EtherNet/IP++ – Funktionsdokumentation

Plattform: Android (minSdk 26 / compileSdk 35)

Technologiestack: Kotlin 2.0 · Jetpack Compose · Material 3 · Room · Coroutines

Protokoll: EtherNet/IP (CIP Explicit Messaging, TCP Port 44818)

Inhaltsverzeichnis

1. App-Überblick & Einstieg

EtherNet/IP++ ist eine professionelle EtherNet/IP-Diagnose- und Steuerungsapp für Android. Sie implementiert das CIP Explicit Messaging-Protokoll (ODVA-Standard) vollständig in Kotlin ohne externe Bibliotheken und richtet sich an Automatisierungstechniker, Elektriker und Entwickler.

Unterstützte Verbindungsart:

Typ	Beschreibung
EtherNet/IP (Explicit)	TCP-Verbindung, Port 44818 (Standard)

Unterstützte CIP-Services:

Service	Code	Beschreibung

Get Attribute Single	0x0E	Einzelnes Attribut lesen
Set Attribute Single	0x10	Einzelnes Attribut schreiben
Get Attribute All	0x01	Alle Attribute eines Objekts lesen
Read Tag	0x4C	Logix-Tag lesen (Allen-Bradley ControlLogix/CompactLogix)
Write Tag	0x4D	Logix-Tag schreiben

App-Struktur beim Start:

Die App öffnet das **Dashboard** als Startseite. Von dort aus sind alle Funktionen über die untere Navigationsleiste erreichbar.

2. Navigation & Menüstruktur

Die Navigation erfolgt über die **Bottom Navigation Bar** mit fünf Tabs sowie über Floating Action Buttons (FAB, „+"-Symbol) und Icons in der oberen Leiste.

```
App-Start
└── Dashboard (MainScreen)                    ← Bottom Nav: Haus-Icon
    │   ├── [▶ Play]   → Polling starten
    │   ├── [↻ Refresh] → Einmaliges Lesen
    │   └── [Tab-Auswahl] → Gerät wechseln (bei mehreren Geräten)
    │
├── Geräte (DeviceListScreen)                 ← Bottom Nav: Listen-Icon
    │   ├── [+ FAB]    → Neues Gerät (DeviceFormScreen)
    │   ├── [✏ Stift]  → Gerät bearbeiten (DeviceFormScreen)
    │   ├── [🗑 Korb]  → Gerät löschen (Bestätigungsdialog)
    │   └── [Karten-Tap] → Gerätedetail (DeviceDetailScreen)
    │       ├── [+ FAB]  → Neues Tag (TagFormScreen)
    │       ├── [✏ Stift] → Tag bearbeiten (TagFormScreen)
    │       └── [🗑 Korb] → Tag löschen (Bestätigungsdialog)
    │
├── Protokoll (LogScreen)                     ← Bottom Nav: Verlauf-Icon
├── Alarme (AlarmScreen)                      ← Bottom Nav: Glocken-Icon (Badge)
└── Einstellungen (SettingsScreen)            ← Bottom Nav: Zahnrad-Icon
    └── [Diagnose-Button] → DiagScreen
```

3. Dashboard (MainScreen)

Erreichen: App starten – Dashboard erscheint automatisch.

Das Dashboard zeigt alle konfigurierten Tags des aktiven Geräts in Echtzeit. Werte werden als Karten dargestellt.

3.1 Gerät wechseln

Weg: Dashboard → Tab-Leiste (erscheint nur bei mehr als einem Gerät) → Gewünschten Tab antippen

- Beim Tab-Wechsel werden die angezeigten Werte zurückgesetzt
- Das zuletzt aktive Gerät wird für die Session gespeichert

3.2 Einmaliges Lesen

Weg: Dashboard → ⟳**-Button** (oben rechts in der TopAppBar) antippen

Schritt	Bedienelement	Beschreibung
1	Gerät im Dashboard aktiv	(Tab des gewünschten Geräts ausgewählt)
2	Button ⟳ antippen	Alle Tags des aktiven Geräts werden einmalig gelesen
—	Tag-Karten	Werte werden sofort aktualisiert

3.3 Polling starten/stoppen

Weg: Dashboard → ▶**-Button** (oben rechts) → Polling aktiv → ■**-Button** zum Stoppen

Schritt	Bedienelement	Beschreibung
1	Button ▶ antippen	Zyklisches Lesen startet (Intervall aus Einstellungen)
—	Button wechselt zu ■	Zeigt aktiven Polling-Zustand
2	Button ■ antippen	Polling wird gestoppt, laufende Sitzung beendet

Während des Pollings werden Werte automatisch in der Datenbank aufgezeichnet.

3.4 Fehlermeldungen

Bei Verbindungsproblemen erscheint ein **roter Banner** unterhalb der Tab-Leiste mit der genauen Fehlermeldung (z. B. „Connection refused", „Timeout").

3.5 Leerer Zustand

Sind noch keine Geräte konfiguriert, erscheint ein Hinweistext mit direktem **„Gerät hinzufügen"**-Button.

4. Geräteliste (DeviceListScreen)

Erreichen: Bottom Navigation → **Listen-Icon**

Zeigt alle konfigurierten EtherNet/IP-Geräte als Karten. Jede Karte zeigt Name, IP-Adresse, Port und Anzahl der konfigurierten Tags.

4.1 Neues Gerät anlegen

Weg: Geräteliste → **„+"**-Button (unten rechts) → DeviceFormScreen

4.2 Gerät öffnen (Tags anzeigen)

Weg: Geräteliste → Gerätekarte antippen → DeviceDetailScreen

4.3 Gerät bearbeiten

Weg: Geräteliste → ✏**-Icon** (rechts auf der Gerätekarte) → DeviceFormScreen

4.4 Gerät löschen

Weg: Geräteliste → 🗑**-Icon** (rechts auf der Gerätekarte) → Bestätigungsdialog → „Löschen"

> *Das Löschen eines Geräts entfernt auch alle zugehörigen Tags. Aufzeichnungen bleiben in der Datenbank erhalten.*

5. Gerätedetail (DeviceDetailScreen)

Erreichen: Geräteliste → Gerätekarte antippen

Zeigt alle konfigurierten Tags eines Geräts und ermöglicht deren Verwaltung.

5.1 Tag hinzufügen

Weg: Gerätedetail → **„+"**-Button (unten rechts)

- In der **kostenlosen Version** ist der FAB gesperrt (🔒 -Icon) sobald 5 Tags erreicht sind
- Ein Tipp auf den gesperrten FAB öffnet den **Pro-Upgrade-Dialog**

5.2 Tag-Zähler

Der **Zähler** (z. B. `3 / 5`) oben rechts im Gerätekopf zeigt den Fortschritt:

- **Grüner Hintergrund:** Noch Platz verfügbar
- **Roter Hintergrund:** Limit erreicht (kostenlose Version)
- Pro-Version: Kein Zähler sichtbar

5.3 Tag bearbeiten

Weg: Gerätedetail → ✏**-Icon** (rechts auf der Tag-Karte) → TagFormScreen

5.4 Tag löschen

Weg: Gerätedetail → 🗑**-Icon** (rechts auf der Tag-Karte) → Bestätigungsdialog → „Löschen"

5.5 Geräteinformationen

Der Gerätekopf unterhalb der TopAppBar zeigt:

- IP-Adresse und Port

- Backplane-Slot (nur wenn > 0)
- Beschreibung (wenn vorhanden)

6. Gerät anlegen/bearbeiten (DeviceFormScreen)

Erreichen (neu): Geräteliste → „+"-Button

Erreichen (bearbeiten): Geräteliste → ⋮-Icon auf Gerätekarte

oder: Gerätedetail → ⋮-Icon in der TopAppBar

Feld	Pflichtfeld	Beschreibung
Name	✓	Gerätebezeichnung
Beschreibung		Freitextnotiz
IP-Adresse	✓	IP-Adresse des EtherNet/IP-Geräts
Port		TCP-Port (Standard: 44818)
Backplane-Slot		Slot-Nummer im Chassis (0 = direkte Verbindung, kein Routing)

> ***Slot-Hinweis:** Bei ControlLogix-Systemen gibt der Slot an, in welchem Chassis-Slot die CPU sitzt. Bei direkt adressierbaren Geräten (z. B. Drives, Feldbusadapter) bleibt Slot auf 0.*

Speichern: Button „Speichern" (unten) – nur aktiv wenn Name und IP ausgefüllt sind.

Verwerfen: Zurück-Pfeil (←) navigiert zurück ohne zu speichern.

6.2 NFC-Tag beschreiben

Weg: Gerät anlegen/bearbeiten → **NFC-Button** (Chip-Icon, rechts neben „Speichern") → Dialog „Halte das Gerät an einen NFC-Tag" → Tag antippen

> *Der NFC-Button erscheint nur auf Geräten mit NFC-Hardware. Er ist nur aktiv, wenn Name und IP ausgefüllt sind.*

Ablauf:

Schritt	Bedienelement	Beschreibung
1	**NFC-Button** antippen	Warte-Dialog erscheint
2	Android-Gerät an NFC-Tag halten	Tag wird automatisch erkannt
3	—	App schreibt Profil auf den Tag (NDEF, MIME `application/io.github.erginmusa.eip`)

4	—	Dialog schließt; Ergebnis-Banner erscheint (grün = Erfolg, rot = Fehler)
—	**Abbrechen**-Button	Schreibvorgang abbrechen

Gespeichertes NFC-Profil:

```
name        – Gerätename
description – Beschreibung
ip          – IP-Adresse
port        – TCP-Port
slot        – Backplane-Slot
```

6.3 Geräteprofil von NFC-Tag lesen

Wenn die App im Vordergrund ist und ein beschriebener Tag ans Gerät gehalten wird, werden die Felder im Formular automatisch vorausgefüllt. Dies funktioniert auch, wenn die App durch Antippen des Tags geöffnet wird.

7. Tag anlegen/bearbeiten (TagFormScreen)

Erreichen (neu): Gerätedetail → „+"-Button

Erreichen (bearbeiten): Gerätedetail → ✏-Icon auf Tag-Karte

Ein **Tag** repräsentiert einen einzelnen CIP-Datenpunkt (Attribut, Logix-Tag o. Ä.).

7.1 Felder

Feld	Pflichtfeld	Beschreibung
Name	✓	Bezeichnung des Tags
Beschreibung		Freitextnotiz
CIP-Pfad	✓	Adressierung des Datenpunkts (siehe 7.2)
CIP-Service	✓	Lese- oder Schreiboperation (Dropdown)
Datentyp	✓	Wie die Rohdaten interpretiert werden (Dropdown)
Skalierung		Multiplikator: Anzeigewert = (Rohwert × Skalierung) + Offset
Offset		Addition nach Skalierung
Einheit		Angezeigter Text (z. B. °C, bar, rpm)
Warnschwelle		Ab diesem Wert → Warn-Alarm
Alarmschwelle		Ab diesem Wert → kritischer

		Alarm
Schreibwert		Nur bei Schreib-Services sichtbar

7.2 CIP-Pfad Syntax

Format	Beispiel	Bedeutung
`Klasse.Instanz.Attribut`	`1.1.1`	Identity Object, Instanz 1, Attribut 1 (Vendor ID)
`Klasse.Instanz`	`1.1`	Identity Object, alle Attribute (mit Get Attribute All)
`Tagname`	`MyMotorSpeed`	Logix-Symbolischer Tag (Allen-Bradley)

Häufige CIP-Objekte:

Klasse	Hex	Beschreibung
Identity	0x01 (1)	Geräteinformationen: Hersteller, Produktname, Revision
Message Router	0x02 (2)	CIP-Nachrichten-Router
Connection Manager	0x06 (6)	Verbindungsverwaltung
Assembly	0x04 (4)	I/O-Daten-Assemblies
Parameter	0x0F (15)	Geräteparameter

7.3 CIP-Services

Service	Richtung	Geeignet für
Get Attribute Single	Lesen	Einzelnes bekanntes Attribut
Get Attribute All	Lesen	Alle Attribute eines Objekts auf einmal
Set Attribute Single	Schreiben	Einzelnes Attribut setzen
Read Tag	Lesen	Allen-Bradley Logix-Tagname
Write Tag	Schreiben	Allen-Bradley Logix-Tagname

7.4 Datentypen

Typ	CIP-Code	Größe	Beschreibung
BOOL	0xC1	1 Byte	Boolean (0/1)
SINT	0xC2	1 Byte	8-Bit vorzeichenbehaftet

INT	0xC3	2 Byte	16-Bit vorzeichenbehaftet
DINT	0xC4	4 Byte	32-Bit vorzeichenbehaftet
LINT	0xC5	8 Byte	64-Bit vorzeichenbehaftet
USINT	0xC6	1 Byte	8-Bit vorzeichenlos
UINT	0xC7	2 Byte	16-Bit vorzeichenlos
UDINT	0xC8	4 Byte	32-Bit vorzeichenlos
REAL	0xCA	4 Byte	IEEE 754 Float32
LREAL	0xCB	8 Byte	IEEE 754 Float64
STRING	0xD0	variabel	CIP-String (4-Byte-Länge + Daten)
RAW	—	variabel	Hex-Rohdaten

Speichern: Button „Speichern" (unten) – nur aktiv wenn Name und CIP-Pfad ausgefüllt sind.

8. Aufzeichnungen (LogScreen)

Erreichen: Bottom Navigation → **Verlauf-Icon**

Zeigt alle aufgezeichneten Polling-Sitzungen. Die Aufzeichnung startet automatisch mit dem Polling und endet beim Stoppen.

8.1 Sitzungsübersicht

Jede Sitzungskarte zeigt:

- Gerätename
- Startdatum und -uhrzeit
- Enddatum und -uhrzeit (wenn beendet)
- Anzahl gespeicherter Einträge

8.2 Einträge anzeigen (aufklappen)

Weg: LogScreen → Sitzungskarte antippen

- Karte klappt auf und zeigt die **letzten 5 Einträge pro Tag** (Timestamp + Wert)
- Bei mehr als 5 Einträgen erscheint ein Hinweis „… X weitere Einträge (CSV-Export für alle)"
- Erneutes Antippen klappt die Karte wieder zu

8.3 CSV exportieren

Weg: LogScreen → Sitzungskarte → **Teilen-Icon** (↗) → System-Share-Dialog → Ziel-App wählen

CSV-Format:

```
Timestamp,TagId,TagName,Value
2026-05-20 14:23:05.123,uuid-123,"Temperatur","23.5 °C"
2026-05-20 14:23:05.124,uuid-456,"Drehzahl","1450 rpm"
```

Der Teilen-Button ist deaktiviert (ausgegraut) wenn die Sitzung keine Einträge enthält.

8.4 Sitzung löschen

Weg: LogScreen → 🗑**-Icon** auf der Sitzungskarte → Bestätigungsdialog zeigt Datum → „Löschen"

Löschen entfernt auch alle zugehörigen Einträge aus der Datenbank.

9. Alarme (AlarmScreen)

Erreichen: Bottom Navigation → **Glocken-Icon** (mit rotem Badge bei unbestätigten Alarmen)

9.1 Alarmübersicht

Zeigt alle Alarm-Events chronologisch absteigend. Jede Alarmkarte zeigt:

- Gerätename und Tag-Name
- Alarm-Typ (WARN / ALARM) mit farbigem Icon
- Ausgelöster Wert und Schwellwert
- Datum und Uhrzeit
- Bestätigungsstatus

Alarm-Typen:

Typ	Icon	Auslöser
ALARM	Rotes Fehler-Icon	Alarmschwellwert überschritten
WARN	Gelbes Warn-Icon	Warnschwellwert überschritten

9.2 Alarm bestätigen

Weg: AlarmScreen → Alarmkarte → ✓**-Icon** (grüner Haken rechts)

- Alarm bleibt in der Liste erhalten
- Badge-Zähler in der Bottom Navigation verringert sich
- Haken-Icon wechselt zu ausgegrautem Outline-Icon

9.3 Alle Alarme löschen

Weg: AlarmScreen → 🗑**-Icon** (oben rechts in der Toolbar) antippen

Nur sichtbar wenn Einträge vorhanden sind.

10. Diagnose (DiagScreen)

Erreichen: Einstellungen (SettingsScreen) → Diagnose-Button

Protokollanalyse und Statistik aller EtherNet/IP-Kommunikation. Zeichnet automatisch auf wenn aktiviert. Der Screen ist in zwei Tabs unterteilt.

10.1 Aufzeichnung starten/stoppen

Weg: DiagScreen → **▶/⏸-Button** in der Toolbar antippen

- Bei aktiver Aufzeichnung: rotes **„REC"**-Badge im Titel sichtbar
- Ringpuffer speichert die letzten **500 Frames**
- Status wird über `AppPreferences` gespeichert (bleibt nach App-Neustart erhalten)

10.2 Tab „Frames" – Frames anzeigen und filtern

Weg: DiagScreen → Tab **„Frames (n)"** antippen

Schritt	Bedienelement	Beschreibung
1	Filter-Chip **„Alle"**	Alle aufgezeichneten Frames anzeigen
1	Filter-Chip **„OK"**	Nur erfolgreiche Frames anzeigen
1	Filter-Chip **„Fehler"**	Nur Fehler-Frames anzeigen
2	Frame-Karte antippen	Klappt auf → Hex-Dump Request + Response

Frame-Karte zeigt:

- IP-Adresse und Port
- Zeitstempel (HH:mm:ss.SSS)
- Latenz-Badge in Millisekunden
- Erfolgs-Icon (grün) oder Fehler-Icon (rot) mit Fehlermeldung
- Aufgeklappt: Hex-Dump im Wireshark-Stil (16 Bytes/Zeile mit Offset und ASCII)

10.3 Tab „Statistik" – Auswertung

Weg: DiagScreen → Tab **„Statistik"** antippen

Zeigt automatisch:

- **Übersicht:** Gesamt-Transaktionen · Erfolge · Fehler · Erfolgsrate (%)
- **Fehler-Aufschlüsselung:** Timeouts / keine Antwort · Sonstige I/O-Fehler
- **Latenz-Sparkline:** grafische Verlaufskurve mit 200-ms-Schwellenlinie
- **Latenz-Kennwerte:** Min · Ø · P95 · Max (in ms)

10.4 PCAP-Datei exportieren (für Wireshark)

Weg: DiagScreen → **PCAP-Icon** in der Toolbar → Share-Dialog → Ziel-App wählen

Die exportierte `.pcap`-Datei enthält valide Ethernet + IPv4 + TCP + CIP-Header und kann direkt in Wireshark geöffnet werden.

10.5 Frame-Buffer leeren

Weg: DiagScreen → 🗑**-Icon** in der Toolbar → Bestätigungsdialog → „Löschen"

11. Einstellungen (SettingsScreen)

Erreichen: Bottom Navigation → **Zahnrad-Icon**

11.1 Farbschema ändern

Weg: Einstellungen → Abschnitt „Darstellung" → Dropdown „Farbschema"

Option	Beschreibung
System	Folgt der Android-Systemeinstellung (Standard)
Hell	Helles Material-3-Theme (Primärfarbe: Teal)
Dunkel	Dunkles Material-3-Theme

Änderung wird sofort ohne Neustart übernommen.

11.2 Polling-Intervall ändern

Weg: Einstellungen → Abschnitt „Polling-Intervall" → Slider bewegen

- Bereich: 0,5 s bis 30 s
- Aktueller Wert wird in Sekunden über dem Slider angezeigt
- Gilt für alle Geräte

11.3 MQTT-Integration konfigurieren

Weg: Einstellungen → Abschnitt „MQTT-Integration"

Die MQTT-Integration leitet Messwerte aus dem Polling automatisch an einen MQTT-Broker weiter.

Feld	Beschreibung
MQTT aktiviert	Toggle zum Ein-/Ausschalten der gesamten MQTT-Funktion
Broker-Adresse	Hostname oder IP-Adresse des MQTT-Brokers (z. B. `192.168.1.10`)
Broker-Port	TCP-Port (Standard: 1883)
Benutzername	Optional; nur wenn der Broker Authentifizierung erfordert
Passwort	Optional; Eingabe wird verborgen (👁-Icon zum Anzeigen)
Topic-Präfix	Präfix für alle Topics (Standard: `eip`)

QoS-Stufe	0 = At most once, 1 = At least once

Topic-Format: `{Präfix}/{Gerätename}/{Tagname}`

Beispiel: `eip/PLC1/Motortemperatur`

> *Leerzeichen in Geräte- und Tagnamen werden automatisch durch Unterstriche ersetzt.*

Verbindungssteuerung:

Status	Anzeige	Aktion
Verbunden	Grüner Badge „Verbunden"	Button „Trennen"
Verbinde…	Gelber Badge	— (warten)
Getrennt	Grauer Badge	Button „Verbinden"
Fehler	Roter Badge „Verbindungsfehler"	Button „Verbinden"

> *MQTT-Zugangsdaten werden ausschließlich lokal auf dem Gerät gespeichert. Es werden keine Daten an Server des App-Herstellers übertragen.*

11.4 Lizenzstatus

Weg: Einstellungen → Abschnitt „Lizenz"

- **Pro aktiv:** Grünes Verified-Icon + Text „Pro-Version aktiv"
- **Kostenlose Version:** Infotext + „Auf Pro upgraden"-Button

11.5 Über die App

Zeigt Versionsnummer, Entwickler und implementierte Protokolldetails.

12. Lizenzmodell

EtherNet/IP++ verwendet ein **Freemium-Modell** über Google Play In-App Purchases.

12.1 Kostenlose Version

Funktion	Kostenlos
Anzahl Geräte	Unbegrenzt
Tags pro Gerät	Max. 5
Polling	✓
Aufzeichnung	✓
CSV-Export	✓
Alarmierung	✓

Diagnose-Tracer	✓

12.2 Pro-Version

Funktion	Pro
Tags pro Gerät	Unbegrenzt
Alle kostenlosen Funktionen	✓

12.3 Upgrade durchführen

Weg: Einstellungen → Abschnitt „Lizenz" → Button **„Auf Pro upgraden"**

oder: Gerätedetail → 🔒 -FAB antippen → Pro-Dialog → **„Auf Pro upgraden"**

Der Kauf wird über **Google Play Billing** abgewickelt. Produkt-ID: `ethernetplusplus_pro`.

12.4 Kauf wiederherstellen

Der Kauf wird automatisch beim App-Start wiederhergestellt, wenn das Google-Konto mit dem ursprünglichen Kauf verknüpft ist.

13. Datenmodelle

EipDevice

```
id           – UUID (automatisch generiert)
name         – Gerätebezeichnung
description  – Beschreibung
ip           – IP-Adresse (Standard: 192.168.1.1)
port         – TCP-Port (Standard: 44818)
slot         – Backplane-Slot (Standard: 0)
tags         – Liste der EipTag-Einträge
```

EipTag

```
id             – UUID
name           – Bezeichnung
description    – Beschreibung
cipPath        – CIP-Pfad (z. B. "1.1.1" oder Logix-Tagname)
serviceCode    – CIP-Service (GET_ATTRIBUTE_SINGLE u. a.)
dataType       – Datentyp (BOOL, INT, DINT, REAL, STRING ...)
ip             – IP-Override (leer = Gerät-IP verwenden)
port           – Port-Override (0 = Gerät-Port verwenden)
scale          – Skalierungsfaktor (Standard: 1.0)
offset         – Offset nach Skalierung (Standard: 0.0)
unit           – Einheitentext
widgetType     – AUTO / VALUE / LED / GAUGE / SPARKLINE
gaugeMin/Max   – Bereich für Balkenanzeige
thresholdWarn  – Warnschwelle (optional)
thresholdAlarm – Alarmschwelle (optional)
alarmOnChange  – Alarm bei Wertänderung
writeValue     – Vorbelegter Schreibwert
```

14. EtherNet/IP-Protokollimplementierung

14.1 Encapsulation Layer (`EipEncapsulation.kt`)

Der Encapsulation Layer kapselt CIP-Nachrichten gemäß ODVA EtherNet/IP-Spezifikation (Vol. 2).

Header-Format (24 Byte):

```
Byte  0- 1: Command (z. B. 0x0065 = Register Session)
Byte  2- 3: Length (Datenlänge ohne Header)
Byte  4- 7: Session Handle (nach RegisterSession gesetzt)
Byte  8-11: Status (0x00000000 bei Anfragen)
Byte 12-19: Sender Context (8 Byte, wird gespiegelt)
Byte 20-23: Options (0x00000000)
```

Ablauf einer Transaktion:

```
1. TCP-Verbindung zu IP:44818 aufbauen (Timeout: 4 s)
2. Register Session senden (Command 0x0065, Protocol Version = 1)
3. Session Handle aus Antwort extrahieren
4. Send RR Data senden (Command 0x0065) mit CIP-Request im CPF-Wrapper
5. Antwort lesen → CIP-Response aus CPF extrahieren
6. Unregister Session senden (Command 0x0066)
7. TCP-Verbindung schließen
```

> *Jeder Tag-Lesevorgang öffnet eine eigene TCP-Verbindung. Dies entspricht dem Standard für Explicit Messaging ohne persistente Verbindungen.*

Common Packet Format (CPF):

```
Interface Handle (4 Byte, immer 0x00000000 für CIP)
Timeout (2 Byte)
Item Count (2 Byte, immer 2)
  Address Item: Type 0x0000 (Null Address), Length 0
  Data Item:    Type 0x00B2 (Unconnected Data), Length + CIP-Bytes
```

14.2 CIP-Schicht (`CipBuilder.kt`)

CIP-Anfrage-Format:

```
Byte 0:    Service Code (z. B. 0x0E = Get Attribute Single)
Byte 1:    Path Size in Words
Byte 2..n: EPATH (CIP-Pfad)
Byte n+1..: Request Data (nur bei Schreib-Services)
```

CIP-Antwort-Format:

```
Byte 0:    Service Code | 0x80 (gespiegelter Service mit gesetztem Bit 7)
Byte 1:    Reserved (0x00)
Byte 2:    General Status (0x00 = Success)
Byte 3:    Extended Status Size (in Words)
Byte 4...: Response Data
```

EPATH-Kodierung:

Segment-Typ	Format	Beispiel für Klasse 1
Class ID (≤ 0xFF)	`0x20 [Class]`	`0x20 0x01`
Class ID (> 0xFF)	`0x21 0x00 [Lo] [Hi]`	`0x21 0x00 0x00 0x01`
Instance ID (≤ 0xFF)	`0x24 [Inst]`	`0x24 0x01`
Attribute ID (≤ 0xFF)	`0x30 [Attr]`	`0x30 0x01`
Symbolic	`0x91 [StrLen] [Bytes…] [Pad]`	`0x91 0x0B "MyMotorSpd" 0x00`

14.3 CIP General Status Codes

Code	Bedeutung
0x00	Success
0x08	Service not supported
0x09	Invalid attribute value
0x0C	Object state conflict
0x0E	Attribute not settable
0x14	Attribute not supported
0x16	Too much data
0x17	Object does not exist
0x25	Path segment error
0x26	Path destination unknown

14.4 Byte-Reihenfolge

EtherNet/IP / CIP verwendet durchgängig **Little Endian** für alle numerischen Werte (außer bei manchen herstellerspezifischen Implementierungen). Die Dekodierung in `CipBuilder.decodeValue()` verwendet `ByteOrder.LITTLE_ENDIAN`.

14.5 Logix Read Tag (0x4C)

Für Allen-Bradley ControlLogix / CompactLogix:

```
Anfrage:
  Service:    0x4C
  Path:       Symbolic Segment (Tag-Name)
  Data:       Element Count (2 Byte, Little Endian)

Antwort (bei Erfolg):
  Byte 0-3:   CIP-Antwort-Header (Service | 0x80, Reserved, Status,
ExtStatusSize)
  Byte 4-5:   CIP Data Type Code (z. B. 0x00CA = REAL)
  Byte 6-7:   Element Count
  Byte 8-...: Wertdaten (Little Endian, Typ aus Bytes 4-5)
```

15. Datenbank

Room-Datenbank (Version 1) mit drei Tabellen:

log_sessions

Aufzeichnungssitzungen pro Gerät

```
id          – autoincrement (PK)
deviceId    – ID des aufgezeichneten Geräts
deviceName  – Name zum Zeitpunkt der Aufzeichnung
startTime   – Unix-Timestamp (ms)
endTime     – Unix-Timestamp (ms), null wenn aktiv
entryCount  – Anzahl gespeicherter Einträge
```

log_entries

Einzelne Messwerte einer Sitzung

```
id          – autoincrement (PK)
sessionId   – Referenz zur Sitzung
timestamp   – Unix-Timestamp (ms)
tagId       – UUID des Tags
tagName     – Name zum Zeitpunkt der Aufzeichnung
value       – Dekodierter Anzeigewert (mit Einheit)
```

alarm_events

Alarm-Events und deren Bestätigungsstatus

```
id           – autoincrement (PK)
timestamp    – Unix-Timestamp (ms)
deviceId     – UUID des Geräts
deviceName   – Name zum Zeitpunkt des Alarms
tagId        – UUID des Tags
tagName      – Name zum Zeitpunkt des Alarms
value        – Auslösender Wert
threshold    – Schwellwert zum Zeitpunkt des Alarms
type         – WARN / ALARM
acknowledged – Boolean (false = unbestätigt)
```

16. Berechtigungen

Berechtigung	Zweck
`INTERNET`	TCP-Verbindungen zu EtherNet/IP-Geräten (Port 44818) und optionale MQTT-Verbindung zum konfigurierten Broker
`ACCESS_NETWORK_STATE`	Netzwerkstatus prüfen
`POST_NOTIFICATIONS`	Alarm-Benachrichtigungen (Android 13+)
`NFC`	NFC-Tags mit Geräteprofilen beschreiben und lesen (optional, nur auf Geräten mit NFC)

***Hinweis:** `INTERNET` wird für EtherNet/IP-Kommunikation (Port 44818) und – wenn aktiviert – für die Verbindung zum MQTT-Broker verwendet. Es werden keine Daten an Server des App-Herstellers übertragen. `NFC` ist als `required=false` deklariert: die App kann auf Geräten ohne NFC vollständig genutzt werden. `POST_NOTIFICATIONS` wird beim ersten Alarm automatisch angefragt (Android 13+).*

17. Wear OS

Einrichten: Wear-OS-Smartwatch mit dem Android-Gerät koppeln → EtherNet/IP++ läuft automatisch auf der Uhr sobald Daten gesendet werden.

Die Wear-OS-Begleit-App empfängt Messwerte nach jedem Polling-Zyklus:

Anzeigeelemente auf der Uhr:

- Zeitstempel der letzten Aktualisierung (oben)
- Scrollbare Liste aller Tag-Werte (ScalingLazyColumn)
- Pro Eintrag: **Tag-Name** (klein) · **Wert** (groß)
- „Keine Daten" Platzhalter wenn noch keine Werte gesendet wurden

Datentransfer: Wearable Data API, Pfad `/eip/values` – wird nach jeder erfolgreichen Polling-Operation automatisch ausgelöst.

Dokumentation aktualisiert am 2026-05-21 · EtherNet/IP++ v1.0