: 1. `TT*` -> Kanal `vj6530` 2. `LS*` -> Kanal `vj3350` 3. `MA*` -> Kanal `esp-plc` 4. sonst -> `raspi`

# 5. Security und Auth

## 5.1 TLS

1. Web/UI/API laufen per HTTPS (`webui_https=true`).
2. Bei IP-Wechsel muss das Zertifikat neu auf die neue IP (SAN) ausgestellt werden.
3. Für Browser ohne Warnung muss `raspi.crt` als vertrauenswürdig importiert werden.

## 5.2 UI-Token ( `x-Token` )

1. Fast alle Betriebs-/Konfigurations-APIs sind token-geschützt.
2. Token wird in der Settings-UI in `localStorage` abgelegt (`mas004_ui_token`).

## 5.3 Shared Secret ( `x-shared-Secret` )

1. Gilt für eingehendes `POST /api/inbox` (Microtom -> Raspi).
2. Ist `shared_secret` in Config gesetzt, ist Header Pflicht.
3. Ist `shared_secret` leer, ist Prüfung deaktiviert.

## 5.4 Idempotency

1. `X-Idempotency-Key` für robuste Retry-Strategie.
2. Inbox dedupliziert über UNIQUE-Key.
3. Callback-Korrelation über `X-Correlation-Id`.

## 6. API-Gesamtübersicht

---

### 6.1 Öffentliche Endpunkte (ohne Token)

1. GET /
2. GET /ui
3. GET /docs
4. GET /docs/swagger
5. GET /ui/params
6. GET /ui/settings
7. GET /ui/test
8. GET /ui/assets/videojet-logo.jpg
9. GET /health
10. GET /api/ui/status/public
11. POST /api/inbox (optional Shared-Secret)

### 6.2 Token-geschützte Endpunkte

1. GET /api/ui/status
2. GET /api/config
3. POST /api/config
4. GET /api/system/network
5. POST /api/system/network
6. POST /api/outbox/enqueue
7. POST /api/test/send
8. GET /api/inbox/next
9. POST /api/inbox/{msg\_id}/ack
10. POST /api/params/import
11. GET /api/params/export
12. GET /api/params/list
13. POST /api/params/edit
14. GET /api/ui/logs/channels
15. GET /api/ui/logs
16. POST /api/ui/logs/clear
17. GET /api/ui/logs/download
18. GET /api/logfiles/list
19. GET /api/logfiles/download

## 7. Vollständige API-Referenz

---

### 7.1 UI und Basis

---

`GET /`

1. Liefert Home-HTML mit Live-Countern für Outbox/Inbox.
2. Keine Auth.

`GET /ui`

1. Alias auf Home.
2. Keine Auth.

`GET /docs`

1. Wrapper-Seite mit eingebettetem Swagger (`/docs/swagger`).
2. Keine Auth.

`GET /docs/swagger`

1. FastAPI Swagger UI.
2. Keine Auth.

`GET /ui/assets/videojet-logo.jpg`

1. Liefert Logo-Asset.
2. Keine Auth.
3. `404`, falls Datei fehlt.

`GET /health`

Response:

```
{"ok": true}
```

### 7.2 Status API

---

`GET /api/ui/status/public`

1. Keine Auth.
2. Für Home-Liveanzeige.

Response:

```
{  
  "ok": true,  
  "outbox_count": 0,  
  "inbox_pending": 0  
}
```

#### GET /api/ui/status

1. Header: X-Token
2. Liefert Betriebsstatus inkl. Device-Konfiguration.

Response (Beispiel):

```
{  
  "ok": true,  
  "outbox_count": 0,  
  "inbox_pending": 0,  
  "peer_base_url": "https://192.168.210.10:9090",  
  "devices": {  
    "esp": {"host": "", "port": 0, "simulation": true, "watchdog_host": ""},  
    "vj3350": {"host": "", "port": 0, "simulation": true},  
    "vj6530": {"host": "", "port": 0, "simulation": true}  
  }  
}
```

## 7.3 Konfiguration

#### GET /api/config

1. Header: X-Token
2. Liefert komplette Runtime-Konfiguration.
3. ui\_token und shared\_secret werden maskiert ( \*\*\* ).

#### POST /api/config

1. Header: X-Token
2. Body: partielles JSON gemäß ConfigUpdate.
3. Speichert Config und startet Service neu.

Body-Felder ( ConfigUpdate ): 1. peer\_base\_url , peer\_watchdog\_host , peer\_health\_path 2.   
tls\_verify , http\_timeout\_s , eth0\_source\_ip 3. webui\_port , ui\_token , shared\_secret 4.   
esp\_host , esp\_port , esp\_simulation , esp\_watchdog\_host 5. vj3350\_host , vj3350\_port ,   
vj3350\_simulation 6. vj6530\_host , vj6530\_port , vj6530\_simulation 7. logs\_keep\_days\_all ,   
logs\_keep\_days\_esp , logs\_keep\_days\_tto , logs\_keep\_days\_laser

Response:

```
{"ok": true}
```

## 7.4 Netzwerk

**GET /api/system/network**

1. Header: `X-Token`
2. Liefert gespeicherte Netzwerkconfig und live erkannte IP/GW Infos.

**POST /api/system/network**

1. Header: `X-Token`
2. Body gemäß `NetworkUpdate` :
3. `eth0_ip`, `eth0_prefix`, `eth0_gateway`, `eth0_dns`
4. `eth1_ip`, `eth1_prefix`, `eth1_gateway`, `eth1_dns`
5. `apply_now` (bool)
6. Validiert DNS auf IPv4-Format.
7. Speichert Config.
8. Optional `apply_now` : versucht Live-Anwendung über `dhcpcd/nmcli`.

Beispiel:

```
{  
    "eth0_ip": "192.168.210.20",  
    "eth0_prefix": 24,  
    "eth0_gateway": "192.168.210.1",  
    "eth0_dns": "10.28.193.4 10.27.30.201",  
    "eth1_ip": "192.168.2.100",  
    "eth1_prefix": 24,  
    "eth1_gateway": "",  
    "eth1_dns": "",  
    "apply_now": true  
}
```

## 7.5 Outbox/Test/Inbox

### POST /api/outbox/enqueue

1. Header: `X-Token`
2. Body (`OutboxEnqueue`):
3. `method` (Default `POST`)
4. `path` (Default `/api/inbox`)
5. `url` (optional, überschreibt `path`)
6. `headers` (JSON)
7. `body` (JSON)
8. `idempotency_key` (optional)
9. Legt Job in Outbox an.

### POST /api/test/send

1. Header: `X-Token`
2. Body (`TestSendReq`):
3. `source`: `raspi|esp-plc|vj3350|vj6530`
4. `msg`: Einzel- oder Mehrfachnachricht (Komma/Semikolon/Zeilenumbruch)
5. `ptype_hint`: optional, 3 Buchstaben
6. Persistiert schreibende Parameternachrichten lokal (wenn erlaubt) und queued Versand an Peer.

Beispiel:

```
{  
  "source": "vj6530",  
  "msg": "TTP00002=23, TTP00003=10",  
  "ptype_hint": "TTP"  
}
```

### POST /api/inbox

1. Keine Token-Auth.
2. Optional Header `X-Shared-Secret` (wenn aktiv).
3. Optional Header `X-Idempotency-Key`.
4. Akzeptiert JSON oder Plaintext.
5. Speichert in Inbox dedupliziert.

Response:

```
{  
  "ok": true,  
  "stored": true,  
  "idempotency_key": "..."  
}
```

Fehler: 1. `401 Unauthorized (shared secret)` bei falschem/fehlendem Secret.

#### `GET /api/inbox/next`

1. Header: `X-Token`
2. Debug: nächste pending Inbox-Nachricht.

#### `POST /api/inbox/{msg_id}/ack`

1. Header: `X-Token`
2. Markiert Nachricht als erledigt.

## 7.6 Parameter API

---

#### `POST /api/params/import`

1. Header: `X-Token`
2. Multipart Upload Feld `file`.
3. Nur `.xlsx` erlaubt.
4. Importiert Parametertabelle inkl. Regeln.

#### `GET /api/params/export`

1. Header: `X-Token`
2. Query optional: `ptype`, `q`
3. Download als `params_export.xlsx`.

#### `GET /api/params/list`

1. Header: `X-Token`
2. Query:
3. `ptype` optional
4. `q` optional
5. `limit` default `200`
6. `offset` default `0`
7. Liefert Parameterliste für UI-Tabelle.

**POST /api/params/edit**

1. Header: `X-Token`
2. Body ( `ParamEdit` ):
  - 3. `pkey` (Pflicht)
  - 4. `default_v` optional
  - 5. `min_v` optional
  - 6. `max_v` optional
  - 7. `rw` optional ( `R|W|R/W` )
8. Validiert und aktualisiert Metadaten.
9. `400` bei ungültigen Eingaben.

## 7.7 Log APIs

---

**GET /api/ui/logs/channels**

1. Header: `X-Token`
2. Liefert bekannte Logkanäle.

**GET /api/ui/logs**

1. Header: `X-Token`
2. Query:
  - 3. `channel` (Pflicht)
  - 4. `limit` default `250`
5. Liefert Logeinträge als JSON.

**POST /api/ui/logs/clear**

1. Header: `X-Token`
2. Query: `channel` (Pflicht)
3. Löscht Kanal-Log.

**GET /api/ui/logs/download**

1. Header: `X-Token`
2. Query: `channel` (Pflicht)
3. Download als `<channel>.log`.

**GET /api/logfiles/list**

1. Header: `X-Token`
2. Wendet Retention an und listet Tagesdateien.

```
GET /api/logfiles/download
```

1. Header: X-Token
2. Query: name (Pflicht)
3. Download einer Tagesdatei.
4. 404 falls Datei nicht vorhanden.

## 8. Microtom-Schnittstelle (fachlich)

### 8.1 Microtom -> Raspi

Empfohlener Aufruf:

```
curl -k -X POST "https://192.168.210.20:8080/api/inbox" \  
-H "Content-Type: application/json" \  
-H "X-Idempotency-Key: mt-20260219-0001" \  
-H "X-Shared-Secret: <SECRET>" \  
-d "{\"cmd\":\"TTP00002=?\", \"source\":\"microtom\"}"
```

### 8.2 Raspi -> Microtom Callback

Raspi sendet an peer\_base\_url + /api/inbox, z. B.: 1. URL: https://192.168.210.10:9090/api/inbox 2. Body: {"msg": "TTP00002=55", "source": "raspi"} 3. Header: X-Idempotency-Key, X-Correlation-Id

Microtom muss auf 2xx antworten, sonst bleibt Nachricht in Outbox und wird erneut versucht.

## 9. Performance und Zuverlässigkeit

### 9.1 Aktuelle Optimierungen

1. Persistenter HTTP-Client mit Connection-Reuse.
2. Schnellere Sender-/Router-Loop-Pollingzeiten.
3. Health-Endpoint als primäre Watchdog-Quelle.
4. Verbesserte Outbox-Reihenfolge (next\_attempt, retry\_count, created\_ts).
5. Ungültige URLs werden verworfen statt ewig retried.

## 9.2 Relevante Config-Felder

1. http\_timeout\_s
2. watchdog\_interval\_s
3. watchdog\_timeout\_s
4. watchdog\_down\_after
5. retry\_base\_s
6. retry\_cap\_s
7. tls\_verify
8. eth0\_source\_ip

## 10. Betrieb: Git/Deploy

### 10.1 Lokal -> Remote

```
git add .
git commit -m "Update docs"
git push origin main
```

### 10.2 Raspi Update

```
cd /opt/MAS-004_RPI-Databridge
git pull --ff-only
sudo systemctl restart mas004-rpi-databridge.service
sudo systemctl status mas004-rpi-databridge.service
```

## 11. Troubleshooting

### 11.1 Outbox bleibt > 0

Prüfen:

```
sudo sqlite3 /var/lib/mas004_rpi_databridge/databridge.db \
"SELECT id,method,url,retry_count,datetime(next_attempt_ts,'unixepoch','localtime') FROM outbox"
```

Typischer Fall: alte Peer-IP in URL (z. B. 192.168.1.x ) führt zu Timeouts.

### 11.2 Service langsam oder verzögert

Prüfen:

```
sudo journalctl -u mas004-rpi-databridge.service -n 200 --no-pager | grep "\[OUTBOX\]"
```

## 11.3 DNS/Routing Probleme

Prüfen:

```
ip route  
cat /etc/resolv.conf  
getent hosts github.com
```

## 11.4 Browser zeigt "Nicht sicher"

1. Zertifikat neu mit aktueller IP/SAN erstellen.
2. `raspi.crt` in Windows Root-Store importieren.
3. Browser neu starten.

## 12. Erweiterung auf reale Geräte

---

Aktuell sind ESP/VJ3350/VJ6530 in Simulation möglich. Für reale Anbindung:

1. Device-Endpunkte in Settings setzen.
2. Protokolladapter aktivieren/erweitern (`device_bridge.py`).
3. Parameter-Mapping aus der XLSX-Tabelle pflegen.
4. Kommunikations- und Fehlerfälle pro Gerät testen.

## 13. Änderungslog

---

1. **v3.0 (2026-02-19):** Vollständige API-Beschreibung aller Endpunkte, neue IP-Umgebung `192.168.210.x`, UI-Screenshots, aktualisierte Betriebs- und Security-Kapitel.